

Document Version

Final published version

Licence

CC BY

Citation (APA)

Georgescu, C. A., Schalkers, M. A., & Möller, M. (2026). Fully quantum lattice gas automata building blocks for computational basis state encodings. *Journal of Computational Physics*, 549, Article 114595. <https://doi.org/10.1016/j.jcp.2025.114595>

Important note

To cite this publication, please use the final published version (if applicable). Please check the document version above.

Copyright

In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.

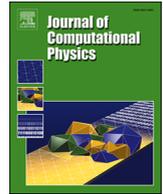
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.



Fully quantum lattice gas automata building blocks for computational basis state encodings

Călin A. Georgescu ^{*}, Merel A. Schalkers , Matthias Möller 

Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, Delft, 2628CD, Zuid Holland, The Netherlands

ARTICLE INFO

Keywords:

Quantum computing
Lattice gas automata
Computational fluid dynamics

ABSTRACT

Lattice Gas Automata (LGA) is a classical method for simulating physical phenomena, including Computational Fluid Dynamics (CFD). Quantum LGA (QLGA) is the family of methods that implement LGA schemes on quantum computers. In recent years, QLGA has garnered attention from researchers thanks to its potential of efficiently modeling CFD processes by either reducing memory requirements or providing simultaneous representations of exponentially many LGA states. In this work, we introduce novel building blocks for QLGA algorithms that rely on computational basis state encodings. We address every step of the algorithm, from initial conditions to measurement, and provide detailed complexity analyses that account for all discretization choices of the system under simulation. We introduce multiple ways of instantiating initial conditions, efficient boundary condition implementations for novel geometrical patterns, a novel collision operator that models less restricted interactions than previous implementations, and quantum circuits that extract quantities of interest out of the quantum state. For each building block, we provide intuitive examples and open-source implementations of the underlying quantum circuits.

1. Introduction

Computational Fluid Dynamics (CFD) has become an indispensable tool across a broad spectrum of disciplines and industries. From civil engineering [1] to aerospace applications [2], and medicine [3], CFD enables the analysis of complex fluid systems for which theory is intractable and experimentation is impractical. With such a broad range of applications, the capabilities that practitioners demand of CFD technologies have increased significantly. For decades, the capacity of CFD methods has grown in tandem with the scale of the available hardware, leading to a surge of practical use cases. However, even at the scale of today's hardware capabilities, many CFD applications that would benefit countless industries remain out of reach due to sheer computational demands. Motivated by these practical limitations, and further fueled by growing concerns about the decreased rate of computational hardware advancement [4], researchers began investigating alternatives to classical CFD methods. One emerging and promising alternative that has received increasing attention in recent years is quantum computing.

Quantum Computing (QC) [5] relies on the exploitation of quantum mechanical phenomena to manipulate basic units of information called quantum bits (qubits). The appeal of QC stems from two fundamental properties of quantum information – superposition and entanglement. Unlike classical bits, a register of n qubits can carry a superposition of 2^n basis states, which can be processed simultaneously in a process known as quantum parallelism. This allows for certain kinds of classical information to be compressed into logarithmically many qubits, which together with entanglement may enable significant computational speedups compared to

^{*} Corresponding author.

E-mail addresses: c.a.georgescu@tudelft.nl (C.A. Georgescu), m.a.schalkers@tudelft.nl (M.A. Schalkers), m.moller@tudelft.nl (M. Möller).

classical algorithms. These advantages of QC have drawn the attention of CFD researchers, leading to the creation of Quantum CFD (QCFD) methods.

The still nascent QCFD research field has branched out into several directions. Variational Quantum Algorithms (VQAs) cast the equations of fluid dynamics onto an optimization framework with the aim of classically optimizing shallow parameterized quantum circuits suitable for near-term noisy quantum devices. Such approaches have proven successful in targeting the advection-diffusion [6] equation, the quasi-1D Navier-Stokes equations [7], and the Poisson equation [8], among others. Recently, quantum realizations of physics-informed neural networks [9] such as [10] and [11] have also shown promising results for different partial differential equations. Though less demanding on the quantum hardware, such approaches incur the drawback of quantum-classical communication in the parameter optimization pipeline. Further hindrances of variational methods include costly function evaluations and the barren plateau problem [12].

A different branch of QCFD research pursues the application of Quantum Linear Solvers (QLSs). QLS algorithms evolve quantum states that encode the solutions of linear algebraic systems of equations, as first introduced by Harrow et al. [13] through a routine that has become known as HHL. Extensions of the HHL algorithm have been applied to develop quantum implementations of the Finite Element Method [14] and the Finite Volume Method [15], as well as to solve the Poisson equation [16]. The greatest appeal of QLS-based algorithms is their exponentially lower computational time with which the solution state can be prepared. However, three obstacles that QLSs face include (i) preparing the necessary starting state to encode the appropriate system of equations (the input problem), (ii) extracting the information out of the prepared quantum state (the output problem), and (iii) the high dependence of the algorithm on the structure of the matrix encoding the system [13,17]. These remain open challenges in the pursuit of practical quantum advantage.

In addition to the development of VQA- and QLS-based solvers, QCFD research has branched out into a third, separate direction – lattice-based discrete velocity methods. Classically, this family of methods includes Lattice Gas Automata (LGA) [18] and the Lattice Boltzmann Method (LBM) [19]. The foundation of these CFD methods lies in the kinetic theory of gases, and their implementations follow different principles than the two other branches of QCFD. Both LGA and the LBM follow a time-marching procedure that tracks the evolution of a system of fictitious particles through phase space. Both LGA and LBM consist of a repeating routine of two cornerstone steps. First, particles travel across discrete *velocity channels* in a step called *streaming*, followed by inter-particle interactions, called *collision* or *scattering*. Two features of these steps that make LGA and the LBM promising candidates for QC implementations are the linearity of streaming and the locality of collision. The former allows for straightforward implementations of particle transport steps in several quantum encodings, while the latter makes it easy to take advantage of quantum parallelism, since collision operators can be applied simultaneously throughout the lattice. Where the LGA and the LBM fundamentally differ is in their model of particles. LGA particles are typically boolean and follow an exclusionary principle, that is, at most one particle can occupy a velocity channel at a time. In contrast, the LBM operates at a larger, mesoscopic scale, which does not track individual particle trajectories but rather the behavior of *populations* of particles interpreted through a global probability distribution. Historically, the LBM has superseded LGA as a fluid dynamics simulation tool, in large part due to the latter's susceptibility to statistical noise. Replacing a discrete number of boolean particles with a smooth distribution function innately circumvents the requirement of stochastic ensemble averaging that LGA algorithms suffer from. For this reason, much recent research has focused on porting LBM primitives to the quantum computing paradigm. In doing so, Quantum LBM (QLBM) research has itself taken three different directions.

The first direction is based on the Linear Combinations of Unitaries (LCU) [20] paradigm, in which complex operators are expressed as sums of unitary operators controlled on the state of ancillary qubits. These approaches include the work of Budinski [21,22], Wawrzyniak et al. [23], and Tiwari et al. [24]. Though these methods allow the lattice information to be encoded in logarithmically many qubits, this compression comes at the cost of an inexact time-evolution operator. Due to this inexactness, each time step of the algorithm introduces states that are orthogonal to the representation of the physical system, an, in turn, significantly increase the number of measurements required to extract meaningful information out of the quantum state. In practice, applying these methods requires a procedure resembling quantum state tomography [25] to implement rejection sampling, as well as re-initialization of the quantum state after each time step [23,25]. Expressing the unitary matrices that perform collision and reinitialization into hardware-native gates introduces further substantial classical overhead.

The second direction of QLBM research has focused on implementing the separate QLBM streaming and collision steps, without the need for the inexact LCU decomposition. This includes the works of Todorova and Steijl [26], Budinski et al. [27], and Schalkers and Möller [28], which detail the implementation of the streaming step, while Steijl [29] and Moawad et al. [30] implement the nonlinear collision step by means of quantum arithmetic. However, Schalkers and Möller [31] and Fonio et al. [32], have shown that the quantum encodings previously used to implement streaming cannot model collision, and vice-versa.

The third direction of QLBM research has been studying applications where the governing Lattice Boltzmann Equation (LBE) either does not necessitate any nonlinear terms in the computation of the collision step, or can be practically linearized. The former includes the works of Xu et al. [25] and Wawrzyniak et al. [33], who design circuits for the application of the QLBM to the advection-diffusion equation. The linearization approach by means of Carleman linearization was studied by Itani et al. [34,35] and Sanavio et al. [36,37], while Kumar and Frankel [38] propose a linear decomposition of the collision operator matrix. To the best of our knowledge, however, expressing the linearized version of the LBE in a manner that is efficiently implementable through the one- and two-qubit gates that are native to quantum hardware while targeting nonlinear flow regimes remains an open challenge.

In light of the challenges facing QLBM development, researchers have simultaneously turned their attention to the several variations of algorithms within the LGA umbrella. In what follows, we introduce the LGA algorithm and the features that make it a prime candidate for quantum implementations, as well as the current state of the art and the challenges we address in this work.

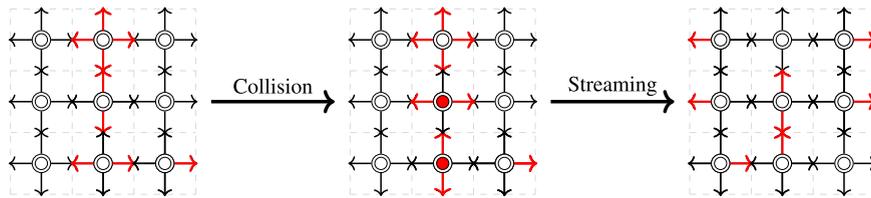


Fig. 1. Overview of the LGA collision and streaming steps.

1.1. Classical and quantum lattice gas automata

The first theory of Cellular Automata (CA) traces back to the seminal work of John von Neumann and Arthur Burks around the middle of the 20th century [39]. CA are models of physical systems in which space is discretized into cells that have a finite number of states, and which evolve and interact with one another according to a set of rules in discrete time. Despite their apparent simplicity, CA are a foundational model of computation that can accurately model numerous physical processes [40]. For the purposes of this work, we use the term Lattice Gas Automata to refer to the subset of CA that model hydrodynamical processes. LGA models have been successfully applied to core problems in hydrodynamics, including the Navier-Stokes equations, as well as related applications to magneto-hydrodynamics [41,42]. Two of the best-known variations of LGA have become known under the HPP [43,44] and FHP [45] acronyms, and model two-dimensional lattices. Fig. 1 shows the time evolution of an LGA system that adheres to HPP model, which prescribes 4 velocity channels in a two-dimensional lattice. Velocity channels that are occupied are highlighted in red, whereas empty channels are black. The LGA loop consists of collision and streaming. Collision locally redistributes the occupancy of velocity channels for each lattice site, such that mass and momentum are conserved. Gridpoints where this collision occurs and is non-trivial are highlighted in red in the middle panel of Fig. 1. Streaming simply propagates the occupancy state of each channel to the neighboring gridpoints, according to the velocity discretization.

The first rough link between CA and quantum computing can be traced back to Richard Feynman's idea of utilizing quantum computers for the purpose of simulating physical processes in 1982 [46]. It was not until the beginning of the following decade that the foundation of Quantum Lattice Gas Automata (QLGA) was established in a series of articles by Meyer [47–51] and Boghosian [52,53], while Succi [54,55] independently established the link between quantum mechanics and kinetic theory.¹ Shortly following this initial wave of research, a series of works by Yezepz introduced QLGA implementations for distributed quantum computer networks [56,57], showing their link to the LBM [58], as well as their suitability for solving the Burgers equation [59] and the many-body Schrödinger equation [60]. Following this wave of research around the turn of the millennium, QLGA research remained stagnant for nearly two decades.

More recently, the interest in QLGA has surged in tandem with its QLBM counterpart. Unlike the LBM, the boolean nature of LGA particles and the exclusionary principle of velocity channel occupancy make linear operations sufficient to express the evolution of the model. In addition, the ensemble averaging that was classically conducted to combat the statistical noise inherent to coarse LGA simulations can be naturally mitigated through the superposition that quantum registers afford. Recent developments of in the QLGA space include the work of Love [61], who introduced quantum circuits for streaming and collision for a two-dimensional lattice discretization, and showed the invariants that emerge as a consequence of the quantum encoding. The encoding used by Love [61] had already been established more than two decades earlier by Boghosian [52,53], who outlined that QLGA can model the Schrödinger equation through propagation and collision rules. Fonio et al. [32] provide additional quantum circuits for collision and measurement in an encoding that shares the same foundational principles as that of Yezepz [57]. Schalkers and Möller [31] introduce the novel Space-Time data encoding and include circuits for streaming and superposed collision for a two-dimensional lattice. Kocherla et al. [62] extend the encoding of Love [61] with a phase estimation protocol to reduce the measurement overhead. Zamora et al. [63] adapt the LGA loop to an encoding that affords exponential memory compression of the grid at the cost of requiring measurement and reinitialization after every time step. Zamora et al. [64] develop a quantum variation of Integer LGA (ILGA) developed by Blommel and Wagner [65] to model fractional collisions between lattice sites that model more expressive integer (as opposed to boolean) occupancy states. Fonio et al. [66] introduce an alternative extension to ILGA by designing a novel collision operator that aims to recover LBM equilibria, implemented through the LCU [20] technique.

The approaches detailed in the most recent QLGA developments by Fonio et al. [66] and Zamora et al. [63,64] all require reinitialization at each time step. Similar to the LCU-based QLBM approaches, this severely hinders the performance of the algorithms by requiring quantum state tomography and state preparation to occur at every iteration. Schalkers and Möller [31] and Fonio et al. [32] showed that such limitations are inherent to the encodings commonly pursued in the QLGA and QLBM literature, and efficient reinitialization remains an open challenge for such approaches. However, encodings where reinitialization is not a necessity *do* exist. Boghosian [52,53] and Love [61] introduced and detailed such an encoding, which utilizes a qubit for each velocity channel in the system. The Space-Time data encoding introduced by Schalkers and Möller [31] is an expanded computational basis state encoding, with additional qubit subregisters that allow for information to propagate from neighboring lattice sites. The number of qubits re-

¹ Quantum Cellular Automata (QCA) and QLGA are distinct classes of algorithms [88]. For a review of QCA, research, we refer the reader to the review of Farrelly [89].

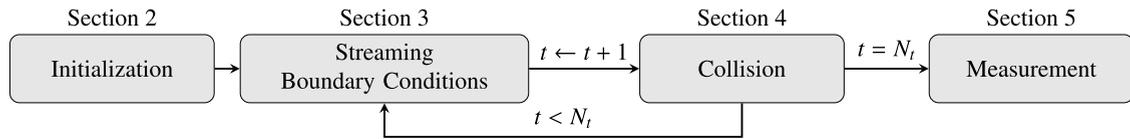


Fig. 2. QLGA algorithm overview for N_t time steps.

quired for information propagation scales with the number of time steps to be simulated, with the upper bound of this scaling reduces to exactly the same encoding used by Boghosian [52,53] and Love [61]. Though this encoding does not allow for any memory compression in the limit of the number of time steps [32,64], it does allow for the simultaneous simulation of exponentially many lattice configurations in parallel by means of quantum superposition.

Though recent advances have increased the capabilities of QLGA algorithms, several challenges persist. In this work, we specifically target three key limitations of current QLGA methods based on computational basis state encodings. First, the literature seldom addresses specific methods for initialization and boundary condition treatment, which are prerequisites for solving real-world applications. Second, the QLGA collision operators proposed in current algorithms are discretization-specific and often operate on heavily restricted state-equivalence criteria. Third, measurement techniques for computing quantities of interest are either formulated for different encodings [67] or utilize costly phase estimation procedures [32]. In this work, we introduce novel quantum circuits building blocks that cater to each step of the QLGA loop and address these three obstacles. We offer different variations of implementations, that trade off expressiveness for efficiency, and detail their capabilities and limitations. We analyze the complexity of each building block in terms of one- and two-qubit gates and accompany theoretical developments with practical examples for small-scale LGA systems.

The structure of the paper follows the layout of the QLGA loop depicted in Fig. 2. Section 2 introduces circuits for initialization followed by streaming and boundary conditions in Section 3, collision in Section 4, and measurement with respect to quantities of interest and forces in Section 5. Section 6 shows the end-to-end application of the introduced building blocks in simulating one- and two-dimensional systems, and Section 7 concludes the paper. The remainder of this section restates the basic properties of the encoding we use throughout, which we additionally generalize to different lattice discretizations and compare to competing methods in the literature.

1.2. The Space-Time data encoding and running examples

Classical LGA systems that model N_g gridpoints with q velocity channels each require qN_g bits to simulate. The QLGA encoding introduced by Boghosian [52,53] and later utilized by Love [61] and Kocherla et al. [62] maps bits and qubits in a one-to-one fashion and as such requires exactly $N_g q$ qubits. By contrast, the sublinear encoding detailed by, *i.e.*, Fonio et al. [32] exponentially compresses the grid into $\lceil \log_2 N_g \rceil$ qubits, that are entangled to a separate register of size q , at the cost of reinitializing the quantum state after each time step. The Space-Time data encoding introduced by Schalkers and Möller [31] can be seen as a compromise between the two previous examples. All three variations have in common that the size of the velocity register is *not* compressed, which allows for the representation of all 2^q local velocity configurations simultaneously.

The versatility of the Space-Time encoding comes from the fact that, unlike the sublinear encoding, the size of the velocity register is a parameter that can be adjusted to allow for the efficient simulation of multiple time steps, at the cost of additional qubits. As the size of the velocity register grows, its upper bound is naturally given by qN_g , which is the same as that of Boghosian [52,53]. The fundamental feature of all three encodings is that they encode velocities using the computational basis state encoding, *i.e.*, they do *not* compress the q velocity channel information into $\lceil \log_2 q \rceil$ qubits, which implies that building blocks can be compatible between algorithms. Since the Space-Time data encoding is formulated in the most general manner out of the three, we introduce all building blocks in this work with respect to this encoding as a baseline. However, since all encodings rely on the same interpretation of the velocity register, the methods we introduce can be adapted to the other two encodings with minimal modifications.

The remainder of this section provides an overview of the properties and the rationale behind the Space-Time data encoding, and introduces two running examples used throughout the remainder of this work. The Space-Time data encoding can be naturally seen as an extension of the computational basis state encoding that circumvents the inherent non-unitarity of streaming in this setting [31]. This is achieved by entangling a grid (or positional) register to a velocity register that grows with the number of time steps to accommodate the information of neighboring gridpoints, thus fundamentally altering the nature of streaming compared to other encodings. The size of the velocity register depends on two factors – the lattice structure and the number of time steps to be simulated.

The first factor is the specific lattice discretization that the simulation is carried under. Particularly, the dimensionality of the system and the number of channels connecting lattice sites determines the so-called *neighborhood structure* of the lattice. This neighborhood structure dictates both the trajectory that discrete particles can travel in the physical grid, as well as the required size and structure of the velocity register. Throughout this work, we distinguish between discretizations using the commonplace $D_d Q_q$ taxonomy, indicating a d -dimensional system with q discrete velocity channels. The second factor is the number of time steps to be simulated. The Space-Time data encoding requires that, for a given number of time steps to be simulated N_t , all information (*i.e.*, particle occupancy of neighboring gridpoints) that can reach a gridpoint within the N_t time steps is contained within the velocity register. In general, however, each gridpoint relies on the information of its $\mathcal{O}(q)$ neighbors for 1 time step, and expanding the neighborhood in the d -dimensional space across N_t time steps envelops $\mathcal{O}(N_t^d)$ gridpoints with q discrete velocity channels each [31].

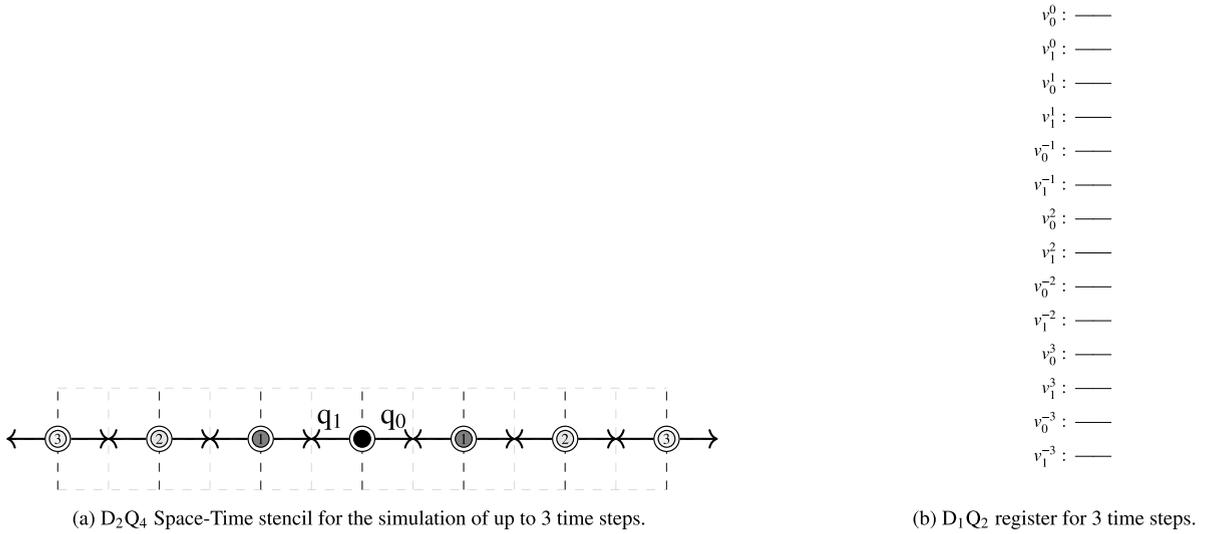


Fig. 3. Space-time data structure and quantum register for the D_1Q_2 discretization.

Naturally, the maximum number of qubits required to perform the simulation for an arbitrary number of time steps is bounded by N_g , the number of lattice sites of the entire grid. In what follows, we provide two concrete instances of qubit registers that encode the information affiliated with the D_1Q_2 and D_2Q_4 discretizations, as well as the exact scaling for the D_3Q_6 stencil.

Example – D_1Q_2 , D_2Q_4 , and D_3Q_6 . We introduce three examples showing how the Space-Time data encoding generalizes to multiple time steps across different discretizations, beginning from the simplest instance of D_1Q_2 depicted in Fig. 3. The D_1Q_2 discretization only distinguishes two channels that particles can travel across in the 1 dimension of the system, to either the left or the right neighboring gridpoint. Though physically trivial, this scenario is small enough to demonstrate the core mechanisms and challenges of the Space-Time encoding, and in this case allows for the simulation of multiple time steps without reinitialization on quantum simulators for commercial classical hardware available today.

The cornerstone data structure of the Space-Time encoding, which we refer to as the *Space-Time stencil*, is shown in Fig. 3, and allows for the simulation of up to $N_t = 3$ time steps. The information contained within the stencil includes the velocity profile of the gridpoint residing at the center of the stencil, which we refer to as the *physical origin*, as well as the velocity profile of all neighboring gridpoints up to a distance of N_t . We label each gridpoint encoded in the stencil with its distance from the physical origin, as this information determines under which circumstances the gridpoint is subject to computation depending on the time step under simulation. The number of gridpoints in the stencil is $2N_t + 1$, and the number of discrete velocity channel occupancies to track is $4N_t + 2$. The corresponding quantum register is depicted in Fig. 3. Each velocity channel is assigned a qubit labeled $v_j^{d_o}$ with d_o its distance from the physical origin, and $j \in \{0, 1\}$ the velocity channel it corresponds to (0 for the positive direction and 1 for the negative direction).

The extension of this encoding to D_2Q_4 and 4 time steps is depicted in Fig. 4, and in general, shares the same fundamental features as von Neumann cellular automata [71]. The number of gridpoints from which information can propagate to the physical origin in N_t time steps scales quadratically with

$$\mathcal{N}_g^{D_2Q_4}(N_t) = \sum_{t=1}^{N_t} 4t = 2N_t^2 + 2N_t. \tag{1}$$

Accounting for the 4 discrete velocity channels and the physical origin itself, the number of particles (and therefore qubits) required to model the system is $4(\mathcal{N}_g^{D_2Q_4}(N_t) + 1) = 8N_t^2 + 8N_t + 4$. Fig. 4 displays the neighborhood of 41 gridpoints, while Fig. 4 shows the quantum register that encodes the data required for the simulation of 4 time steps. Throughout this paper, we order the qubits to align with the typical convention of q_0 signifying the positive direction of the x -axis, and all other indices following a counterclockwise order sorted first by direction (positive or negative) and second by dimension, as shown on the labels of the velocity origins corresponding to the physical origin in Figs. 3 and 4. In the computational basis state encoding of the D_1Q_2 discretization, the state $|01\rangle$ thus encodes a single particle traveling to the right, while a the state $1/\sqrt{2}(|1010\rangle + |0101\rangle)$ of a D_2Q_4 stencil represents a superposition of basis states, where particles are occupying both velocity channels of the x - and y -axes, respectively.

Adding a third dimension with two discrete and opposing velocity channels, one obtains the D_3Q_6 discretization, which we use as a running example for our collision methods. For this 3-dimensional discretization, the number of gridpoints in the Space-Time stencil is equal to the N_t th *Haiiy octahedral number* [72] and is given by

$$\mathcal{N}_g^{D_3Q_6}(N_t) = \frac{(2N_t + 1)(2N_t^2 + 2N_t + 3)}{3}. \tag{2}$$

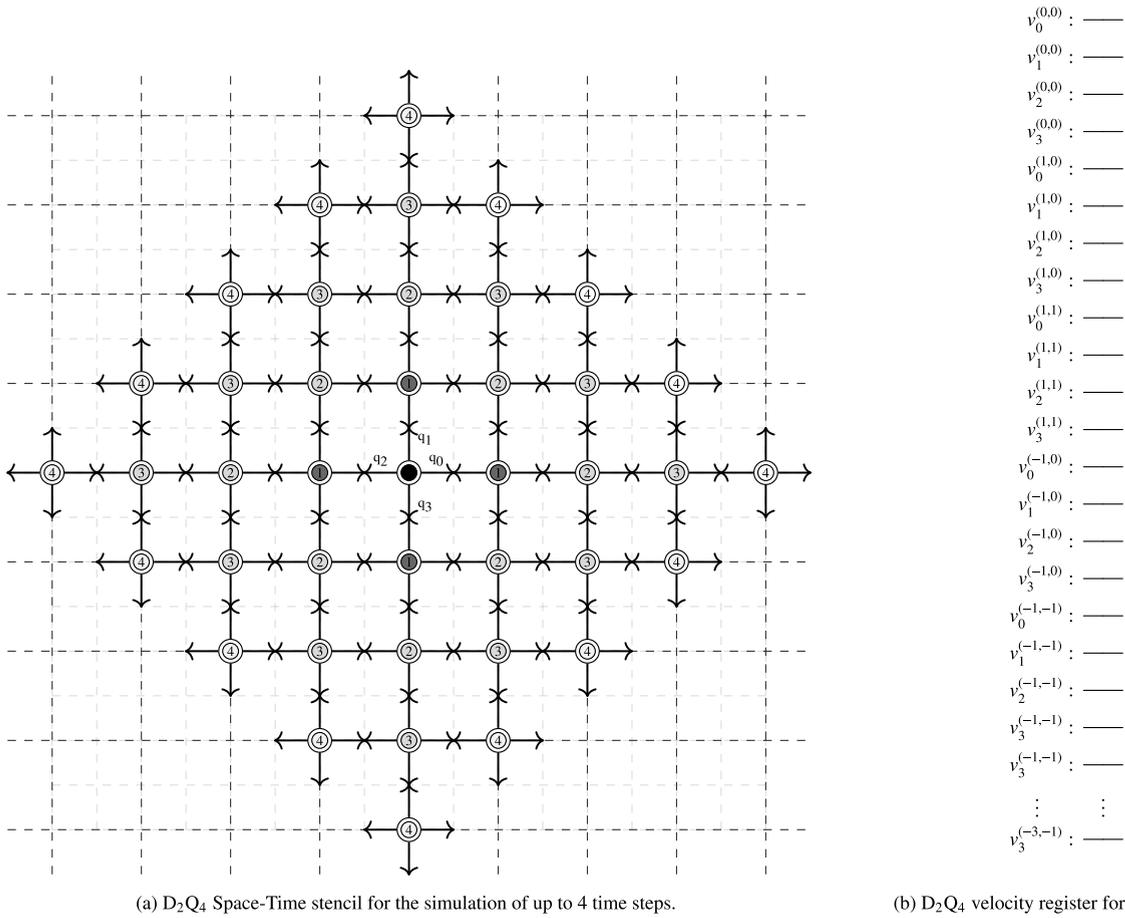


Fig. 4. Space-time data structure and quantum register for the D_2Q_4 discretization.

Since each gridpoint has 6 velocity channels, the entire discretization in turn requires $6 \cdot \mathcal{N}_g^{D_3Q_6}(N_t) = 8N_t^3 + 12N_t^2 + 16N_t + 6$ qubits to model N_t time steps. It is worth noting that in addition to the velocity register, and unless the register reaches its bound of qN_g qubits, the Space-Time data encoding requires additional qubits to entangle the velocity register. We refer to these additional qubits as *grid* or *positional* qubits. Irrespective of the N_t , the number of grid qubits required is

$$n_g = \sum_d \lceil \log_2 N_{g_d} \rceil, \quad (3)$$

with N_{g_d} the number of lattice sites for each dimension of the system.

Complexity analysis assumptions. To accurately capture the computational requirements of the Space-Time QLGA algorithm, we describe the complexity of each introduced building block in terms of *native quantum gates*. By native quantum gates, we generally refer to single- and two-qubit gate sets that are known to be part of universal gate sets [5]. Where possible, we derive the complexity in terms of commonplace, non-parameterized gates such as the X, H, and CX gates. We assume the number of dimensions d , the number of gridpoints in each dimension (i.e. N_{g_x}), the number of discrete velocity channels q , and the number of time steps to be simulated N_t are all variables. For readability, we assume grid operations that act on one dimension of the grid affect $\mathcal{O}(n_g = \sum_d \lceil N_{g_d} \rceil)$ qubits, which is an overestimation in $d > 1$ dimensions. Where no efficient decomposition is known, we assume the decomposition of the unitary matrix requires a number of native quantum gates that grows exponentially with size of the register it is applied on, as first shown by Barenco et al. [73]. We express the complexity of each operation with respect to *one time step*, as the complexity throughout the runtime of the algorithm scales linearly with N_t . Finally, for a unitary matrix U , we use the notation $C^p U$ to indicate that the operation is controlled on p qubits.

2. Initial conditions

The first step in the LGA pipeline is initializing the flow field by setting the occupancy of each lattice according to the simulation specification. The quantum circuit that performs this function is tasked with evolving the initial $|0\rangle^{\otimes N}$ state into the state that encodes

the classically prescribed flow field. Such state preparation procedures are notoriously challenging in quantum computing and often require exponentially complex quantum circuits or approximations to realize. Within LGA in particular, complex initial conditions are paramount for realizing realistic simulations.

As with all building blocks described in this work, the circuits implementing initial conditions should meet two criteria to be of any use. First, they should be expressive enough to encode initial conditions that are adequate for real-world problems and benchmarks. Second, they should be efficient enough both to be constructed using the classical hardware available today, and to justify a potential advantage to executing the quantum algorithms over their classical counterparts. In this section, we propose two methods that favor expressivity and efficiency, respectively. We begin by introducing the expressive *pointwise* method, which allows the assignment of arbitrary velocity profiles to each gridpoint.

2.1. Pointwise initialization

The initial conditions of complex flow fields may necessitate finely detailed velocity profiles that differ even within small neighborhoods of gridpoints. For this purpose, we introduce a quantum primitive for pointwise initialization, that addresses the grid qubit register in a way that caters individually to each gridpoint on the lattice.

Let us assume a D_2Q_4 lattice discretization and consider the situation in which we want to simulate a system where all particles initially reside at one lattice site. We refer to this lattice site as the physical origin. To assign the velocity profile to the one particular gridpoint without affecting any of the other lattice sites, we first apply a layer of X gates to the grid register that transforms the $|x\rangle|y\rangle$ state to the $|1\rangle^{\otimes n_{gx}}|1\rangle^{\otimes n_{gy}}$ state. Then, several layers of C^pX gates act upon the appropriate velocity register to flip the values of the appropriate qubits to $|1\rangle$, where C^pX is a p -controlled X gate. Finally, another layer of X gates resets the grid register to its previous state. To appropriately address the locality encoded in the Space-Time stencil, this procedure is repeated for each lattice site where information from the grid point can propagate to the physical origin within the number of time steps to be simulated. That is, controlled on the position of all neighboring gridpoints that contain the information of the physical origin, the qubits corresponding to the relative position of the physical origin within the neighbor's data structure must be initialized with the same particle occupancy.

Example – D_2Q_4 pointwise initialization. Let us consider a brief example of pointwise initialization that demonstrates the application of the method to an 8×8 D_2Q_4 lattice with $N_t = 1$. We aim to initialize a single gridpoint at location $(1, 5)$ with velocity profile $|1100\rangle$, corresponding to a particle moving in the positive x direction and one moving in the positive y direction. Fig. 5 depicts the quantum circuits that creates such a state. The qubit register spans 6 positional qubits and 20 velocity qubits that encode the information that can travel to the relative origin in one time step. The circuit begins with a layer of H gates that prime the positional qubits in a uniform superposition of $(|0\rangle + |1\rangle)^{\otimes n}$. Following this step, there are 5 layers of operations that perform the same procedure on basis states that correspond to different physical lattice sites. All layers permute the basis states of the grid register such that a particular target state is converted to the $|1\rangle^{\otimes 6}$ state, before setting the appropriate velocity qubits and undoing to first operation. The first layer corresponds to the physical origin and targets the $|1\rangle|5\rangle \equiv |001\rangle|101\rangle$ state, whereas all other operations address the neighbors one distance away in each direction.

Complexity analysis. Though expressive, the pointwise method poses significant drawbacks in terms of efficiency. Each individual operation by itself is efficient, only requiring $\mathcal{O}(n_g)$ X gates to set and unset the state of the grid qubits, as well as $\mathcal{O}(q)$ C^qX gates, which can be decomposed into $\mathcal{O}(qn_g^2)$ CX gates [73]. As most building blocks introduced in this paper, this initialization technique traverses the stencil of the physical origin and initializes the state of each neighbor, thus repeating each operation $\mathcal{O}(N_t^d)$ times to initialize a single grid point. To realize more intricate initial conditions, this operation must be repeated for each individual gridpoint, which therefore results in an overall complexity of $\mathcal{O}(N_g N_t^d (n_g + qn_g^2))$. Under this complexity, any potential advantage obtained by compressing grid information is lost, as the exponential overhead of addressing each basis state sequentially leads to performance comparable to classical serial computers. To address this pitfall of pointwise methods, we introduce a more efficient, though less expressive alternative – *volumetric* initialization.

2.2. Volumetric initialization

The key drawback of the pointwise method is that it does not take advantage of the superposition of the grid qubits and therefore incurs an overhead that is linear the number of physical lattice sites. In what follows, we introduce a more efficient approach that does exploit the superposition of the positional qubits to efficiently perform one operation over arbitrarily many lattice sites. This operation adapts the previous boundary condition implementation by Schalkers and Möller [28] and extends it to the application of initial conditions and the corresponding edge cases that come with the Space-Time encoding. The core idea of volumetric operations is to make use of *quantum comparator circuits* to isolate a desired *volume* of physical space where one operation can be uniformly applied. Quantum comparator circuits make use of the periodicity of binary additions to entangle states that adhere to prescribed bounds to the $|1\rangle$ state of an ancillary qubit. To realize these circuits, we make use of Draper adders based on the Quantum Fourier Transform (QFT), described in [74], and more broadly surveyed in [75]. For a detailed description of how the Draper adder can be utilized as a comparator, we refer the reader to [28]. Within the context of initial conditions, this would be akin to initializing all qubits within a certain volume with the same velocity profile. In this section, we describe how cuboid-shaped volumes can be efficiently implemented using a small number of ancilla qubits.

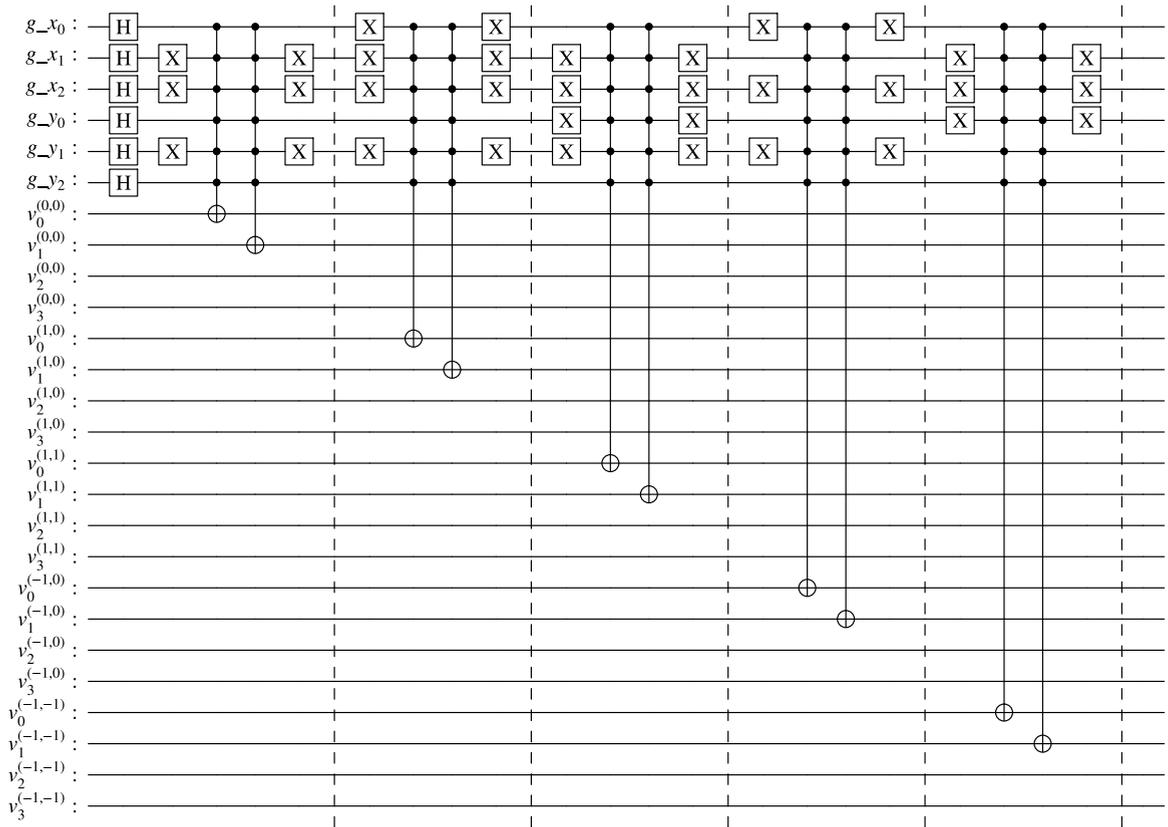


Fig. 5. D_2Q_4 pointwise initialization of the gridpoint at location (1, 5) and velocity profile |1100) for one time step.

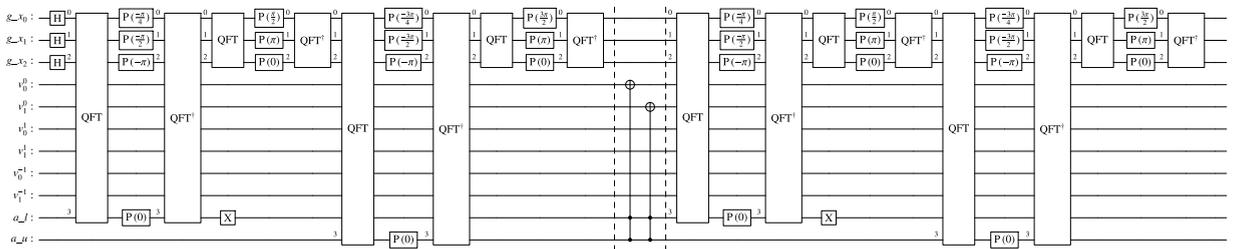


Fig. 6. D_1Q_2 volumetric initialization of the interval [2, 5] and velocity profile |11) for part of one time step.

To model a cuboid-shaped volume, we only require information about its bounds in each dimension. Implementing this in a quantum circuit is straight-forward by utilizing one ancilla qubit per bound and per dimensions, for a total of $2d$ qubits. The purpose of each qubit is to entangle with the positional qubits and encode the information of whether the site lies within (|1) the target volume or not (|0). Then, controlled on the appropriate ancilla qubit(s), we apply the same series of multi-controlled X gates that we used to assign the prescribed velocity profile to a single gridpoint in the pointwise case. The circuit which creates this entanglement is identical to the quantum comparator implementation described by Schalkers and Möller [28] – where our extension differs is in its application to the Space-Time data structure and its edge cases. To illustrate the application of these circuits, let us consider a simple D_1Q_2 example.

Example – D_1Q_2 volumetric initialization. Let us consider the application of volumetric initial conditions to a D_1Q_2 lattice with 8 lattice sites that we aim to simulate for 3 time steps. The initial conditions we aim to initialize are such that gridpoints 2, 3, 4, and 5 all share with the velocity profile |11). The circuit implementing this procedure is given in Fig. 6. To make use of volumetric operations, 2 additional qubits a_l and a_u are appended to the end of the register, to determine whether gridpoints lie within the bounds of the interval. As in the pointwise case, the circuits first primes the grid register to the $|+\rangle^{\otimes 3}$ state such that all gridpoints are encoded,

which leads to the state

$$|\psi_1\rangle = \frac{1}{2^{10}} \sum_{k=0}^7 |k\rangle|0\rangle^{\otimes 6}|0\rangle|0\rangle. \quad (4)$$

Following this, two Quantum Fourier Transform (QFT) based comparator circuits set a_l and a_u to states indicating whether the points belong in the target volume as

$$\begin{aligned} |\psi_2\rangle = & \frac{1}{2^{10}} (|0\rangle + |1\rangle)|0\rangle^{\otimes 6}|0\rangle|1\rangle \\ & + (|2\rangle + |3\rangle + |4\rangle + |5\rangle)|0\rangle^{\otimes 6}|1\rangle|1\rangle \\ & + (|6\rangle + |7\rangle)|0\rangle^{\otimes 6}|1\rangle|0\rangle). \end{aligned} \quad (5)$$

This reflects the fact that (i) the gridpoints at locations 0 and 1 are not greater than the lower bound, but are lower than the upper bound ($|a_l a_u\rangle = |01\rangle$), (ii) gridpoints between 2 and 5 obey both bounds ($|a_l a_u\rangle = |11\rangle$), and (iii) gridpoints 6 and 7 are over the lower bound, but not under the upper bound ($|a_l a_u\rangle = |10\rangle$). Subsequently, the application of the doubly-controlled X gates sets the two velocity qubits that correspond to the origin of the stencil of all gridpoints within the interval to $|11\rangle$, resulting in the state

$$\begin{aligned} |\psi_3\rangle = & \frac{1}{2^{10}} (|0\rangle + |1\rangle)|0\rangle^{\otimes 6}|0\rangle|1\rangle \\ & + (|2\rangle + |3\rangle + |4\rangle + |5\rangle)|1\rangle|1\rangle|0\rangle^{\otimes 4}|1\rangle|1\rangle \\ & + (|6\rangle + |7\rangle)|0\rangle^{\otimes 6}|1\rangle|0\rangle). \end{aligned} \quad (6)$$

The final step consists of reversing the setting of the ancilla qubits by means of the mirrored comparator circuits, which evolves the state to

$$\begin{aligned} |\psi_4\rangle = & \frac{1}{2^{10}} (|0\rangle + |1\rangle)|0\rangle^{\otimes 8} \\ & + (|2\rangle + |3\rangle + |4\rangle + |5\rangle)|1\rangle|1\rangle|0\rangle^{\otimes 6} \\ & + (|6\rangle + |7\rangle)|0\rangle^{\otimes 8}). \end{aligned} \quad (7)$$

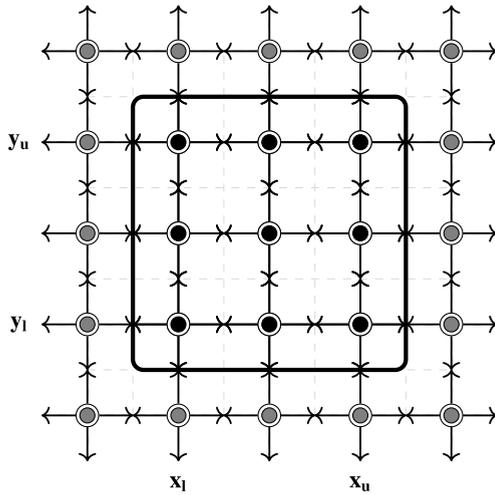
In this final configuration $|\psi_4\rangle$ encodes the desired velocity profile for the physical origin corresponding to the gridpoints in the target interval, without affecting any other basis states. To consistently set the state throughout the entire grid, the circuit shown in Fig. 6 should be adjusted and repeated for all gridpoints encoded in the Space-Time stencil. The only differences in the application of the circuit are that (i) the phases of the P gates are adjusted to fit the different regions of space the gridpoints correspond to and that (ii) the targets of the C^2X gates change accordingly. The advantage of the volumetric initialization method lies in that it can address arbitrarily many physical gridpoints simultaneously. As a consequence of this method, however, several edge cases arise that were otherwise trivial in the pointwise counterpart.

Edge cases – D_2Q_4 initialization. Due to its treatment of contiguous regions of space simultaneously, the volumetric initialization technique requires careful analysis in situations in which information can propagate through periodic boundary conditions. We illustrate this challenge in a D_2Q_4 scenario, and provide a general construction mechanism that works across 1 – 3D spaces. To illustrate the challenge, we use a 5×5 lattice in which we select the gridpoints in the square domain bound by $(x_l, x_u) = (1, 3)$ and $(y_l, y_u) = (1, 3)$ with an arbitrary velocity profile, which is not relevant for this analysis. We further assume that we are interested in simulating this system for $N_t = 4$ time steps. Fig. 7 depicts this system, and the three kinds of edge cases that emerge in this scenario.

Fig. 7a depicts the nominal initialization scenario, in which the gridpoints that the volumetric operation primes all belong to one contiguous region, and the periodic boundary conditions do not affect initialization. In this instance, the 4 ancilla qubits encoding conditions listed in Table 1 act as controls for the corresponding CX gate(s) simultaneously. A more complex scenario occurs when initializing the volume of gridpoints that are at a distance of $(+2, 0)$ from the physical origin, as shown in Fig. 7b. Under these conditions, the gridpoints belonging to the volume are split by the x -boundary of the domain, and the 4 conditions describing the square are no longer sound, as now $x_l > x_u$.

Notably, the same encoding over the 4 ancilla qubits, together with minimal changes to the quantum circuit are sufficient to fit such a scenario. The two volumes can now be addressed separately, by using a subset of the 4 conditions for each of them, as listed in Table 2. The CX gate(s) that instantiate the desired velocity profile onto the part of the volume delimited by the solid line are controlled on all qubits except for the one encoding $x_u \geq x$, as the upper bound is now the boundary of the domain in the x dimension, and therefore all gridpoints (including the ones in the target domain) implicitly adhere to this condition. Symmetrically, the remainder part of the domain no longer requires the $x_l \leq x$ condition to hold, and therefore controls are only placed on both bounds of the non-overflowing dimension and on the upper bound of the overflowing dimension, x_u . Using this method, the comparator circuits used for the scenario of Table 1 are structurally similar, except for the positional shift, and the only difference in complexity stems from the different utilization of the CX gate(s).

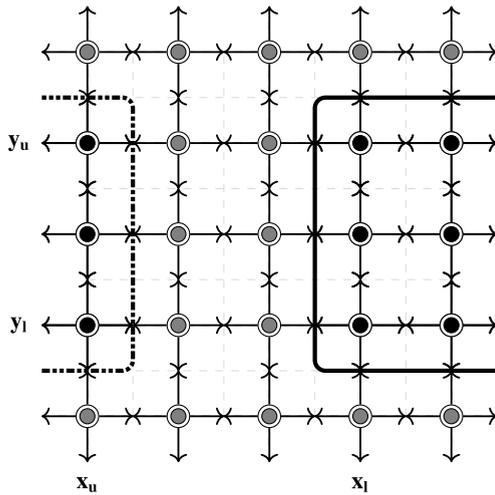
The same straightforward reasoning about the boundaries of the domain enable the efficient initialization of the flow field in the most complex scenario that can occur in two dimensions, described in Fig. 7c and Table 3. This instance corresponds to initializing the grid qubits distanced $(+2, +2)$ from the physical origin, which is the farthest away that information can propagate within 4 time steps. Under these circumstances, the contiguous volume is split into 4 segments, each of which can be characterized by two conditions.



Condition	Domain
$x_l \leq x$	✓
$x_u \geq x$	✓
$y_l \leq y$	✓
$y_u \geq y$	✓

Table 1: Conditions determining the domain.

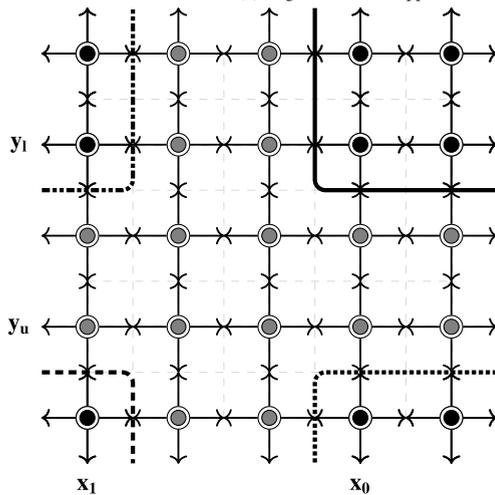
(a) Nominal case for the application of volumetric initial conditions.



Condition	Domain	
		
$x_l \leq x$	✓	
$x_u \geq x$		✓
$y_l \leq y$	✓	✓
$y_u \geq y$	✓	✓

Table 2: Conditions determining the domains.

(b) Edge case for the application of boundary conditions with periodic overflow in 1 dimension.



Condition	Domain			
				
$x_l \leq x$	✓	✓		
$x_u \geq x$			✓	✓
$y_l \leq y$	✓			✓
$y_u \geq y$		✓	✓	

Table 3: Conditions determining the domains.

(c) Edge case for the application of boundary conditions with periodic overflow in 2 dimensions.

Fig. 7. Comparison of edge cases for D_2Q_4 volumetric initialization.

The realizations of this circuit only requires 4 consecutive applications of C^2X gates per segment. In general, the number of overflows that can occur is at most 2^d , as the domain can overflow in each dimension independently, and is therefore not linked to the velocity discretization. For a domain that has overflowed in $0 \leq o \leq d$ dimensions, the velocity profile can be instantiated onto the volume by means of 2^{o+1} multi-controlled X, each controlled on $2d - o$ qubits.

Complexity analysis. Generalizing the previous example to an arbitrary $D_d Q_q$ discretization, the operation of setting one relative neighbor to the appropriate velocity profile requires $2d$ comparator operations, each consisting of one application of the QFT and one application of the QFT^\dagger . Both of these operations are applied to $n_g + 1$ qubits and can therefore be decomposed into $\mathcal{O}(n_g^2)$ Hadamard and controlled phase shift gates [76]. In addition to the QFT layers, the comparators require an additional $\mathcal{O}(n_g)$ phase gates, which does not affect the order of the scaling. There are at most 2^d controlled X gates that set the velocity profile of each gridpoint, which require $\mathcal{O}(d)$ controls each. Therefore, the $\mathcal{O}(2^d)$ $C^d X$ gates of this step can be decomposed into $\mathcal{O}(2^d d^2)$ CX gates each, following the decomposition of Barenco et al. [73]. To obtain a consistent quantum state, the application should be repeated for all gridpoints in the space-time data structure from which information can reach the origin, incurring a cost of N_t^d , much like the pointwise method. The key difference lies in that, unlike the pointwise method, the volumetric initialization technique addresses all gridpoints in the target volume simultaneously, requiring only one traversal of the stencil instead of the N_g . Thus, the cumulative complexity of the volumetric initialization technique requires $\mathcal{O}(N_t^d (n_g^2 + 2^d d^2))$ native quantum gates. Finally, we note that the lower computational cost of the volumetric application of initial conditions comes at the expense of expressiveness. For volumetric operations to be applicable, the entire volume must be initialized to exactly the same velocity configuration for all gridpoints contained within.

3. Streaming and boundary conditions

In this section, we address the streaming and boundary condition steps of the LGA loop, as they both account for the movement of particles across the discrete velocity channels. Streaming is particularly efficient in the Space-Time encoding, as the extended computational basis state allows for particle occupancy sites to propagate by means of swap gates, as described in Section 4 of the original paper [31].

The core concept behind the generalization of the streaming procedure is that of a so-called *streaming line*. A streaming line in the Space-Time stencil consists of the qubits that encode the same velocity channel c of neighboring gridpoints linked by c and its opposing velocity channel \bar{c} . The $D_1 Q_2$ discretization contains a single streaming line, while $D_2 Q_4$ includes $4(N_t - 1) + 2$ streaming lines, as each new time step introduces 4 additional streaming lines and increases the size of each existing streaming line by 2. Generally $D_d Q_q$ discretizations require that each gridpoint is traversed by $\lfloor q/2 \rfloor$ streaming lines per gridpoint due to the potential inclusion of rest particles. Since particles only travel along discrete velocity channels once per time step, each additional time step of the simulation requires $\mathcal{O}(q)$ additional swaps per streaming line, for each of the $\mathcal{O}(N_t^d)$ gridpoints that reside at the edges of their respective streaming lines, leading to a total number of swap gates that scales with $\mathcal{O}(q N_t^d)$. Since streaming lines are independent, the same arrangement of swap gates described in the original paper for $D_2 Q_4$ [31] can be applied per each direction of the streaming line, leading to a logarithmic circuit depth.

Conceptually, the application of boundary conditions in LGA is closely related to streaming, as particles are simply transported in physical space along velocity channels. Unlike in streaming, however, the precise trajectory particles traverse during their interaction with solids depends on the shape of the geometry, and the particular kind of boundary condition that is prescribed (*i.e.* bounce-back or specular reflection). Moreover, the implementation of this subroutine must preserve the locality and reversibility that the Space-Time encoding relies on. In the remainder of this section, we address how bounce-back and specular reflection boundary conditions around rigid bodies can be implemented in the Space-Time encoding, and introduce both pointwise and volumetric building blocks.

3.1. Pointwise imposition of boundary conditions

The application of boundary conditions using the pointwise method follows the same rationale detailed in Section 2.1, where each gridpoint is addressed sequentially, and each relative position of the Space-Time stencil is treated individually. The implementation differs, however, in the operation that takes place at each gridpoint. Where initial conditions simply require that the state of the qubit relating to one particular velocity channel is conditionally flipped, boundary conditions necessitate that the information is transferred from one gridpoint to another. Implementing such an operation is equivalent to realizing a controlled streaming step with subtly different semantics. The circuit that performs such an operation can be realized by means of a multi-controlled swap operation for pre-determined velocity channels.

Example – $D_2 Q_4$ square. Fig. 8 depicts the application on boundary conditions on a 6×6 $D_2 Q_4$ grid where particles reflect off a solid square obstacle spanning $(1, 4) \times (1, 4)$. The physical grid is shown in Fig. 8, where red gridpoints exchange particle occupancy information across the y -axis channels, while gridpoints colored blue do so along the x -axis. Purple gridpoints are at the corners of the object, and exchange information through both axes. For instance, the red gridpoint at $(1, 2)$ only reflects the particle that traveled inside the object through $(0, 2)$, while the blue gridpoint at position $(2, 1)$ exchanges information with its neighbor at $(0, 1)$.

Fig. 8 tracks this interaction in the Space-Time stencil, where v_0 and v_1 are the two particles described in the previous scenario *before streaming*. Following streaming, v_j is swapped onto the qubit corresponding to its neighbor, v'_j , which in this case is positioned inside the solid obstacle. The boundary condition circuit then performs a second swap between v'_j and v''_j , placing the particle onto the opposite direction of the same streaming line of the neighbor it originated from. Since there are no particles inside the object before

The feasibility of specular reflection. The application of post-streaming swaps between fixed points in the Space-Time stencil can be tweaked to implement various kinds of LGA boundary conditions. Bounce-back boundary conditions are applicable in the Space-Time encoding in all common $D_d Q_q$ discretizations, as following reflection against the wall through channel c , particles occupy channel \bar{c} of the same gridpoint, which guarantees that information is contained within the locality of the stencil. Specular reflection boundary conditions, on the other hand, require that the trajectory of particles is reversed only in the dimension(s) in which the particle made contact with the solid surface (*i.e.*, the direction(s) normal to the surface of contact). Whether this velocity channel belongs to a gridpoint already encoded in the stencil is dependent on the discretization. While commonplace discretizations like $D_2 Q_9$ are fully suitable for specular reflection in the Space-Time encoding, Stencils that contain diagonal channels that connect gridpoints in such a way that the gridpoint at which the particle arrives post-collision is not connected directly to the origin by another channel are less suitable. The $D_3 Q_8$ model proposed by [78] is one such discretization. To accommodate models with these properties, the notion of locality in the stencil has to be extended such that all gridpoints affected by streaming and boundary conditions within N_t time steps are reachable. This does not, however, affect the Space-Time encoding in the limit, where all velocity channels in the lattice are allotted a qubit. In this instance, sound reflection semantics of many kinds are efficiently implemented by means of regular (non-controlled) SWAP gates, irrespective of the discretization.

Complexity analysis. As is the case of initial conditions, pointwise boundary conditions can be used to implement arbitrary geometrical shapes under arbitrary discretizations, at the expense of performance. To reflect 1 particle between two positions in the Space-Time stencil, the circuit requires $\mathcal{O}(n_g)$ X gates twice to prepare and revert the $|1\rangle^{\otimes n_g} |v\rangle$ state, as well as 2 CX gates and one C^{n_g+1} X gate to decompose the C^{n_g} SWAP gate, as described in [77], for a total of $\mathcal{O}(n_g^2)$ native gates. Each particle reflection must take place in each stencil where the pair of points that exchange information are both present. For $D_2 Q_4$, the number of pairs is

$$2N_t + 4 \sum_{t=1}^{N_t} t = 2N_t^2 + 2N_t - 2, \quad (8)$$

as each additional time step to be simulated adds 2 additional pairs of points for the previous layer of the stencil, as well as 2 additional pairs at the extremities. In general this number scales again with $\mathcal{O}(qN_t^d)$. This procedure should in general be repeated for the $\mathcal{O}(q)$ velocity channels that particles travel on, as well as for each grid point at the perimeter of the solid domain. Assuming there are N_p such perimeter points, and $N_p \leq N_g$, the overall complexity of the boundary condition step is $\mathcal{O}(N_p N_t^d q(n_g + n_g^2))$.

The locality of the Space-Time encoding affords several techniques that in practice can diminish the complexity of the circuits that implement boundary conditions. First, any pair of points in the Space-Time stencil that would be subject to boundary condition treatment, but which are entangled to gridpoints inside of the solid domain can be neglected. Since information is always local to the stencil, the state of the velocity qubits at that grid location remains $|0\rangle^{\otimes \mathcal{O}(N_t^d)}$, and the swaps are therefore pointless. The same argument can be used to reduce the complexity of the initial conditions described in the previous section, as any velocity information entangled to positional qubits that encode positions inside the solid domain can be skipped. Finally, there are instances in which the application of the pointwise method requires $\mathcal{O}(N_g)$ repetitions. Two such instances include (i) situations in which the space-time stencil is large enough to approximate a sizeable proportion of the entire physical domain, and (ii) cases in which the surface of the solid geometry covers a large area of the domain. In cases where the stencils are relatively small and the geometry is enclosed to a small volume of the domain, pointwise boundary conditions may prove practically feasible. For instances where this is not the case, we next introduce the volumetric alternative to boundary conditions, which significantly improves scaling.

3.2. Volumetric imposition of boundary conditions

The application of volumetric operations to boundary conditions follows the same rationale as in Section 2.2 – each volume of gridpoints where the same operation is simultaneously addressed after isolating it by means of comparator operations. Similar to initial conditions, this operation must be performed for each volume where the information is present in the Space-Time stencil. Unlike initial conditions, however, the properties of this volume are dependent on the shape of the solid domain and the lattice discretization. In general, the volumes affected by boundary conditions are only concerned with the gridpoints that lie at the interface between the solid and the fluid domains. As a consequence, this volume is at most $(d - 1)$ -dimensional, which decreases both the number of qubits and the complexity of the circuits that implement these operations.

Fig. 10 shows the edge cases volumetric operations must account for in axis-aligned geometry for a $D_2 Q_4$ discretization. As before, gridpoints highlighted in red exchange information along the x -axis, while blue gridpoints do so in the y -axis. The gridpoints surrounded by the dotted lines highlight points inside the solid domain that particles stream into and therefore require boundary treatment. Importantly, since these points belong to a line on the y -axis (x -axis), all relative neighbors in the space-time stencil that they exchange information with also belong to a similar line. Therefore, the use of volumetric operations described in Section 2.2 generalizes to boundary conditions in a straightforward way, as the same techniques can be applied to isolate the gridpoints where boundary interactions occur. Unlike initial conditions, however, boundary conditions address volumes that are $(d - 1)$ -dimensional, and can overflow therefore occurs in at most $(d - 1)$ (as opposed to d) dimensions. In Fig. 10, gridpoints surrounded by the dashed outline show contiguous volumes of space in the nominal case, while gridpoints surrounded by solid lines (corresponding to distances $(+2, +4)$ and $(-2, -4)$, respectively) highlight how overflow can occur on the physical grid.

Fig. 11 shows the quantum circuit that performs the volumetric imposition of boundary conditions for the gridpoints surrounded by the red dashed line in Fig. 10. This volume spans the interval $[2, 5]$ on the y -axis and has a fixed position of 6 on the x -axis. Up to the

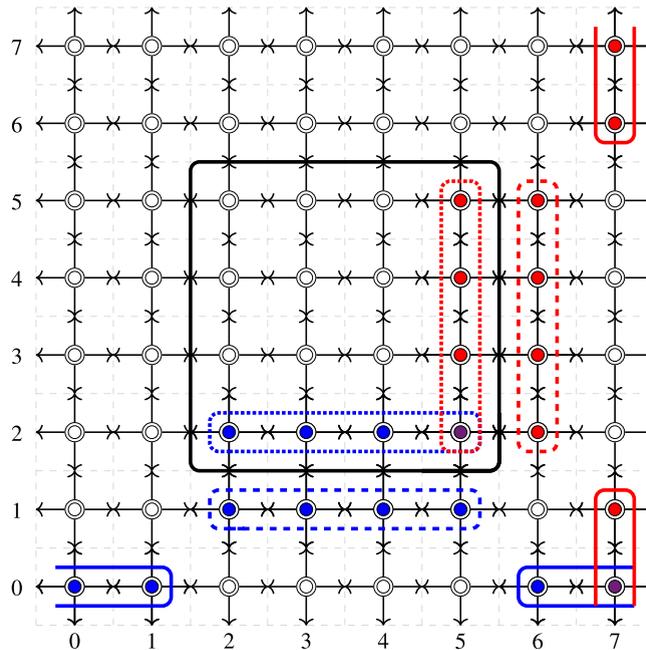


Fig. 10. D_2Q_4 volumetric reflection cases for a square spanning $(2, 5) \times (2, 5)$ for up to 5 time steps.

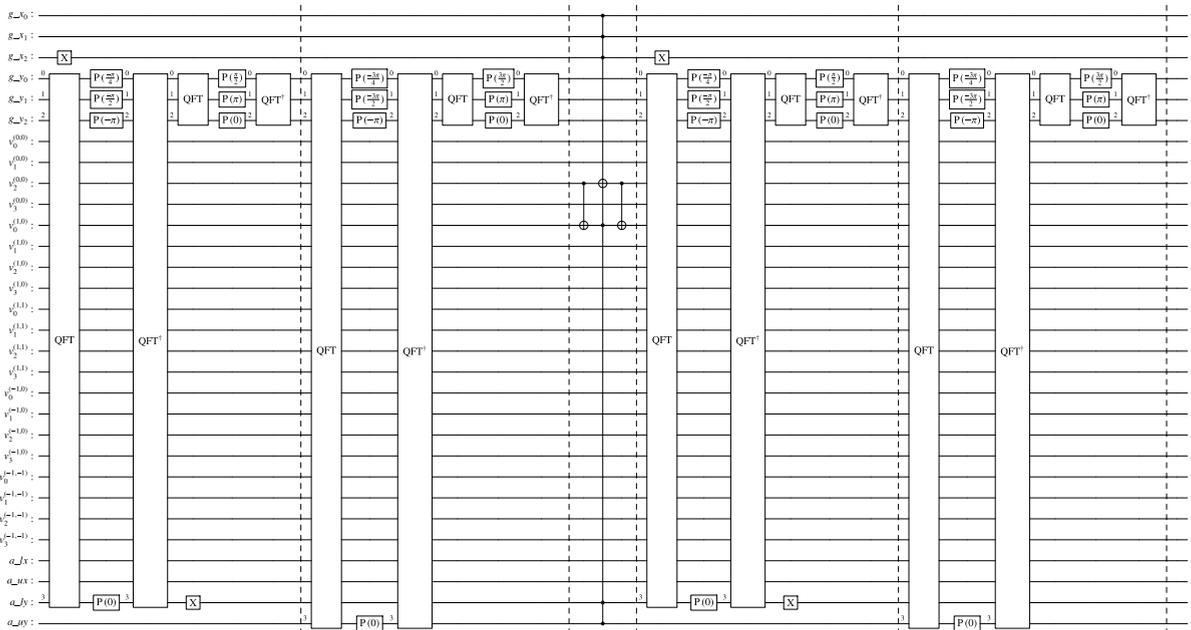


Fig. 11. D_2Q_4 volumetric reflection for the wall of a square spanning $(2, 5) \times (2, 5)$.

second barrier, the circuit sets the a_y and a_x ancillae to the appropriate state by means of the QFT-based Draper comparator, while the X gate on the g_{x_2} converts the state of the x grid qubits from $|6\rangle \equiv |110\rangle$ to $|111\rangle$ in preparation for the multi-controlled operation. Following this preparation step, the same multi-controlled swap decomposition outlined in Section 3.1 performs the swap operation that places the particles back in the fluid domain. Finally, the preceding operations are reversed. The edge cases outlined by the solid demarcations are addressed identically as described in Fig. 7 i.e., the same ancilla qubits act as controls in multiple subsequent swaps. Following the same line of reasoning, the volumetric imposition of boundary conditions require $\mathcal{O}(N_s N_r^d q(n_g^2 + 2^{d-1}(d-1)^2))$ one- and two-qubit gates, where N_s is the number of axis-aligned segments that are subject to boundary treatment. The additional q term accounts for the fact that, depending on the stencil, multiple operations may be required per segment to traverse the corresponding physical space for each reflected channel.

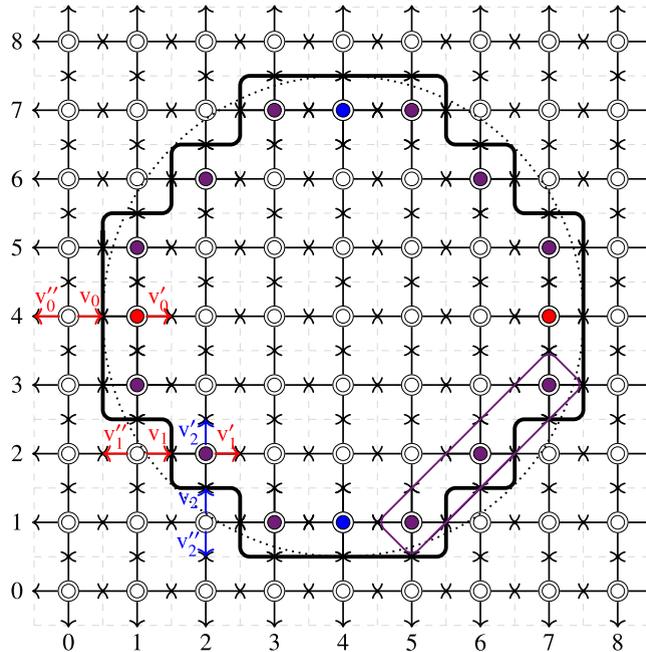


Fig. 12. Discretized approximation of a circle centered at (4, 4) with a radius of 3.5 gridpoint intervals on a 9×9 D_2Q_4 lattice.

While the volumetric methods introduced thus far decrease the complexity of initialization and boundary treatment, they only allow for these operations to be applied in volumes bound by axis-aligned segments. In what follows, we introduce extensions that broaden volumetric operations to more complex shapes.

3.3. Staircase approximations

To overcome the limitations of axis-aligned volumetric operations, we require quantum circuits that are capable of efficiently describing the components of more intricate shapes. In this section, we introduce primitives that isolate diagonal segments on a 2-dimensional grid, providing similar scaling to the axis-aligned counterparts. To demonstrate the practical utility of these primitives, we consider an instance of a circle discretization, which shows how the combination of axis-aligned and diagonal segments can describe more complex shapes.

Consider the instance displayed in Fig. 12, where we are concerned with reflecting particles off a circle centered at (4, 4) on a 9×9 grid. The dotted line displays the continuous shape of the circle (with a radius of 3.5 gridpoint intervals), while the solid line traces its discretized approximation. We refer to this method of encoding solid objects with smooth boundaries into the D_2Q_4 grid as a *staircase approximation*. As before, the points highlighted in red are subject to reflection across the horizontal channel, whereas gridpoints in blue reflect only the vertical channel. Purple gridpoints reflect both. Of note in this staircase approximation of a circle is that points belong to only one of three categories. The first two categories are the axis-aligned segments described in previous sections, which include only the 4 red and blue points. The third category includes segments of purple points which, importantly, are always situated on diagonal segments.

This observation calls for the design of a primitive that, as with axis-aligned segments for cuboid objects, can efficiently separate diagonal points from non-diagonal points irrespective of the segment’s size. To design such a primitive, we note that a diagonal segment is described by 3 characteristics – a lower bound, an upper bound, and an increment. The increment describes the discrete change in physical space that when iteratively applied to the lower bound, yields all points of the segment, up to the upper bound. Assuming an explicit ordering of the bounds, such as clockwise, can make the increment implicit. Consider the diagonal highlighted in purple in Fig. 12. This segment would be described by the lower bound (5, 1), the upper bound (7, 3), and the increment (+1, +1). In what follows, we describe a quantum circuit implementation that isolates the qubits of this segment by setting ancilla qubit a_d to $|1\rangle$.

Fig. 13 shows the quantum circuit. For brevity, we omit the velocity qubits and the exact gates that implement the addition and subtractions, as their structure is identical to the Draper adders shown in previous examples. To ensure the lower bounds are adhered to, the circuit begins by subtracting x_l and y_l from their respective registers, effectively offsetting the diagonal such that it starts at the origin. This is an alternative way of implementing the lower bounds, which uses a similar number of gates to the comparator, but does not require any ancilla qubits. Next, two comparator operations set the state of the upper bound ancilla qubits to $|1\rangle$ in the same way as for the axis-aligned case, except that upper bounds must be adjusted as *i.e.*, $x_u - x_l$ to account for the lower bound subtraction. To isolate gridpoints where the condition $x = y$ holds, 2 adders perform the addition of x and $-y$ onto a third ancilla

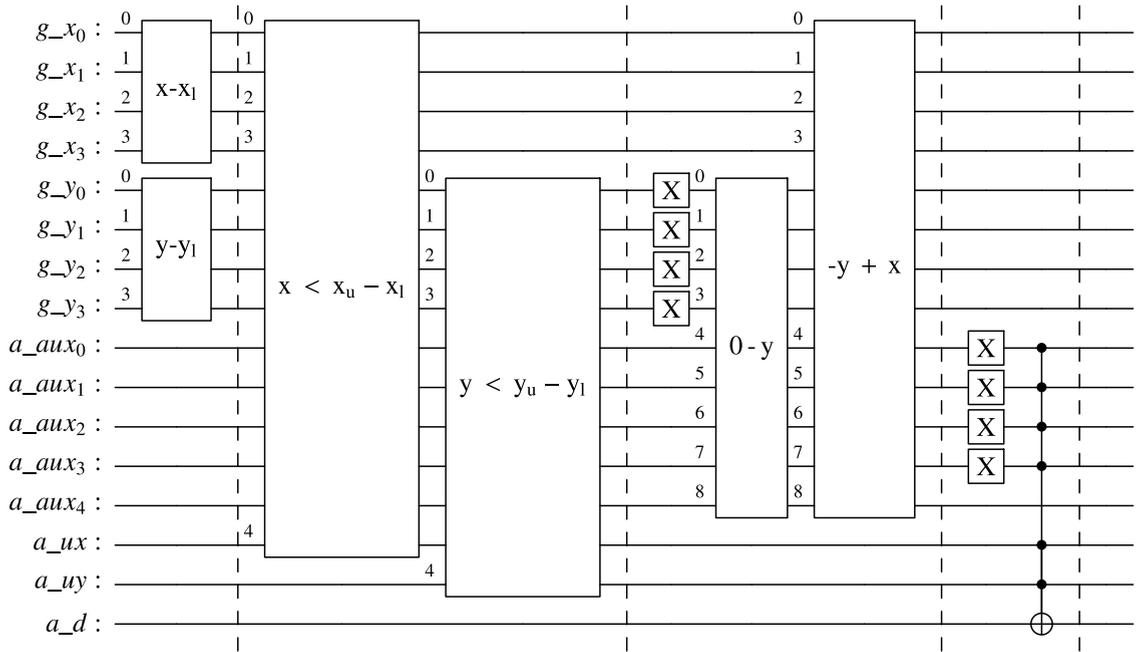


Fig. 13. Volumetric identification operation of the a diagonal segment.

register a_{aux} , which we use to encode the number $|x - y\rangle$. Points belong to the diagonal we are targeting only if $x = y$, and therefore $|a_{aux}\rangle = |0\rangle^{\otimes 5}$. Finally, we invert the state of the auxiliary register such that the state $|1\rangle^{\otimes 4}$ is entangled with the gridpoints belonging to the diagonal, and perform a C^6X operation controlled on the auxiliary register and the upper bound qubits and targeting the a_d qubit. Under this state, the same controlled swap operations of previous sections can be employed to realize the boundary conditions on all points of the diagonal simultaneously, before undoing each addition to reset the grid state.

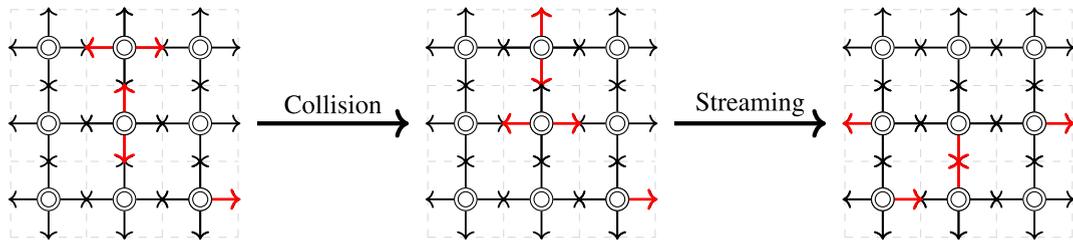
Complexity analysis. The complexity of this single operation is dominated by the 2 Draper adders that compute the $x - y$ state on the auxiliary register. Assuming the same QFT-based implementation is employed, each adder requires $\mathcal{O}(4n_g^2)$ gates, as the size of the auxiliary register is one qubit larger than the largest physical register, to account for potential overflow. Extending this circuit to cover the entire circle, we obtain a bound of $\mathcal{O}(N_s N_t^d q(4n_g^2 + 2^{d-1}(d - 1)^2))$, which is similar to the axis-parallel segments.

4. Collision

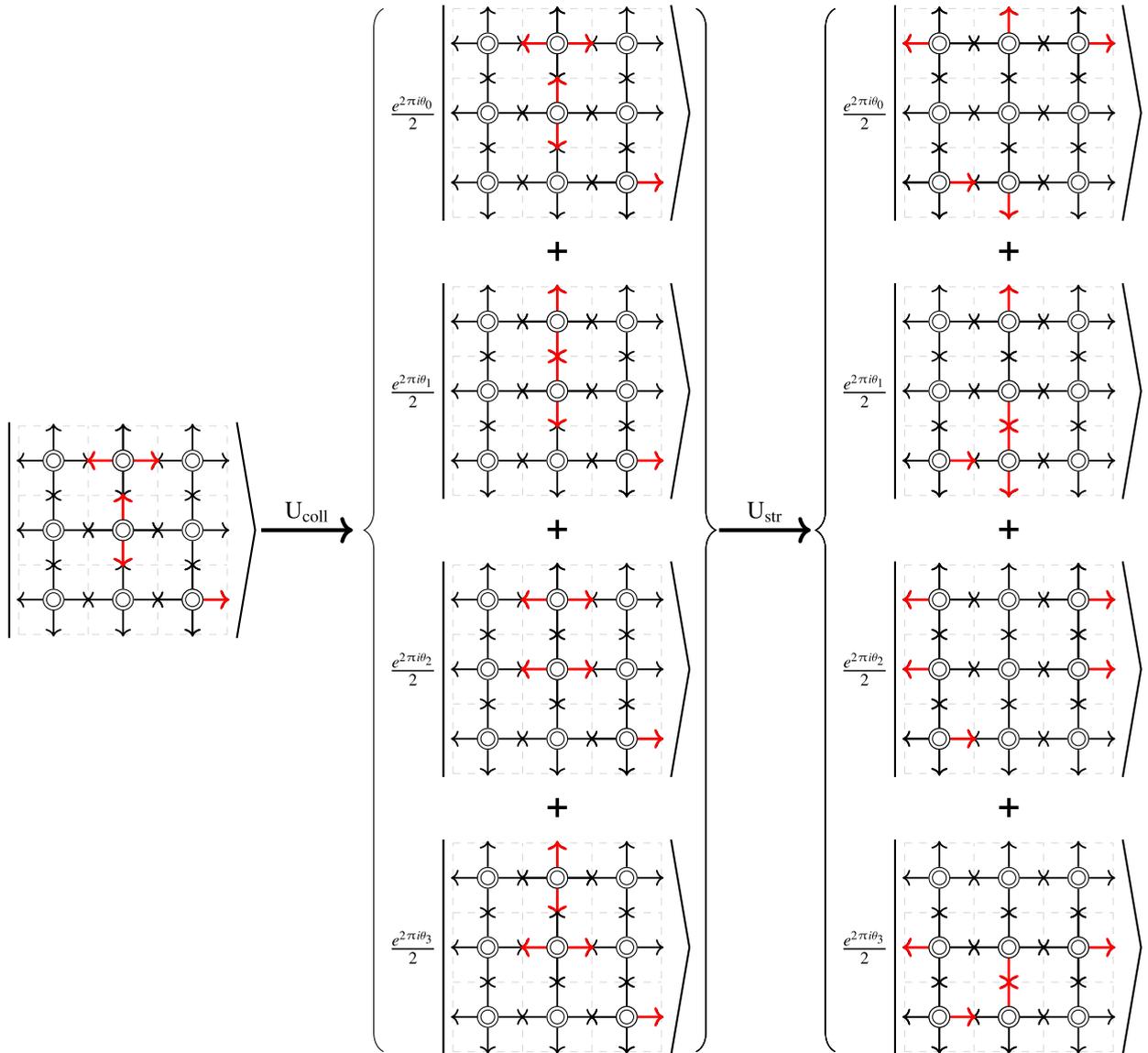
Collision operators in LGA redistribute particles located at the same lattice site inkeeping with mass and momentum conservation [18]. The exact rules that govern collision are specific to the velocity discretization and, classically, the redistribution of particles can either be deterministic or stochastic. Importantly, the boolean nature of particle discretizations in LGA makes it such that collision operators are generally linear, and therefore good candidates for a quantum circuit implementations.

In this section, we consider two ways of designing and implementing LGA collision operators in the Space-Time encoding – *one-to-one* and *superposed*. Fig. 14 shows the difference between the two models in a D_2Q_4 setting. One-to-one collision models act as depicted in Fig. 14, where the state of each lattice site is permuted onto a pre-determined outcome such that macroscopic quantities (i.e. mass, momentum) are conserved. This was the approach proposed by the earliest LGA models, and was later superseded by ensemble averaging over stochastic models, with the aim of reducing noise.

Fig. 14 depicts the superposed collision operator in the D_2Q_4 discretization. The difference between the two models is that the superposed approach introduces a stochastic component to the permutation of states. In practice, classical implementations of LGA would sample random numbers that would then map each state onto a discrete outcome from a set of states with equivalent macroscopic quantities. In designing the quantum circuit for this operator, we need not select one sample out of the set of possible outcomes, but can instead represent all feasible states simultaneously. This in turn allows for the simultaneous simulation of exponentially many independent lattice configurations, which is equivalent to a lattice-level ensemble in the taxonomy proposed by Wang et al. [79]. It is this representational power of the extended computational basis state encoding that circumvents the need for ensemble averaging in classical LGA. Though both one-to-one and superposed QLGA collision operators have already been proposed in the literature [31,32,61,63], they are all designed for specific discretizations or follow a rigid definition of state equivalency. In the remainder of this section, we provide both concrete quantum circuits and a generic formulation that apply superposed collision for arbitrary D_dQ_q stencils.



(a) One-to-one collision and streaming in the D_2Q_4 discretization.



(b) Superposed collision through uniform superposition followed by streaming in the D_2Q_4 discretization.

Fig. 14. Comparison of one-to-one and superposed collision models in the D_2Q_4 discretization.

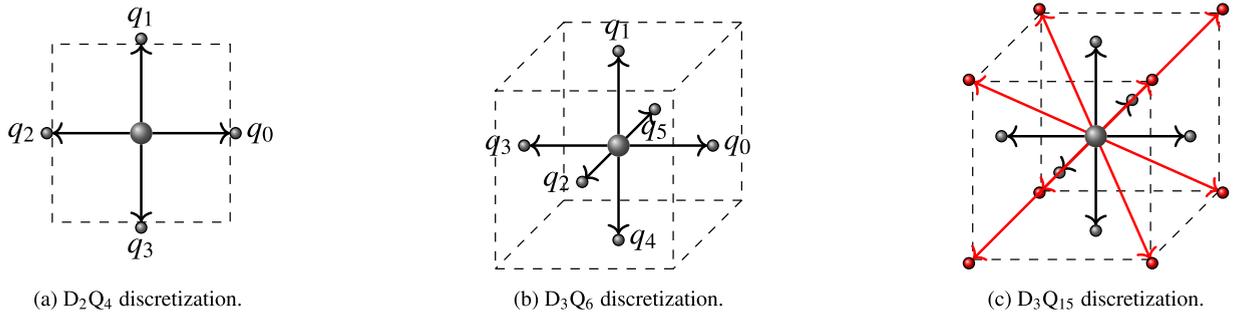


Fig. 15. Example velocity discretizations.

Throughout this section, we use the D_2Q_4 , D_3Q_6 , and D_3Q_{15} velocity discretizations as depicted in Fig. 15 to illustrate the challenges different configurations give rise to, and how our methods address them. We define collision operators that perform computations in terms of *equivalence classes*. To build up the set of all equivalence classes attached to velocity discretization, we first define the set all possible momentum values of a D_dQ_q discretization as

$$\mathbb{M}_{D_dQ_q} = \left\{ \sum_{j=0}^{q-1} k_j e_j \mid k \in \{0, 1\}^q \right\}, \tag{9}$$

with the e_j vectors defining the momentum contribution of each velocity channel according to the discretization, and k_j the j th bit in the binary representation of k . The exact values of these vectors are given by Krüger et al. [19] for several common discretizations. Let $E_{(D_dQ_q, m, \mu)} \subseteq \{0, 1\}^q$ be a set of local velocity configurations that have mass m and momentum $\mu \in \mathbb{M}_{D_dQ_q}$ under a given velocity discretization D_dQ_q , where each velocity is represented as in the computational basis state encoding. Furthermore, let $\|E_{(D_dQ_q, m, \mu)}\|$ be the number of velocity profiles in the class and let

$$\mathbb{E}_{D_dQ_q} = \left\{ E_{(D_dQ_q, m, \mu)} \mid m \in \mathbb{N}, \mu \in \mathbb{M}_{D_dQ_q} \right\} \tag{10}$$

be the set of all equivalence classes of the velocity discretization. We note that our definition of equivalence classes leads to larger sets of equivalent lattice configurations, as we do not constrain them based on specific rotation angles as in [32] and [64]. This increases the expressiveness of the model, as more novel states can emerge from the same configuration. When interpreting LGA as a random walk, this model change inherently increases the exploration space.

So far as designing collision operators is concerned, we restrict ourselves to transformations acting on configurations with $m \geq 2$ as in simpler equivalence classes the collision is trivial. For the D_2Q_4 discretization depicted in Fig. 15, there is a single set non-trivial equivalence class $E_{(D_2Q_4, 2, (0,0)^T)} = \{1010, 0101\}$, that contains two particles moving in opposing directions across the same streaming line [31]. The 8 relevant equivalence classes of D_3Q_6 (Fig. 15) are:

$$E_{(D_3Q_6, 2, (0,0,0)^T)} = \{100100, 010010, 001001\}, \tag{11a}$$

$$E_{(D_3Q_6, 4, (0,0,0)^T)} = \{110110, 101101, 011011\}, \tag{11b}$$

$$E_{(D_3Q_6, 3, (1,0,0)^T)} = \{110010, 101001\}, \tag{11c}$$

$$E_{(D_3Q_6, 3, (-1,0,0)^T)} = \{010110, 001101\}, \tag{11d}$$

$$E_{(D_3Q_6, 3, (0,1,0)^T)} = \{110100, 011001\}, \tag{11e}$$

$$E_{(D_3Q_6, 3, (0,-1,0)^T)} = \{100110, 001011\}, \tag{11f}$$

$$E_{(D_3Q_6, 3, (0,0,1)^T)} = \{101100, 011010\}, \tag{11g}$$

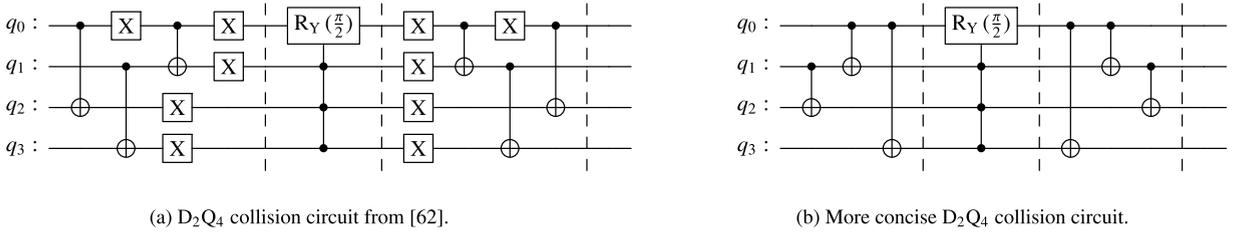
$$E_{(D_3Q_6, 3, (0,0,-1)^T)} = \{100101, 010011\}, \tag{11h}$$

while there are a total of 2832 equivalence classes for D_3Q_{15} .² In the following sections, we describe the process behind implementing quantum circuits for arbitrary equivalence classes, as well as how these circuits can be concatenated to obtain generic complete collision operators for a given discretization.

4.1. Rationale and small collision circuits

In previous work, Schalkers and Möller [31] implemented a collision operator for the Space-Time encoding of the D_2Q_4 discretization as depicted in Fig. 16. Conceptually, the circuit can be broken down into 3 logical steps that transform the quantum state encoding

² The code we used to generate all equivalence classes in according to our definition is available with the replication package [85].

Fig. 16. D_2Q_4 collision circuits.

information at a lattice site as

$$|\psi\rangle_C \leftarrow U_P^\dagger U_R U_P |\psi\rangle. \quad (12)$$

The three steps proceed as follows. The local lattice configuration first undergoes *permutation* by a unitary matrix U_P that, for an equivalence class E , maps relevant states to a superposition of basis states $\sum_{k=0}^{\|E\|-1} |k\rangle |1\rangle^{\otimes(q-\lceil \log_2 \|E\|)}$. For D_2Q_4 , these states would a linear combination of $|0111\rangle$ and $|1111\rangle$. One possible and simple implementation of this circuit is by the first partition of the circuit shown in Fig. 16. The second step then *redistributes* the amplitude of the states that belong to the equivalence class in a uniform-magnitude superposition on the first qubit, controlled on the 3 other qubits. The third step is a reversal of the first step, which permutes the basis states back to their original configuration. This circuit effectively models an evenly superposed LGA collision, as it evenly distributes the probability mass between the two states of the equivalence class. Each step in this *permute-redistribute-unpermute* (PRP) sequence requires careful considerations and is feasible only under certain constraints, which we delve into in the following paragraphs.

Permutations. Permutations are only necessary to map particular basis states onto pre-determined outcomes. For an arbitrary equivalence class $E_{(D_d Q_q, m, \mu)}$ with $E_C = \|E_{(D_d Q_q, m, \mu)}\|$ elements, the permutation should map the members of the equivalence class onto the $\sum_{k=0}^{E_C-1} |k\rangle |1\rangle^{\otimes(q-\lceil \log_2 E_C \rceil)}$ states, where (without loss of generality) $0 \leq k < E_C$ and $|k\rangle$ is encoded in the first $\lceil \log_2 E_C \rceil$ qubits of the E_C -wide register. The states that velocity configurations not belonging to $E_{(D_d Q_q, m, \mu)}$ are mapped to is not relevant under these conditions, as we explore in the next paragraph. In practice, this means there are multiple feasible permutations that the quantum circuit can perform to model the same physical behavior. Fig. 16 gives an example of such an alternative for the D_2Q_4 discretization, where the equivalence class members are mapped to the appropriate domain using fewer gates.

Particle redistribution. Much like the permutation step, the redistribution step can take several forms, all of which act on the first $\lceil \log_2 E_C \rceil$ qubits of the register. The $R_Y(\pi/2)$ gate that induces the superposition on the states of interest can thus be substituted for a different (controlled) gate depending on the physical behavior that the model attempts to capture. Permutations applied to the first $\lceil \log_2 E_C \rceil$ qubits of the register (a simple C^3X gate for D_2Q_4) model one-to-one collisions, whereas operations that introduce superpositions simultaneously model all possible outcomes of collisions. Crucially, whichever model the circuit attempts to implement, the redistribution step must not introduce any basis state $|x\rangle$ that falls in the range $E_C < x < 2^{\lceil \log_2 E_C \rceil}$, as (under the assumptions established in the previous paragraph) that would encode a velocity configuration that does not belong to the equivalence class. Uncomputing the permutation of such a state would result in behavior that is non-physical. This is not an issue under the D_2Q_4 discretization, but becomes challenging with stencils such as D_3Q_6 , as we explore next.

Example – D_3Q_6 collision. Let us consider how one could apply Eq. (12) to $E_{(D_3 Q_6, 2, (0,0,0)^T)}$ described in Eq. (11a). Fig. 17 shows one such circuit that follows the PRP principle acting on the 6 qubits required to encode the D_3Q_6 velocity set. The first permutation step is comprised of 8 CX gates and has a depth of 4, while 2 $R_Y(\pi/2)$ gates create a superposition of the first $\lceil \log_2 (\|E_{(D_3 Q_6, 2, (0,0,0)^T)}\|) \rceil = 2$ qubits. This means that the states affected by the rotation matrices are $|001111\rangle$, $|011111\rangle$, $|101111\rangle$, and $|111111\rangle$. While the permutation correctly maps the states of $E_{(D_3 Q_6, 2, (0,0,0)^T)}$ onto the first three of these states, it also leaves the $|111111\rangle$ state unchanged. This in turn means that applying the circuit in Fig. 17 to, for instance, the state $|100100\rangle$ results in

$$\begin{aligned} U_P^\dagger U_R U_P |100100\rangle &= U_P^\dagger U_R |101111\rangle \\ &= U_P^\dagger \frac{1}{2} (|001111\rangle - |011111\rangle + |101111\rangle - |111111\rangle) \\ &= \frac{1}{2} (|001001\rangle - |010010\rangle + |100100\rangle - |111111\rangle) \end{aligned} \quad (13)$$

which in addition to the three constituents of $E_{(D_3 Q_6, 2, (0,0,0)^T)}$ also encodes the state where all streaming channels of the lattice site are occupied. Clearly, this violates the mass conservation requirement of the collision operator and compromises the validity of the simulation.

This incompatibility is not tied to any particular equivalence class but generally occurs when the cardinality of an equivalence class is not exactly a power of 2. While redistribution operators such as the $R_Y(\cdot)$ gate are concise and exact for stencils such as D_2Q_4 , their utility is limited on more complex discretizations. In the following section, we describe a generic unitary redistribution operator that can act on arbitrary stencil discretizations that addresses this issue.

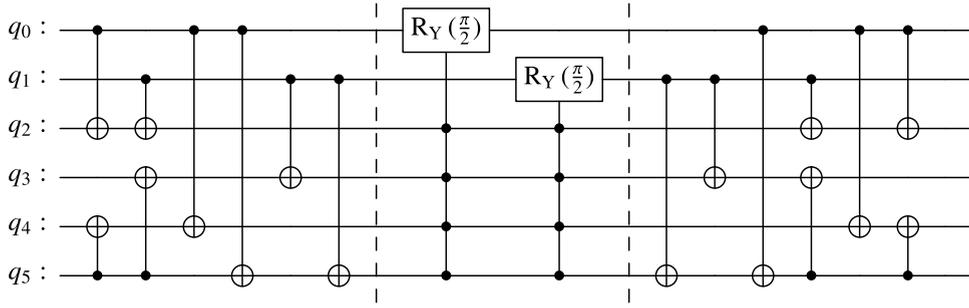


Fig. 17. Inexact collision circuit for $E_{(D_3 Q_6, 2, (0,0,0)^T)}$.

4.2. Arbitrary and complete collision circuits

To build up the collision operators for arbitrary discretizations, we start from a discrete Fourier transform (DFT) matrix. For the purposes of this section we define this matrix as

$$\begin{aligned}
 DFT(N) &= \frac{1}{\sqrt{N}} \left(e^{\frac{-2jk\pi i}{N}} \right)_{j,k=0,\dots,N-1} \\
 &= \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & e^{\frac{-2\pi i}{N}} & e^{\frac{-4\pi i}{N}} & e^{\frac{-6\pi i}{N}} & \dots & e^{\frac{-2(N-1)\pi i}{N}} \\ 1 & e^{\frac{-4\pi i}{N}} & e^{\frac{-8\pi i}{N}} & e^{\frac{-12\pi i}{N}} & \dots & e^{\frac{-2 \cdot 2(N-1)\pi i}{N}} \\ 1 & e^{\frac{-6\pi i}{N}} & e^{\frac{-12\pi i}{N}} & e^{\frac{-18\pi i}{N}} & \dots & e^{\frac{-2 \cdot 3(N-1)\pi i}{N}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\frac{-2 \cdot 2(N-1)\pi i}{N}} & e^{\frac{-2 \cdot 2(N-1)\pi i}{N}} & e^{\frac{-2 \cdot 3(N-1)\pi i}{N}} & \dots & e^{\frac{-2 \cdot (N-1)(N-1)\pi i}{N}} \end{pmatrix}, \tag{14}
 \end{aligned}$$

with N the size of the of the matrix. We use this to define the larger collision operator $\text{Coll}(k, N)$ as

$$\text{Coll}(k, N) = \begin{pmatrix} DFT(k) & \mathbb{O}_{k \times (N-k)} \\ \mathbb{O}_{(N-k) \times k} & \mathbb{I}_{N-k} \end{pmatrix}, \tag{15}$$

with N the size of the matrix, and $k \leq N$ the size of the DFT subblock. To realize collision in the Space-Time encoding for a discretization $D_d Q_q$, and a velocity class E , we set $k = \|E\|$ and $N = 2^{\lceil \log_2 \|E\| \rceil}$. When applied to a q -qubit basis state $|\psi\rangle$, performs the following transformation

$$\text{Coll}(\|E\|, 2^{\lceil \log_2 \|E\| \rceil})|\psi\rangle = \begin{cases} \frac{1}{\sqrt{\|E\|}} \sum_{k=0}^{\|E\|-1} e^{\frac{-2k\pi i}{\|E\|}} |k\rangle & |\psi\rangle \in \{|0\rangle, \dots, \|\|E\| - 1\rangle\} \\ |\psi\rangle & \text{otherwise,} \end{cases} \tag{16}$$

which can be understood as creating a superposition of the first k basis states when applied to one of them while leaving the remainder $N - k$ basis states unaffected. The amplitudes of the basis states in superposition are all of the form $e^{\frac{-2 \cdot k\pi i}{N}} / \sqrt{N}$, which when squared leads a probability of $1/N$ and therefore a uniform-magnitude superposition of all N basis states of interest.³

It is important to note that q is fixed according to the choice of lattice discretization, and size of the DFT block depends on the equivalence class of the discretization. Because of these properties, we can determine the complexity of the circuit that implements $\text{Coll}(k, N)$ exactly for commonplace discretizations, as well as reason about the complexity of the operator in general. Before obtaining such a general expression of complexity, we first have to analyze how the PRP procedure can be expanded to cover all equivalence classes of a particular discretization.

³ The unitarity of the $\text{Coll}(k, N)$ matrix follows directly from the unitarity of the DFT matrix as

$$\begin{aligned}
 \text{Coll}(k, N)\text{Coll}(k, N)^\dagger &= \begin{pmatrix} DFT(k) & \mathbb{O}_{k \times (N-k)} \\ \mathbb{O}_{(N-k) \times k} & \mathbb{I}_{N-k} \end{pmatrix} \begin{pmatrix} DFT(k) & \mathbb{O}_{k \times (N-k)} \\ \mathbb{O}_{(N-k) \times k} & \mathbb{I}_{N-k} \end{pmatrix}^\dagger \\
 &= \begin{pmatrix} DFT(k) & \mathbb{O}_{k \times (N-k)} \\ \mathbb{O}_{(N-k) \times k} & \mathbb{I}_{N-k} \end{pmatrix} \begin{pmatrix} DFT(k)^\dagger & \mathbb{O}_{k \times (N-k)} \\ \mathbb{O}_{(N-k) \times k} & \mathbb{I}_{N-k} \end{pmatrix} \\
 &= \begin{pmatrix} DFT(k)DFT(k)^\dagger + \mathbb{O}_{k \times (N-k)}\mathbb{O}_{(N-k) \times k} & DFT(k)\mathbb{O}_{k \times (N-k)} + \mathbb{O}_{k \times (N-k)}\mathbb{I}_{N-k} \\ \mathbb{O}_{(N-k) \times k}DFT(k) + \mathbb{I}_{N-k}\mathbb{O}_{k \times (N-k)} & \mathbb{O}_{(N-k) \times k}\mathbb{O}_{k \times (N-k)} + \mathbb{I}_{N-k}\mathbb{I}_{N-k} \end{pmatrix} \\
 &= \begin{pmatrix} \mathbb{I}_k & \mathbb{O}_{k \times (N-k)} \\ \mathbb{O}_{(N-k) \times k} & \mathbb{I}_{N-k} \end{pmatrix} = \mathbb{I}_N.
 \end{aligned}$$

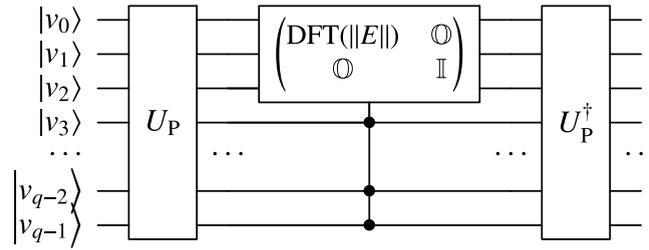


Fig. 18. Schematic of the generic collision operator for an equivalence class E with $4 < \|E\| < 8$.

Let us consider a system of n_g grid qubits, and analyze the circuit that performs collision for one time step, while assuming that N_t time steps are still to be simulated. Under an arbitrary $D_d Q_q$ discretization with equivalence classes in $\mathbb{E}_{D_d Q_q}$, this requires the customary $\mathcal{O}(q N_t^d)$ neighboring qubits. For any individual equivalence class $E \in \mathbb{E}_{D_d Q_q}$, the PRP-based collision operator is given by

$$\mathbb{I}^{\otimes n_g} \otimes \left((U_P(E)^\dagger)^{\otimes \mathcal{O}(N_t^d)} \left(C^{(q - \lceil \log_2 \|E\| \rceil)} \text{Coll}(\|E\|, 2^{\lceil \log_2 \|E\| \rceil}) \right)^{\otimes \mathcal{O}(N_t^d)} U_P(E)^{\otimes \mathcal{O}(N_t^d)} \right), \quad (17)$$

with $C^{(q - \lceil \log_2 \|E\| \rceil)}$ the q -qubit matrix obtained by applying the matrix given in Eq. (15) controlled on the trailing $q - \lceil \log_2 \|E\| \rceil$ qubits. Fig. 18 shows a schematic representation of the quantum circuit acting only on the velocity register. Intuitively, this applies the equivalence class-specific collision operator in parallel $\mathcal{O}(N_t^d)$ times for each q -qubit subregister, while leaving the grid register unaffected. The general collision operator can then be expressed as

$$\mathbb{I}^{\otimes n_g} \otimes \prod_{E \in \mathbb{E}_{D_d Q_q}} \left((U_P(E)^\dagger)^{\otimes \mathcal{O}(N_t^d)} \left(C^{(q - \lceil \log_2 \|E\| \rceil)} \text{Coll}(\|E\|, 2^{\lceil \log_2 \|E\| \rceil}) \right)^{\otimes \mathcal{O}(N_t^d)} U_P(E)^{\otimes \mathcal{O}(N_t^d)} \right), \quad (18)$$

that is, each equivalence class can be addressed sequentially on each q -qubit velocity register. The crucial property that enables this sequential application is that no two equivalence class-specific collision operators act on the same basis states, as there are no common velocity configuration between equivalence classes.

Complexity analysis. Eq. (18) provides an expression of collision operators that allows us to easily derive their gate complexity. Each equivalence class requires two permutation circuits and one redistribution circuit. Recent results by Herbert et al. [80] describe an efficient way to construct circuits that implement transpositions of n -qubit basis states using $\mathcal{O}(n)$ gates. Since at this level the collision circuit acts in parallel on q -qubit subregisters, and the permutation step only requires that the $\|E\|$ states of the equivalence class are mapped to specific states, the permutation step can be implemented as a series of at most $\|E\|$ transpositions, and is therefore $\mathcal{O}(\|E\|q)$ per subregister, and $\mathcal{O}(N_t^d \|E\|q)$ across the entire circuit. The final step of undoing the permutation can be implemented by mirroring the permutation circuit and thus the two steps share the same complexity. The most complex redistribution circuit requires the decomposition of the $C^{(q - \lceil \log_2 \|E\| \rceil)} \text{Coll}(\|E\|, 2^{\lceil \log_2 \|E\| \rceil})$ matrix.

In general, the decomposition of any arbitrary unitary matrix onto a simple universal gate set requires a number of single- and two-qubit gates that is in general exponential in the number of qubits the matrix is applied on [73]. While this does generally compromise the efficiency of the algorithm and requires heavy classical processing, there are several properties of this matrix that may allow for practical decompositions. First, the size of the matrix is bound by q , the number of discrete velocities in the discretization. Second, the bottom $2^{q - \lceil \log_2 \|E\| \rceil}$ entries of the matrix are only used to restrict the application of the $\text{Coll}(\|E\|, 2^{\lceil \log_2 \|E\| \rceil})$ matrix through controls, and the matrix is therefore sparse and structured. Third, the denser $\text{Coll}(\|E\|, 2^{\lceil \log_2 \|E\| \rceil})$ submatrix is itself bounded by the size of the equivalence class with the highest cardinality of the specific discretization, and the number of 0 entries in the matrix is equal to $q^2 - \|E\|^2 - q + \|E\|$. For instance, the most populated equivalence classes of the $D_3 Q_{15}$ discretization are $E_{D_3 Q_{15}, j, (0,0,0)^T}$, for $j \in \{6, 7, 8, 9\}$, which contain 73 velocity profiles each. The matrix encoding the corresponding redistribution operator is therefore $\text{Coll}(73, 128)$, which acts on 7 qubits and is $\approx 67.48\%$ sparse.

Further straight-forward techniques, such as caching transpilation and re-using one transpiled circuit across all time-steps and across multiple simulations can further significantly reduce the classical resources necessary to produce the quantum circuits. Convenient situations in which equivalence classes can be treated with sequential applications of (multi-controlled) $R_V(\cdot)$ gates can also be implemented more efficiently and require less complex decompositions. The implementation of one-to-one collision rules can be performed by means of transpositions, which only require $\mathcal{O}(q^2)$ CX gates for the redistribution step. In general, however, the decomposition of the large collision matrix clearly dominates the complexity of the algorithm and therefore the number of native quantum gates required to implement the collision operator scales with $\mathcal{O}(\|\mathbb{E}_{D_d Q_q}\| N_t^d (\|E_M\| q + 2^q))$, where $E_M = \max_{E \in \mathbb{E}_{D_d Q_q}} \|E\|$. Finally, we note that the DFT component of our redistribution operator sometimes appears in the literature as a *reduced order QFT*. For more details, we refer the interested reader to the works of Kitaev [68] and Mosca and Zalka [69], as well as the overview of Kempe [70].

5. Measurement

In this section, we detail techniques for extracting physical information encoded in the quantum state by means of observables and quantum circuits. We first introduce observables that can be used to compute the mass, density, and pressure over arbitrary regions of space, before describing quantum circuits and observables that compute the forces acting on an object by means of the momentum exchange method. Throughout the section and in each example, we express all physical quantities in terms of lattice units.

5.1. Mass, density, and pressure measurement

In lattice gas hydrodynamics, pressure is typically computed as

$$p = c_s^2 \rho, \quad (19)$$

with c_s the *speed of sound* in lattice units and ρ is the local particle density [81]. We formulate an observable P such that the expectation value $\langle P \rangle$ is equal to the (scaled) mass at a fixed lattice site, before generalizing its expression to arbitrary regions of space. Density is directly determined by the local mass and the velocity discretization, and the speed of sound c_s is given by the lattice discretization and therefore independent of the state of the system. We therefore express P in such a way that it computes mass, which in turn enables the calculation of density and pressure by trivial classical post-processing. Since LGA particles are boolean by nature, the particle density at a given lattice site x for a $D_d Q_q$ discretization with equally weighted particle channels is

$$\rho_x = \frac{1}{q} \sum_{j=0}^{q-1} f_j(x), \quad (20)$$

with f_j the particle occupancy of each of the q channels. In the computational basis state encoding of the velocity configuration, this information is encoded in q qubits with $|0\rangle$ encoding an empty channel and $|1\rangle$ an occupied one. For a particular qubit $0 \leq j \leq q-1$ encoding such a channel, the observable that extracts the probability of measuring the state $|1\rangle$ is

$$n_j = \mathbb{1}^{\otimes j} \otimes \frac{1}{2}(\mathbb{1} - \mathbb{Z}) \otimes \mathbb{1}^{\otimes(q-1-j)}, \quad (21)$$

where \mathbb{Z} is Pauli Z -matrix. Intuitively, this is a projection of the $|1\rangle$ state of the particular velocity channel, and the value of $\langle \psi | n_j | \psi \rangle$ is the probability of measuring the j th qubit in state $|1\rangle$. Extending this to compute the mass encoded in a q -qubit velocity register, we get the observable

$$N = \sum_{j=0}^{q-1} n_j, \quad (22)$$

which encodes the probability of measuring any velocity channel in the state $|1\rangle$. Since mass is tied to the amplitude of the $|1\rangle$ state, of the velocity qubits, the matrix N can be re-written as

$$N = \text{diag} \left(\left(\sum_{j=0}^{q-1} k_j \right)_{k=0, \dots, q-1} \right), \quad (23)$$

where k_j is the j th bit in the binary representation of the q -bit number k . Intuitively, this matrix weighs each basis state by the number of 1s in its binary encoding. This is also called the *Hamming weight* of the bitstring, which in the computational basis state encoding corresponds exactly to the number of particles present at the lattice site. Assuming the quantum state is encoded as $|\nu\rangle|g\rangle$ i.e. the velocity qubits precede the positional qubits, and that the positional qubits are in a superposition that includes the target lattice site x , we can isolate this position with the operator

$$N_x = \left(\sum_{j=0}^{q-1} n_j \right) \otimes |x\rangle\langle x|, \quad (24)$$

which projects the particle mass encoded in the velocity register conditioned on the grid qubits being in state $|x\rangle$. Finally, the density, and therefore pressure, over a region Γ can be computed by means of the operator

$$P_\Gamma = \sum_{x \in \Gamma} N_x = \sum_{x \in \Gamma} \left(\sum_{j=0}^{q-1} \left(\mathbb{1}^{\otimes j} \otimes \frac{1}{2}(\mathbb{1} - \mathbb{Z}) \otimes \mathbb{1}^{\otimes(q-1-j)} \right) \right) |x\rangle\langle x|. \quad (25)$$

Assuming a one-to-one collision model and computing the expectation of P with respect to an arbitrary QLGA quantum state $|\psi\rangle$ yields

$$\langle \psi | P_\Gamma | \psi \rangle \propto \sum_{x \in \Gamma} \sum_{j=0}^{q-1} f_j(x), \quad (26)$$

which in turn leads to the mass over Γ being $m_\Gamma = \frac{2^{ng}}{\|\Gamma\|} \langle \psi | P_\Gamma | \psi \rangle$ and the density $\rho_\Gamma = \frac{2^{ng}}{q\|\Gamma\|} \langle \psi | P_\Gamma | \psi \rangle$.

Example – D_1Q_2 mass measurement. We begin with the simplest working example that demonstrates how the operator we described previously correctly computes the mass of a basic D_1Q_2 system. For this instance, consider a lattice with just 2 gridpoints, and therefore one positional qubit. Since there are no non-trivial equivalence classes in the D_1Q_2 discretization, there is no superposition over the velocity qubits. Let the state of the system be

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle|10\rangle + |1\rangle|11\rangle). \quad (27)$$

That is, the leftmost gridpoint $|0\rangle$ has a particle on one of its velocity channels ($|10\rangle$), while the rightmost gridpoint $|1\rangle$ has 2 particles ($|11\rangle$). We verify the claim that the application of P_Γ derived previously to $|\psi_1\rangle$ correctly computes a scalar that can be post-processed into the values of mass, density, and pressure, at the gridpoint corresponding to $|0\rangle$. First, constructing the N matrix as described in Eq. (23), we obtain

$$N_{D_1Q_2} = \text{diag}((0, 1, 1, 2)), \quad (28)$$

which weighs the four 2-qubit basis states by their Hamming weight in the binary encoding. To additionally project the $|0\rangle$ grid point that we are interested in, we adapt the form given in Eq. (24) to obtain

$$\begin{aligned} N_{(D_1Q_2, \{0\})} &= (|0\rangle\langle 0|) \otimes \text{diag}((0, 1, 1, 2)) \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \\ &= \begin{pmatrix} \text{diag}((0, 1, 1, 2)) & \mathbb{O}_{4 \times 4} \\ \mathbb{O}_{4 \times 4} & \mathbb{O}_{4 \times 4} \end{pmatrix}. \end{aligned} \quad (29)$$

Since we are only interested in the properties of a single gridpoint, $N_{(D_1Q_2, \{0\})} = P_\Gamma$ and the expectation value $\langle \psi_1 | P_\Gamma | \psi_1 \rangle = \frac{1}{2}$ gives the probability mass corresponding to the states of interest, weighted by their mass. Multiplying this value by 2^{n_g} , we obtain 1, the total mass over the area, which in turn leads to density and pressure. The physical interpretation of this procedure is straightforward for the deterministic D_1Q_2 model, but becomes more involved once superposition-driven collision is applied, as we show next.

Example – D_2Q_4 mass measurement. To demonstrate the applicability of the mass measurement technique to a more general case, we use a D_2Q_4 instance that includes a collision step. For brevity, we omit the positional qubits and non-local velocity qubits and instead focus on the 4 qubits belonging to a physical origin. Assume the state

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|1000\rangle + |1010\rangle) \quad (30)$$

is obtained as a result of streaming, *i.e.*, from a left neighbor where no collision took place in the previous time step, and from a down neighbor where collision did occur. The $|1010\rangle$ basis state belongs to $E_{(D_2Q_4, 2, (0,0)^T)} = \{1010, 0101\}$, and as a result, its amplitude is shared with the $|0101\rangle$ state, resulting in a post-collision configuration of

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}|1000\rangle + \frac{1}{2}|1010\rangle + \frac{1}{2}|0101\rangle. \quad (31)$$

To verify the correctness of the mass measurement observable with respect to the collision operator, we evaluate its expectation with respect to $|\psi_1\rangle$ and $|\psi_2\rangle$ separately. The corresponding observable is

$$N_{D_2Q_4} = \text{diag}((0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4)). \quad (32)$$

Applying this operator to the pre-collision state yields

$$\langle \psi_1 | N_{D_2Q_4} | \psi_1 \rangle = \frac{1}{2}(1 + 2) = \frac{3}{2}, \quad (33)$$

as the Hamming weight of both basis states in Eq. (30) is equal, and the sum is therefore equal to the average mass of all states in the superposition. Applying the same observable to the post-collision of Eq. (31) configuration results in

$$\langle \psi_2 | N_{D_2Q_4} | \psi_2 \rangle = \frac{1}{2}1 + \frac{1}{4}(2 + 2) = \frac{3}{2}. \quad (34)$$

The three basis states of $|\psi_2\rangle$ retain the same cumulative contribution to the local mass as the pre-collision configuration. In general, assuming collision affects a single basis state $|k\rangle$ with amplitude α that belongs to equivalence class E , the post-collision configuration divides the amplitude of $|k\rangle$ uniformly along $\|E\|$ basis states that, by virtue of belonging to the same equivalence class, all have the same mass. Assuming the mass of $|k\rangle$ is $m_k = \sum_{j=0}^{q-1} |k_j\rangle$, since $|\alpha|^2 m_k = |\alpha|^2 \sum_{j=1}^{\|E\|} \frac{1}{\|E\|} m_k$, collision preserves the expectation of the observable N .

5.2. Momentum exchange method

The Momentum Exchange Method (MEM) was developed for Lattice Boltzmann Methods by Ladd [82,83] as a way to compute the forces acting on solid objects suspended in a fluid. In the context of the continuous probability distributions of the LBM, the MEM prescribes that the force acting on the solid domain is equal to

$$\mathbf{F}_\Gamma = \sum_{j=0}^{q-1} (\mathbf{e}_j f_j(\mathbf{x}_\Gamma) - \mathbf{e}_j f_j(\mathbf{x}_\Gamma)), \quad (35)$$

where Γ is the domain that comprises the solid-fluid interface (under the assumption that the solid is free of fluid particles), and \mathbf{e}_j are vectors that encode the direction of the each streaming channel according to the $D_d Q_q$ discretization. Similarly, \mathbf{e}_j is the directional vector of the channel that a particle traveling channel j occupies after the application of boundary conditions. For simplicity, we assume the simulation models bounce-back boundary conditions, which simplifies the force calculation to

$$\mathbf{F}_\Gamma = \sum_{j=0}^{q-1} 2\mathbf{e}_j f_j(\mathbf{x}_\Gamma), \quad (36)$$

as the opposing velocity channels that particles occupy before and after boundary treatment are \mathbf{e}_j and $-\mathbf{e}_j$, respectively. For the exact values of the \mathbf{e}_j vectors corresponding to common $D_d Q_q$ discretizations, we refer the reader to [19].

The first Quantum Momentum Exchange Method (QMEM) was introduced by Schalkers and Möller [67] for an algorithm based on the amplitude encoding. In their work, $2d$ observables each calculate one component of the force vector, along each direction of each dimension. However, since in the amplitude encoding, the amplitude of particular basis state singularly determines the mass occupying a particular velocity channel, this method is not directly compatible with computational basis state encodings. In this section, we adapt the quantum circuit implementation of the QMEM observable described in [67] to the LGA model underlying the Space-Time encoding.

QMEM in the space-time encoding. The core differences between the LBM and LGA MEM implementations stem from the discretization and quantum register structure, respectively. First, particles in LGA are boolean and therefore the f_j terms in Eq. (36) are either 0 or 1. Second, as previously described in Section 5.1, the weight of each basis state in a superposition at a local gridpoint must be accounted for proportionately to the force it exerts on the boundary of the solid object. Furthermore, the locality of the Space-Time stencil prevents the straightforward implementation from the original work [67] from measuring the appropriate quantity of interest, as it relies on particles having traveled into the boundary layer of the object. Since information is local to grid positions in the Space-Time encoding, this scenario never occurs and therefore requires a different approach.

Formulating observables that account for these criteria is straightforward and can be done following the same steps as in Section 5.1. For each dimension k , we can define a diagonal matrix

$$N_k = \sum_{j=0}^{q-1} 2n_j \cdot \mathbf{e}_j(k), \quad (37)$$

with $\mathbf{e}_j(k)$ the k^{th} entry of the directional vector \mathbf{e}_j , and n_j as defined in Eq. (21). In practice, this eliminates entries of the N_k matrix that do not contribute to the force component corresponding to that dimension. The matrix could be further split up into two sparser observables that only account for components in one particular direction of the force as described in [67]. Unlike in the mass computation example, however, the final observable is no longer constructed with respect to the qubits corresponding to the physical origin, as that would entail initializing gridpoints inside the boundary of the obstacle, which we avoid as detailed in Section 3.1. Instead, we can construct the observable as

$$N_{(x,k)} = \mathbb{1}^{\otimes(q-o)} \left(\sum_{j=0}^{q-1} n_j \right) \mathbb{1}^{\otimes(q(n_r-o-1))} \otimes (|x'\rangle\langle x'|), \quad (38)$$

with x' the position of a gridpoint *within the fluid domain* that contains information regarding gridpoints that belong to the boundary of the solid domain in dimension k , n_r the number of q -qubit velocity registers the simulation contains, and o an appropriate offset. Intuitively, this observable is expressed with respect to the velocity qubits of gridpoints within the solid domain that are entangled with positional qubits in the fluid domain. Since the method follows the same rationale and reasoning as outline in Section 5.1, we omit additional examples and instead describe a quantum circuit implementation that amasses the amplitude of the states of interest onto an ancilla qubit.

Quantum circuit implementation and $D_2 Q_4$ example. Amassing the amplitude of the particles impinging on the object in one particular dimension can be achieved by utilizing the techniques described in Sections 2 and 3, as well as the $2d$ ancillae qubits used for volumetric operations and 1 additional ancilla qubit, a_0 . The quantum circuit first isolates the region of space in the fluid domain that interfaces with the solid domain by means of the volumetric operations described in previous sections. Following this step, at most $q^{2d} C^{2d+1} X$ gates flip the state of the a_0 qubit based on the state of the volumetric ancillae and the velocity qubit(s) that contribute to the particular component of the force vector we attempt to measure.

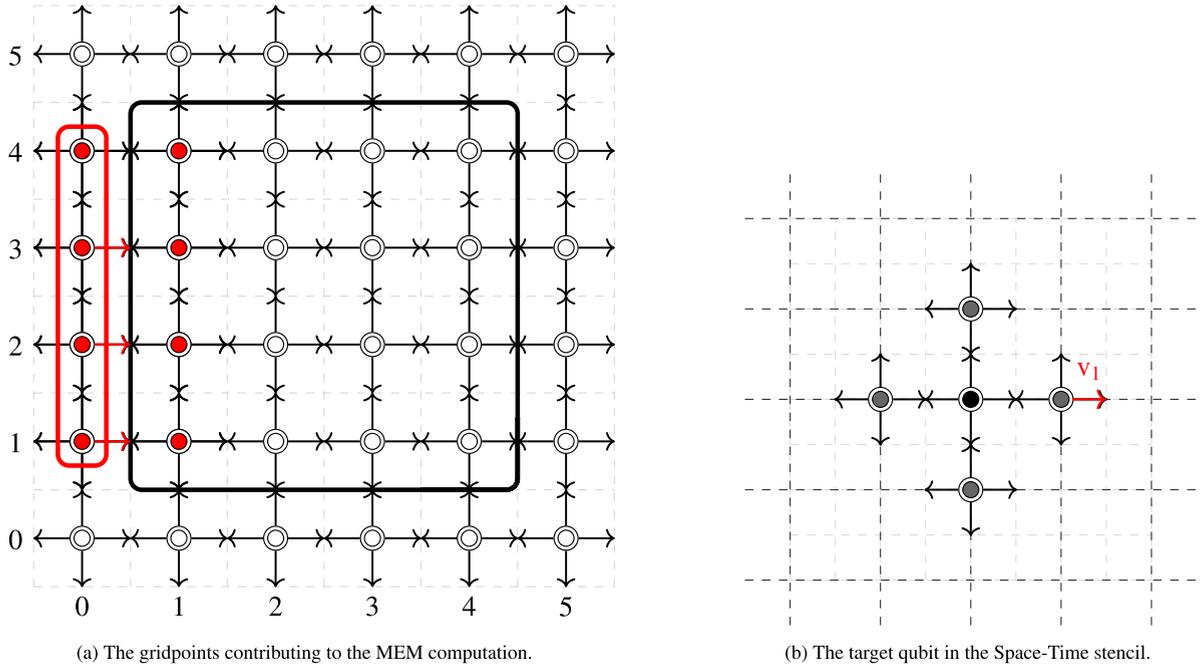


Fig. 19. Momentum exchange example on a 6×6 D_2Q_4 grid on the left wall of a square obstacle.

To demonstrate the applicability of this procedure, we consider a D_2Q_4 example depicted in Fig. 19. Fig. 19 shows the system, in which we are interested in measuring the horizontal component of the force vector acting in the positive streaming direction on the left wall of the square obstacle. The simplified state of the system before streaming is

$$|\psi_{\text{pre}}\rangle \propto |a_o\rangle|a_{l,y}\rangle|a_{u,y}\rangle|x\rangle|y\rangle|v_0^l v_1^l v_2^l v_3^l\rangle|v_0^n v_1^n v_2^n v_3^n\rangle, \tag{39}$$

which includes the measurement and volumetric ancillae for the dimension we are targeting, as well as the positional x and y qubits, the 4 local velocity qubits v_j^l , and the 4 velocity of the right neighbor of the physical origin v_j^n . For brevity, we omit all other qubits of the register, as they are not relevant for this computation. Assume the starting state of the system is

$$|\psi_1\rangle \propto |0\rangle|00\rangle(|0\rangle|1\rangle|1000\rangle + |0\rangle|2\rangle|1100\rangle + |0\rangle|3\rangle|1101\rangle + |0\rangle|3\rangle|0000\rangle)|0000\rangle + |x\rangle|y\rangle|v_{\text{out}}^{\text{loc}}\rangle|v_{\text{out}}^n\rangle). \tag{40}$$

The state of Eq. (40) assumes the four gridpoints enclosed in the red rectangle in Fig. 19 are in 4 different states of $|1000\rangle$, $|1100\rangle$, $|1101\rangle$, and $|0000\rangle$, respectively. The three velocity channels in state $|1\rangle$ that contribute to the MEM computation are highlighted in red in Fig. 19. As no particles have streamed into the object yet and no volumetric operations have been performed, the ancillae and neighboring velocity qubits are all $|0\rangle$. All other basis states are only represented generically as they are not relevant to the MEM. Following streaming, the updated quantum state becomes

$$|\psi_2\rangle \propto |0\rangle|00\rangle(|0\rangle|1\rangle|v_1^{\text{loc}}\rangle|1000\rangle + |0\rangle|2\rangle|v_2^{\text{loc}}\rangle|1000\rangle + |0\rangle|3\rangle|v_3^{\text{loc}}\rangle|1000\rangle + |0\rangle|4\rangle|v_3^{\text{loc}}\rangle|0000\rangle + |x\rangle|y\rangle|v_{\text{out}}^{\text{loc}}\rangle|v_{\text{out}}^n\rangle). \tag{41}$$

In Eq. (41), $|\psi_2\rangle$ encodes that the velocity occupancy of the first velocity channel has streamed from the (now no longer of interest) $|v_j^{\text{loc}}\rangle$ qubits of the physical origin to the qubits representing the right neighbor. As such, the qubit of interest is now in state $|1\rangle$ for three of the four solid gridpoints. At this point, the comparator operations can be applied to set the appropriate state onto the second and third ancillae as

$$|\psi_3\rangle \propto |0\rangle|11\rangle(|0\rangle|1\rangle|v_1^{\text{loc}}\rangle|1000\rangle + |0\rangle|2\rangle|v_2^{\text{loc}}\rangle|1000\rangle + |0\rangle|11\rangle(|0\rangle|3\rangle|v_3^{\text{loc}}\rangle|1000\rangle + |0\rangle|4\rangle|v_3^{\text{loc}}\rangle|0000\rangle + |0\rangle|k_1 k_2\rangle|x\rangle|y\rangle|v_{\text{out}}^{\text{loc}}\rangle|v_{\text{out}}^n\rangle), \tag{42}$$

where $|k_1 k_2\rangle$ symbolizes any 2-qubit quantum state that is not $|11\rangle$. Up to this point, the quantum circuit is identical to the volumetric boundary conditions implementation described in Section 3.2. It is for this reason that we express the quantum circuit

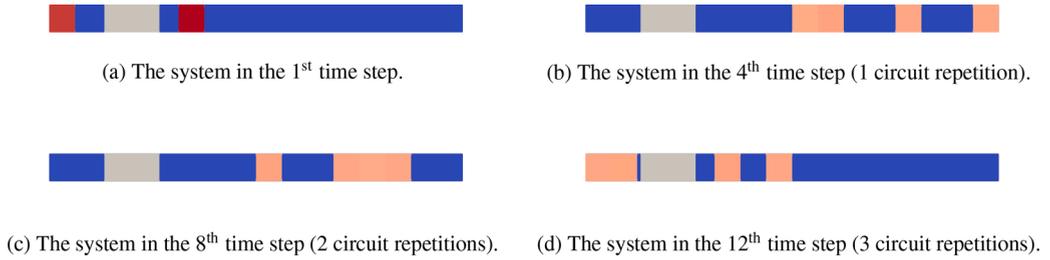


Fig. 20. Evolution of a D_1Q_2 4-time step quantum circuit system for 3 repetitions.

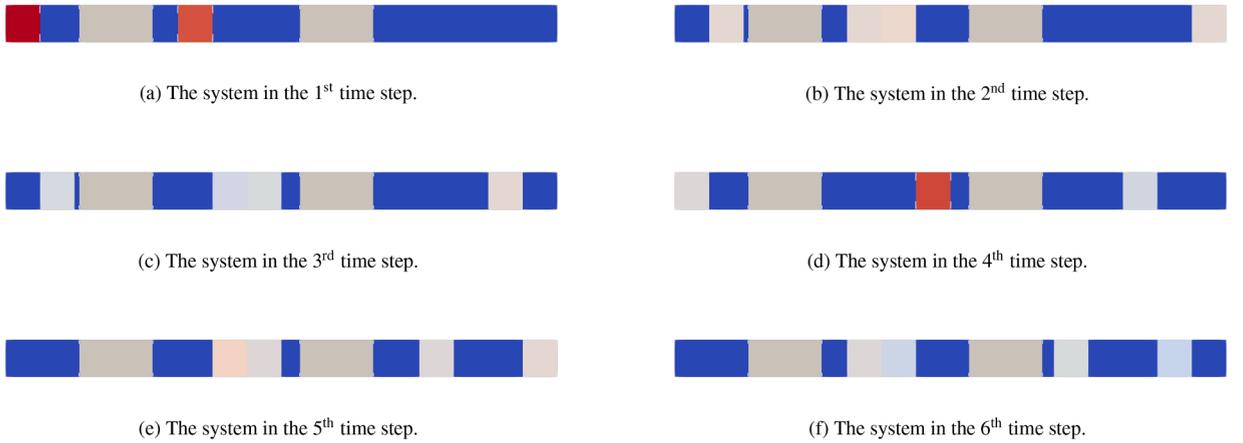


Fig. 21. Evolution of a D_1Q_2 1-time step quantum circuit system for 6 time steps.

with respect to the neighboring qubits of physical origin that belong to the boundary of the fluid domain. We could alternatively compute the same quantity by measuring the post-boundary condition velocity qubit in the opposite channel \bar{c} of the physical origin, or the channel c of the physical origin pre-streaming. Since no collision occurs between these steps, the same information travels across these channels, and the quantum circuits therefore compute the same value. We opt for this manner of implementing as it fits in exactly with the realization of the volumetric boundary conditions. Where the circuits differ is that to appropriately amass the amplitude onto the $|1\rangle$ state of the leading qubit, a C^3X gate is applied to the a_0 qubit, controlled on the two comparator qubits and the velocity qubit(s) that contribute to the force component. In this case, this is the first qubit of the neighboring gridpoint, visually depicted in Fig. 19. The state following this procedure is

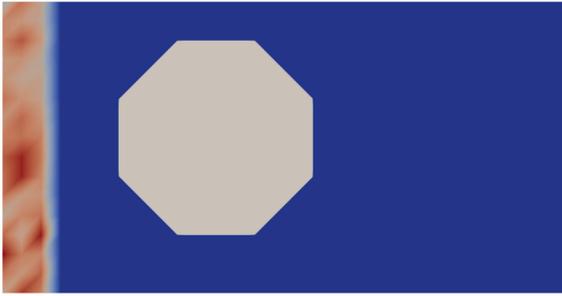
$$\begin{aligned}
 |\psi_4\rangle \propto & |1\rangle|11\rangle \left(|0\rangle|1\rangle|v_1^{\text{loc}}\rangle|1000\rangle + |0\rangle|2\rangle|v_2^{\text{loc}}\rangle|1000\rangle + |0\rangle|3\rangle|v_3^{\text{loc}}\rangle|1000\rangle \right) \\
 & + |0\rangle|11\rangle|0\rangle|4\rangle|v_3^{\text{loc}}\rangle|0000\rangle \\
 & + |0\rangle|k_1k_2\rangle|x\rangle|y\rangle|v_{\text{out}}^{\text{loc}}\rangle|v_{\text{out}}^{\text{n}}\rangle.
 \end{aligned} \tag{43}$$

In $|\psi_4\rangle$, the a_0 qubit can be measured to extract the target amplitude. Since the operations that isolate the positional qubits and the C^3X gate do not affect the amplitude of the target qubits, the same line of reasoning described in the previous section can be followed to demonstrate its compatibility with the superposed collision operators. To generalize this procedure, $\mathcal{O}(q)$ ancilla qubits are necessary, that can each be set to $|1\rangle$ for each velocity channel that travels in the direction of the force component we attempt to measure.

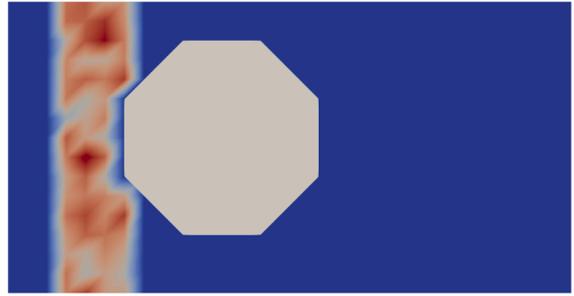
6. Results

In this section, we show the behavior of D_1Q_2 and D_2Q_4 systems simulated through the building blocks introduced in this work. We focus on showing the correct time-evolution of the system, rather than verifying the numerical validity of the model for particular target equations. This is an intentional limitation of the study, as we aim to address the latter through a broader empirical analysis in future works. Implementations for all described building blocks are available in the open-source QLBM software package [84]. Simulations were carried out using the Qiskit [86] AerSimulator, and visualizations are created through Paraview [87]. All scripts used to produce the results and quantum circuits described in this paper are available in the replication package [85].

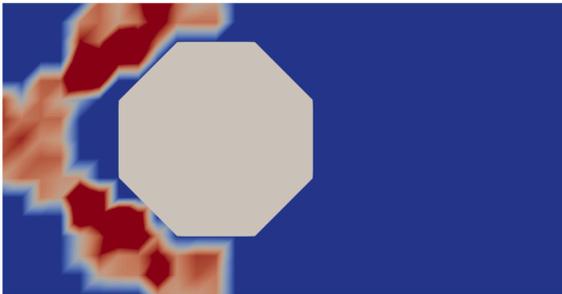
D_1Q_2 – multiple time steps and multiple obstacles. Fig. 20 shows the simulations of a D_1Q_2 system with 16 gridpoints, where the gridpoints with indices 0 and 4 are initialized in the $|11\rangle$ state. The solid domain is depicted in grey and occupies the 2nd and 3rd



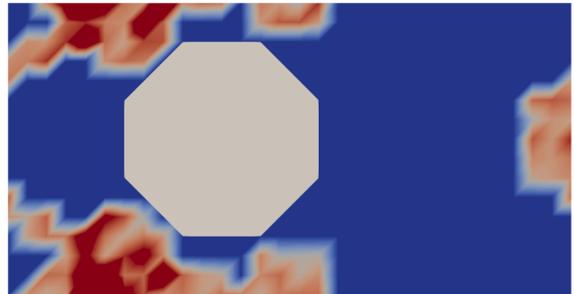
(a) The system in the 1st time step.



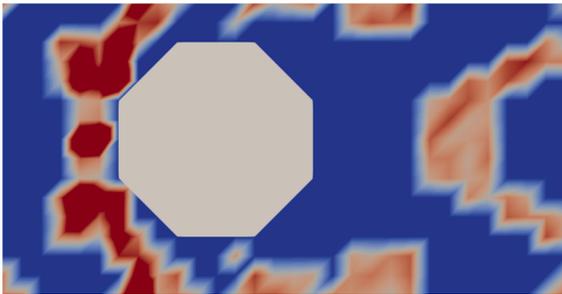
(b) The system in the 5th time step.



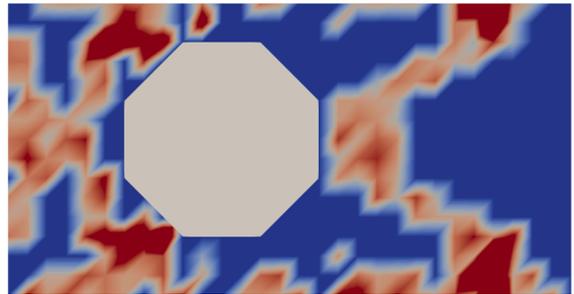
(c) The system in the 10th time step.



(d) The system in the 15th time step.



(e) The system in the 20th time step.



(f) The system in the 25th time step.

Fig. 22. Evolution of a D_2Q_4 system for 25 time steps.

gridpoints. The edges of the fluid of the domain are prescribed periodic boundary conditions. The quantum circuit uses 4 qubits to encode the 16 gridpoints, and an additional 18 qubits to encode the locality of each gridpoint for 4 time steps. Reinitialization is carried out every 4 time steps by means of the pointwise method, and since there are no non-trivial equivalence classes in D_1Q_2 , reinitialization is straightforward and exact. Darker shades of red indicate a higher particle occupancy (2), while lighter shades indicate that a single particle is present. Information is extracted from the quantum state by means of shots taken at the end of each circuit. Fig. 20a shows the initial state of the system, while Fig. 20b–d show the state of the system following 1, 2, and 3 circuit repetitions, with each repetition evolving the system for 4 discrete time steps.

Fig. 21 shows the more granular evolution of a similar 16 gridpoint D_1Q_2 system with two solid obstacles placed spanning the gridpoints indexed 2, 3, 7, and 8. The 0 and 4 gridpoints are initialized in the $|11\rangle$ state. Unlike the previous example, the quantum circuit of this system performs restarts after every time step and as such only requires 6 qubits to encode the Space-Time stencil of one time step, and 10 qubits in total. Shades of red indicate the presence of two particles at a gridpoint, whereas lighter shades of blue and white indicate the presence of a single particle. Numerical inaccuracies are again due to the postprocessing having been carried out on shot, as opposed to expectation values. Fig. 21a through Fig. 21f show the evolution of the system on a step-by-step

basis, with the consistent application of periodic boundary conditions at the edges of the fluid domain, and bounce-back boundary conditions at the boundary of the solid objects.

D_2Q_4 – staircase approximation. Fig. 22 shows the evolution of D_2Q_4 system that simulates a 32×16 lattice, with a solid object in the shape of a circle centered at (12, 8) with a radius of 5 gridpoint intervals. The initial conditions are set up such that the gridpoints within the volume spanning $(0, 0) \times (2, 15)$ are occupied by a single particle moving in the positive x direction. To make the simulation feasible for commonplace classical hardware available today, we perform reinitializations after each time step by reducing the superposition created by the collision operator to a deterministic state. This way, no decomposition of large unitary matrices is required and such simulations can be carried out efficiently. This reinitialization mechanism is also implemented in the QLBM library. The simulation required 29 qubits: 20 qubits to encode the extended computational basis state that performs streaming and collision for 1 time step, and $\log_2 32 + \log_2 16 = 9$ qubits to entangle the velocity register to the physical grid. The choice of simulating one time step is made for pragmatical reasons, as the simulation of two time steps would require 52 qubits, which exceeds practical computational capabilities.

Fig. 22a and b show the collisionless advection of particles through space, as the $|1000\rangle$ is preserved by streaming. These steps are exact, even with the added reinitialization steps. Following the interaction with the solid domain, particles are bounced back, such that the collision-sensitive basis state $|1010\rangle$ occurs at the left boundary of the object. Following several more time steps, the reinitialization approximation leads to higher particle density areas (highlighted by the darker shade of red) occurring near the boundary of the object, as shown in Fig. 22. The process continues in Figs. 22b through f, where the additional complexity of periodic boundary condition is accounted for.

7. Conclusion

This work introduced novel building blocks for every step of the quantum lattice gas automata loop, from initialization to measurement. We described the implementation of expressive pointwise methods and efficient volumetric alternatives for the application of initial and boundary conditions. We extended the efficient volumetric boundary conditions to novel geometric patterns that can be applied to efficiently model more complex shapes. We developed and generalized previous one-to-one collision operators to arbitrary velocity discretization under a new, less restrictive interpretation of equivalence classes. Finally, we outlined the properties of observables that recover quantities of interest from the quantum state. Each building block is accompanied by a complexity analysis in terms of one- and two-qubit gates, and all implementations of described building blocks are made available in an open-source library.

Although we emphasized the utility of the introduced building blocks using the general form of the Space-Time data encoding [31], only minimal modifications are required to implement them in the QLGA algorithms of, for instance, Love [61], Kocherla et al. [62], Zamora et al. [64], and [32,66], which we hope will benefit several independent directions of QLGA research. In the future, we aim to increase the computational efficiency of QLGA and investigate its applicability to realistic use cases. On the computational side, reducing the complexity of each building block and investigating more efficient decompositions could improve the situations in which QLGA may become a feasible alternative to classical solvers. On the use case side, we aim to empirically investigate the implications of our novel collision operator with respect to different flow regimes and governing equations, and further tailor the presented building blocks to specific real-world applications.

CRedit authorship contribution statement

Călin A. Georgescu: Conceptualization, Formal analysis, Investigation, Methodology, Software, Visualization, Writing - original draft, Writing - review & editing; **Merel A. Schalkers:** Conceptualization, Formal analysis, Investigation, Methodology, Software, Writing - review & editing, Visualization; **Matthias Möller:** Conceptualization, Formal analysis, Investigation, Methodology, Software, Writing - review & editing, Visualization.

Data availability

The replication package contains containing all data used in this research.

Declaration of competing interest

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

Acknowledgments

We gratefully acknowledge support from the joint research program *Quantum Computational Fluid Dynamics* by Fujitsu Limited and Delft University of Technology, co-funded by the Netherlands Enterprise Agency under project number PPS23-3-03596728. We furthermore thank Monica Lăcătuș, Alex Sturges, and Jingya Li for many fruitful discussions regarding the QLGA algorithm and the contents of this manuscript.

References

- [1] K. Wijesooriya, D. Mohotti, C.-K. Lee, P. Mendis, A technical review of computational fluid dynamics (CFD) applications on wind design of tall buildings and structures: past, present and future, *J. Build. Eng.* 74 (2023) 106828.
- [2] M. Mani, A.J. Dorgan, A perspective on the state of aerospace computational fluid dynamics technology, *Annu. Rev. Fluid Mech.* 55 (1) (2023) 431–457.
- [3] A. Zingaro, M. Bucelli, R. Piersanti, F. Regazzoni, A. Quarteroni, et al., An electromechanics-driven fluid dynamics model for the simulation of the whole human heart, *J. Comput. Phys.* 504 (2024) 112885.
- [4] T.N. Theis, H.-S.P. Wong, The end of moore's law: a new beginning for information technology, *Comput. Sci. Eng.* 19 (2) (2017) 41–50.
- [5] M.A. Nielsen, I.L. Chuang, *Quantum Computation and Quantum Information*, Cambridge university press, 2010.
- [6] R. Demirdjian, D. Gunlycke, C.A. Reynolds, J.D. Doyle, S. Tafur, Variational quantum solutions to the advection–diffusion equation for applications in fluid dynamics, *Quantum Inf. Process.* 21 (9) (2022) 322.
- [7] O. Kyriienko, A.E. Paine, V.E. Elfving, Solving nonlinear differential equations with differentiable quantum circuits, *Phys. Rev. A* 103 (5) (2021) 052416.
- [8] Y. Sato, R. Kondo, S. Koide, H. Takamatsu, N. Imoto, Variational quantum algorithm based on the minimum potential energy for solving the Poisson equation, *Phys. Rev. A* 104 (5) (2021) 052409.
- [9] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* 378 (2019) 686–707.
- [10] S. Berger, N. Hosters, M. Möller, Trainable embedding quantum physics informed neural networks for solving nonlinear PDEs, *Sci. Rep.* 15 (1) (2025) 1–14.
- [11] G. Panichi, S. Corli, E. Prati, Quantum physics informed neural networks for multi-variable partial differential equations, arXiv preprint arXiv:2503.12244 (2025).
- [12] J.R. McClean, S. Boixo, V.N. Smelyanskiy, R. Babbush, H. Neven, Barren plateaus in quantum neural network training landscapes, *Nat. Commun.* 9 (1) (2018) 4812.
- [13] A.W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, *Phys. Rev. Lett.* 103 (15) (2009) 150502.
- [14] A. Montanaro, S. Pallister, Quantum algorithms and the finite element method, *Phys. Rev. A* 93 (3) (2016) 032324.
- [15] Z.-Y. Chen, C. Xue, S.-M. Chen, B.-H. Lu, Y.-C. Wu, J.-C. Ding, S.-H. Huang, G.-P. Guo, Quantum approach to accelerate finite volume method on steady computational fluid dynamics problems, *Quantum Inf. Process.* 21 (4) (2022) 137.
- [16] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, S. Kais, Quantum algorithm and circuit design solving the Poisson equation, *New J. Phys.* 15 (1) (2013) 013021.
- [17] S. Aaronson, Read the fine print, *Nat. Phys.* 11 (4) (2015) 291–293.
- [18] D.A. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*, Springer, 2004.
- [19] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E.M. Viggen, The lattice Boltzmann method, *Springer Int. Publishing* 10 (978-3) (2017) 4–15.
- [20] A.M. Childs, N. Wiebe, Hamiltonian simulation using linear combinations of unitary operations, arXiv preprint arXiv:1202.5822 (2012).
- [21] L. Budinski, Quantum algorithm for the advection–diffusion equation simulated with the lattice Boltzmann method, *Quantum Inf. Process.* 20 (2) (2021) 57.
- [22] L. Budinski, Quantum algorithm for the Navier–Stokes equations by using the streamfunction-vorticity formulation and the lattice Boltzmann method, *Int. J. Quantum Inform.* 20 (02) (2022) 2150039.
- [23] D. Wawrzyniak, J. Winter, S. Schmidt, T. Indinger, C.F. Janßen, U. Schramm, N.A. Adams, A quantum algorithm for the lattice-Boltzmann method advection-diffusion equation, *Comput. Phys. Commun.* 306 (2025) 109373. <https://doi.org/10.1016/j.cpc.2024.109373>
- [24] A. Tiwari, J. Iaconis, J. Jojo, S. Ray, M. Roetteler, C. Hill, J. Pathak, Algorithmic advances towards a realizable quantum lattice Boltzmann method, arXiv preprint arXiv:2504.10870 (2025).
- [25] L. Xu, M. Li, L. Zhang, H. Sun, J. Yao, Improved quantum lattice Boltzmann method for advection-diffusion equations with a linear collision model, *Phys. Rev. E* 111 (4) (2025) 045305.
- [26] B.N. Todorova, R. Steijl, Quantum algorithm for the collisionless Boltzmann equation, *J. Comput. Phys.* 409 (2020) 109347.
- [27] L. Budinski, O. Niemimäki, R. Zamora-Zamora, V. Lahtinen, Efficient parallelization of quantum basis state shift, *Quantum Sci. Technol.* 8 (4) (2023) 045031.
- [28] M.A. Schalkers, M. Möller, Efficient and fail-safe quantum algorithm for the transport equation, *J. Comput. Phys.* 502 (2024) 112816.
- [29] R. Steijl, Quantum algorithms for nonlinear equations in fluid mechanics, *Quantum Comput. Commun.* 2 1–21 (2020).
- [30] Y. Moawad, W. Vanderbauwhede, R. Steijl, Investigating hardware acceleration for simulation of CFD quantum circuits, *Front. Mech. Eng.* 8 (2022) 925637.
- [31] M.A. Schalkers, M. Möller, On the importance of data encoding in quantum Boltzmann methods, *Quantum Inf. Process.* 23 (1) (2024) 20.
- [32] N. Fonio, P. Sagaut, G. Di Molfetta, Quantum collision circuit, quantum invariants and quantum phase estimation procedure for fluid dynamic lattice gas automata, *Comput. Fluids* 299 (2025) 106688. <https://doi.org/10.1016/j.compfluid.2025.106688>
- [33] D. Wawrzyniak, J. Winter, S. Schmidt, T. Indinger, C.F. Janßen, U. Schramm, N.A. Adams, Dynamic circuits for the quantum lattice-Boltzmann method, arXiv preprint arXiv:2502.02131 (2025).
- [34] W. Itani, S. Succi, Analysis of Carleman linearization of lattice Boltzmann, *Fluids* 7 (1) (2022) 24.
- [35] W. Itani, K.R. Sreenivasan, S. Succi, Quantum algorithm for lattice Boltzmann (QALB) simulation of incompressible fluids with a nonlinear collision term, 2023. <https://arxiv.org/abs/2304.05915>.
- [36] C. Sanavio, S. Succi, Lattice Boltzmann–Carleman quantum algorithm and circuit for fluid flows at moderate Reynolds number, *AVS Quantum Sci.* 6 (2) (2024).
- [37] C. Sanavio, W.A. Simon, A. Ralli, P. Love, S. Succi, Carleman-lattice-Boltzmann quantum circuit with matrix access oracles, 2025. <https://arxiv.org/abs/2501.02582>.
- [38] E.D. Kumar, S.H. Frankel, Decomposition of nonlinear collision operator in quantum lattice Boltzmann algorithm, *Europhys. Lett.* 148 (3) (2024) 38003.
- [39] J. v Neumann, A.W. Burks, *Theory of Self-Reproducing Automata*, University of Illinois Press Urbana, 1966.
- [40] B. Chopard, Cellular automata and lattice Boltzmann modeling of physical systems, in: *Handbook of Natural Computing*, Springer, 2012, pp. 287–331.
- [41] R. Benzi, S. Succi, M. Vergassola, The lattice Boltzmann equation: theory and applications, *Phys. Rep.* 222 (3) (1992) 145–197.
- [42] B.M. Boghosian, Lattice gases and cellular automata, *Future Gener. Comput. Syst.* 16 (2) (1999) 171–185. [https://doi.org/10.1016/S0167-739X\(99\)00045-X](https://doi.org/10.1016/S0167-739X(99)00045-X)
- [43] J. Hardy, Y. Pomeau, O. De Pazzis, Time evolution of a two-dimensional model system. I. Invariant states and time correlation functions, *J. Math. Phys.* 14 (12) (1973) 1746–1759.
- [44] J. Hardy, O. De Pazzis, Y. Pomeau, Molecular dynamics of a classical lattice gas: transport properties and time correlation functions, *Phys. Rev. A* 13 (5) (1976) 1949.
- [45] U. Frisch, B. Hasslacher, Y. Pomeau, Lattice-gas automata for the Navier-Stokes equation, *Phys. Rev. Lett.* 56 (14) (1986) 1505.
- [46] R.P. Feynman, Simulating physics with computers, *Int. J. Theor. Phys.* 21 (1982) 467–488. <https://doi.org/10.1007/BF02650179>
- [47] D.A. Meyer, From quantum cellular automata to quantum lattice gases, *J. Stat. Phys.* 85 (1996) 551–574.
- [48] D.A. Meyer, Unitarity in one dimensional nonlinear quantum cellular automata, arXiv preprint quant-ph/9605023 (1996).
- [49] D.A. Meyer, Quantum mechanics of lattice gas automata: one-particle plane waves and potentials, *Phys. Rev. E* 55 (5) (1997) 5261.
- [50] D.A. Meyer, Quantum lattice gases and their invariants, *Int. J. Modern Phys. C* 8 (04) (1997) 717–735.
- [51] D.A. Meyer, Quantum mechanics of lattice gas automata: boundary conditions and other inhomogeneities, *J. Phys. A Math. Gen.* 31 (10) (1998) 2321.
- [52] B.M. Boghosian, W. Taylor, Quantum lattice-gas models for the many-body Schrödinger equation, *Int. J. Modern Phys. C* 8 (04) (1997) 705–716.
- [53] B.M. Boghosian, W. Taylor, IV, Simulating quantum mechanics on a quantum computer, *Phys. D* 120 (1-2) (1998) 30–42.
- [54] S. Succi, R. Benzi, Lattice Boltzmann equation for quantum mechanics, *Phys. D* 69 (3-4) (1993) 327–332.
- [55] S. Succi, Numerical solution of the Schrödinger equation using discrete kinetic theory, *Phys. Rev. E* 53 (1996) 1969–1975. <https://doi.org/10.1103/PhysRevE.53.1969>
- [56] J. Yepez, Quantum computation of fluid dynamics, in: *NASA International Conference on Quantum Computing and Quantum Communications*, Springer, 1998, pp. 34–60.
- [57] J. Yepez, Quantum lattice-gas model for the diffusion equation, *Int. J. Modern Phys. C* 12 (09) (2001) 1285–1303.

- [58] J. Yepez, Quantum lattice-gas model for computational fluid dynamics, *Phys. Rev. E* 63 (4) (2001) 046702.
- [59] J. Yepez, Quantum lattice-gas model for the Burgers equation, *J. Stat. Phys.* 107 (2002) 203–224.
- [60] J. Yepez, B. Boghosian, An efficient and accurate quantum lattice-gas model for the many-body Schrödinger wave equation, *Comput. Phys. Commun.* 146 (3) (2002) 280–294.
- [61] P. Love, On quantum extensions of hydrodynamic lattice gas automata, *Condensed Matter* 4 (2) (2019) 48.
- [62] S. Kocherla, Z. Song, F.E. Chrit, B. Gard, E.F. Dumitrescu, A. Alexeev, S.H. Bryngelson, Fully quantum algorithm for mesoscale fluid simulations with application to partial differential equations, *AVS Quantum Sci.* 6 (3) (2024) 033806.
- [63] A.D.B. Zamora, L. Budinski, O. Niemimäki, V. Lahtinen, Efficient quantum lattice gas automata, *Comput. Fluids* 286 (2025) 106476.
- [64] A.D.B. Zamora, L. Budinski, V. Lahtinen, P. Sagaut, Lattice gas automata with floating-point numbers: A connection between molecular dynamics and lattice Boltzmann method for quantum computers, *Phys. Rev. E* 112 (1) (2025) 015305.
- [65] T. Blommel, A.J. Wagner, Integer lattice gas with Monte Carlo collision operator recovers the lattice Boltzmann method with Poisson-distributed fluctuations, *Phys. Rev. E* 97 (2) (2018) 023310.
- [66] N. Fonio, L. Budinski, V. Lahtinen, P. Sagaut, Adaptive lattice gas algorithm: classical and quantum implementations, *Phys. Rev. E* 112 (3) (2025) 035302.
- [67] M.A. Schalkers, M. Möller, Momentum exchange method for quantum Boltzmann methods, *Comput. Fluids* 285 (2024) 106453.
- [68] A.Y. Kitaev, Quantum measurements and the Abelian stabilizer problem, arXiv preprint quant-ph/9511026 (1995).
- [69] M. Mosca, C. Zalka, Exact quantum Fourier transforms and discrete logarithm algorithms, *Int. J. Quantum Inf.* 2 (1) (2004) 91–100.
- [70] J. Kempe, Quantum random walks: an introductory overview, *Cont. Phys.* 44 (4) (2003) 307–327.
- [71] T. Toffoli, N. Margolus, *Cellular Automata Machines: A New Environment for Modeling*, MIT press, 1987.
- [72] E. Deza, M. Deza, *Figurate Numbers*, World Scientific, 2012.
- [73] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin, H. Weinfurter, Elementary gates for quantum computation, *Phys. Rev. A* 52 (5) (1995) 3457.
- [74] T.G. Draper, Addition on a quantum computer, arXiv preprint quant-ph/0008033 (2000).
- [75] L. Ruiz-Perez, J.C. Garcia-Escartin, Quantum arithmetic with the quantum Fourier transform, *Quantum Inf. Process.* 16 (2017) 1–14.
- [76] D.R. Musk, A comparison of quantum and traditional Fourier transform computations, *Comput. Sci. Eng.* 22 (6) (2020) 103–110.
- [77] R. Heese, P. Bickert, A.E. Niederle, Representation of binary classification trees with binary features by quantum circuits, *Quantum* 6 (2022) 676.
- [78] C.S. Nor Azwadi, T. Tanahashi, Three-dimensional thermal lattice Boltzmann simulation of natural convection in a cubic cavity, *Int. J. Modern Phys. B* 21 (01) (2007) 87–96.
- [79] B. Wang, Z. Meng, Y. Zhao, Y. Yang, Quantum lattice Boltzmann method for simulating nonlinear fluid dynamics, arXiv preprint arXiv:2502.16568 (2025).
- [80] S. Herbert, J. Sorci, Y. Tang, Almost-optimal computational-basis-state transpositions, *Phys. Rev. A* 110 (1) (2024) 012437.
- [81] U. Frisch, D. Dhumieres, B. Hasslacher, P. Lallemand, Y. Pomeau, J. Rivet, Lattice gas hydrodynamics in two and three dimensions, *Complex Syst.* 1 (1987) 649–707.
- [82] A.J.C. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation, *J. Fluid Mech.* 271 (1994) 285–309.
- [83] A.J.C. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 2. Numerical results, *J. Fluid Mech.* 271 (1994) 311–339.
- [84] C.A. Georgescu, M.A. Schalkers, M. Möller, qlbm—a quantum lattice Boltzmann software framework, *Comput. Phys. Commun.* 315 (2025) 109699.
- [85] C.A. Georgescu, M.A. Schalkers, M. Möller, Replication package for "fully quantum lattice gas automata building blocks for computational basis state encodings, (2025) <https://doi.org/10.5281/zenodo.15636196>.
- [86] A. Javadi-Abhari, M. Treinish, K. Krsulich, C.J. Wood, J. Lishman, J. Gacon, S. Martiel, P.D. Nation, L.S. Bishop, A.W. Cross, B.R. Johnson, J.M. Gambetta, Quantum computing with Qiskit, 2024. <https://doi.org/10.48550/arXiv.2405.08810>
- [87] J. Ahrens, B. Geveci, C. Law, C. Hansen, C. Johnson, 36-paraview: an end-user tool for large-data visualization, *The Visualization Handbook* 717 (2005) 50038–50039.
- [88] A. Shakeel, P.J. Love, When is a quantum cellular automaton (QCA) a quantum lattice gas automaton (QLGA)?, *J. Math. Phys.* 54 (9) (2013).
- [89] T. Farrelly, A review of quantum cellular automata, *Quantum* 4 (2020) 368.