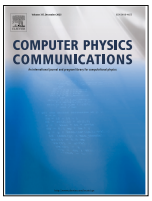
Computer Physics Communications 321 (2026) 110040

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc



Computational Physics

Quantum lattice boltzmann method for multiple time steps without reinitialization for linear advection-Diffusion problems

Aaron Nagel [a,b,*], Johannes Löwe [a]

[a] German Aerospace Center (DLR), Bunsenstrasse 10, Göttingen, D-37073, Lower Saxony, Germany
[b] University of Göttingen, Faculty of Physics, Friedrich-Hund-Platz 1, Göttingen, D-37077, Lower Saxony, Germany

ARTICLE INFO

Editor: Dr W Jong

Keywords:
Quantum computing
Quantum lattice boltzmann method
Fluid dynamics
Linear advection-Diffusion.

ABSTRACT

To simulate highly-resolved flow fields, we extend the Quantum Lattice Boltzmann Method (QLBM) to be able to simulate multiple time steps without state extraction or reinitialization. We adjust and extend given QLBM approaches from the literature to completely remove the need to measure or reinitialize the flow field in between the simulation time steps. Therefore, our algorithm does not require to sample the entire flow field at any time. We solve the linear advection-diffusion problem with periodic boundary conditions and derive all necessary equations and build the corresponding quantum circuit diagrams, including details on the QLBM blocks and explicitly drawing the circuit gates. We discuss the general decay of a QLBM step and how that effects our algorithm. The new algorithm is verified on 1D and 2D test cases using the *shot* method of IBMs *Qiskit* package. We show excellent agreement and convergence between our QLBM and the classical Lattice Boltzmann method. The conclusion section includes a discussion on the advantages of our algorithm as well as limitations and to what extent it is more efficient.

1. Introduction

The effort in developing quantum algorithms has increased rapidly in recent years. But the idea of using quantum particles as operation units in machines, similar as the classical bit in a computer, is not new at all. Richard Feynman as one famous example, has already published first ideas on *Quantum Mechanical Computers* [1] back in the 1980s, and first important quantum algorithms have been developed in 1990s, the well known *Shor's* [2] and *Grover's* [3] algorithms. But with the advancements in quantum hardware after the year of 2000, also the interest has become more and more present. With the increase of usable qubits, especially the promise of an exponential system size scaling in quantum operation units raised a large interest in scale resolving fluid flow simulations.

The first ideas in approaching computational fluid dynamics (CFD) problems with quantum algorithms were proposed by Meyer [4] and Yepez [5] with quantum lattice gas approaches. Improved models by the lattice Boltzmann methods (LBM) have been developed further by the pure fluid transport with a collisionless Boltzmann equation [6,7] using a quantum streaming operation. Further work included a quantum collision step by separation of the collision procedure into a sum of unitary operations [8–10]. These algorithms represent the first "*fully quantum*" LBM algorithms for a single LBM time step. But such a full LBM routine as presented in the literature requires a full state extraction and reinitialization in between every time step. This again results in a complete loss of its quantum advantage in system size scaling, since a full state extraction scales linear in the grid resolution with the number of shots needed to resolve the flow field. That being said, literature claiming to develop a "*fully quantum*" LBM algorithm has to be judged carefully, since this can mean that a quantum LBM is developed only for a single time step.

Additionally to the LBM algorithms of linear advection-diffusion problems, the nonlinearity of the collision operator has been approached using a Carleman linearization within the LBM by Itani et al. [11,12] and for the Burgers equation by Liu et al. [13]. However, the linearization comes at the cost of temporal instability for large nonlinearities as they commonly occur in typical aerospace problems. Since our paper will not cover nonlinearities, the topic itself has to be addressed in future work.

In this paper, we present our algorithm of a quantum lattice Boltzmann method (QLBM) for a linear advection-diffusion equation for multiple time steps which does not require measurement or reinitialization at any time in between the simulation time steps. This is to the best of the authors knowledge the first algorithm of its kind that can perform multiple QLBM time steps without any kind or mid-circuit measurement,

* Corresponding author.
E-mail address: aaron.nagel@dlr.de (A. Nagel).

https://doi.org/10.1016/j.cpc.2026.110040
Received 22 October 2025; Received in revised form 9 January 2026; Accepted 16 January 2026
Available online 18 January 2026

state extraction or reinitialization of the quantum state. Our new algorithm can perform all QLBM time steps up to the end of the simulation, without the need to ever *having* to extract the flow field at any time. For verification reasons, we show in our result section that our algorithm reproduces the correct full flow field up to a certain sampling error. But in contrast to other algorithms, our algorithm allows to be used in such a way, that for the flow field properties you are interested in, quantities can be calculated without ever having to extract the flow field, not even at the end of the simulation. This way, a simulation of a body in the flow field could return a scalar value like a lift or drag coefficient value, while the entire fully resolved flow field was never measured. Our algorithm is a first approach to an actually fully quantum algorithm for multiple time steps, even when still limited in the total number of simulation time that is feasible to simulate.

We further derive a quantum collision routine only using controlled *RY* gates, which turned out to be a similar approach as in for example Xu et al. [14] and Wawrzyniak et al. [15]. However, our algorithm is formulated in a generalized way which can be directly applied for arbitrary LBM velocity set stencils. Both of these papers also avoid the reinitialization in each time steps, but the main difference of our paper are no mid-circuit measurements, i.e. no non-unitary projections, but instead solely unitary QLBM blocks and a fully unitary RE-PREP block as preparation for the next time step. Since the idea of the RE-PREP block is a streaming of velocity subspaces, its complexity essentially scales similar to the streaming block.

The paper is structured as follows: the *Methods* section starts with a short overview of the lattice Boltzmann method (LBM) that is used which will be translated into a quantum lattice Boltzmann method (QLBM) for an advection-diffusion problem. Further subsections describe the necessary quantum state amplitude encoding structure, the quantum collision step that is used, the quantum streaming step that is adopted from the literature, the procedure to calculate the macroscopic values as a quantum state and finally the quantum re-preparation step to make the algorithm work for multiple time steps. The subsections lead through the mathematical derivations and show the implementation of the quantum circuit blocks explicitly by quantum gates. The decay of the quantum state amplitude for multiple time steps is quantified in a final subsection. The following *Verification* section shows 1D and 2D results of an advection-diffusion process and verifies the QLBM to recover the correct LBM solution using the *shot* method of IBMs simulator *Qiskit* [16]. For the 2D test cases, deviations and convergence of QLBM to LBM are discussed. In the final *Conclusion* section, a discussion on the advantage of the algorithm and limitations are included.

2. Methods

2.1. The advection-diffusion equation

In this paper, we focus on solving linear transport equations with the lattice Boltzmann method (LBM), in particular the advection-diffusion equation of a scalar Φ:

$$\frac{\partial \Phi}{\partial t} + u_j \frac{\partial \Phi}{\partial x_j} = D \frac{\partial^2 \Phi}{\partial x_j^2}, \tag{1}$$

with a uniform constant flow velocity u_j and a constant diffusion coefficient D, using the Einstein's index summation notation over the spacial directions $j \in \{1,2,3\}$. An extension of a spacially variable flow velocity is generally possible with our approach by conditioning the collision operations on different grid locations. This will be addressed in future work.

2.2. The lattice boltzmann method

Boltzmann methods generally describe the dynamics of fluids from a statistical perspective of a particle ensemble description of the fluid. This comes with advantages because it allows to describe the fluid dynamic with distinct *streaming* and *collision* steps to determine the change and the transport of the fluid property in time. These two steps are evaluated in two fully separated steps, which then can be developed individually to a quantum algorithm to account for the desired physics modeling.

In this work, we will use the lattice Boltzmann method (LBM) using a fairly simple collision description by the Bhatnagar-Gross-Krook (BGK) collision operator to develop our algorithm.

The Boltzmann equation [17,18] is a transport equation that determines the change of a velocity distribution function f, which is a probability density function for a local particle ensemble of positions and velocities $f(\mathbf{x}, \mathbf{v}, t)$ in phase space $\mathcal{H}$. The Boltzmann equation describes the change of the local fluid quantity by the advective transport on the left hand side and the change due to the collision by the collision operator $S_{\text{coll}}(f)$:

$$\frac{\partial f}{\partial t} + v_j \frac{\partial f}{\partial x_j} + F_j \frac{\partial f}{\partial v_j} = S_{\text{coll}}(f), \tag{2}$$

which implies the Einstein's index summation notation over the spacial directions $j \in \{1,2,3\}$ and where F_j can be some additional external acceleration.

We assume no external forces $F_j = 0$ and use a collision relaxation by the BGK operator [17]

$$S_{\text{BGK}}(f) = -\frac{1}{\tau}\left(f - f^{\text{eq}}\right), \tag{3}$$

where τ is the relaxation time towards a local equilibrium f^{eq}, which is determined by the diffusion coefficient D. In further simplifications, we will choose a fixed relaxation time and model the diffusion by the weight parameters of the LBM.

The position space is discretized on a grid and the velocity is discretized to a set of vectors $\mathbf{e}_i$ that span the velocity space, which are not necessarily linear independent vectors only. The vectors $\mathbf{e}_i$ are chosen in length and direction such that they point exactly onto neighbouring lattice nodes. Each discrete velocity direction $\mathbf{e}_i$ models a distribution function f_i, which has its own local equilibrium to relax to. The discretized Boltzmann equation results in

$$\frac{1}{\Delta t}\left(f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) - f_i(\mathbf{x}, t)\right) = -\frac{1}{\tau}\left(f_i(\mathbf{x}, t) - f_i^{\text{eq}}(\mathbf{x}, t)\right). \tag{4}$$

The equilibrium distribution f_i^{eq} is approximated linearly for an advection diffusion process. For a single grid node, a scalar quantity Φ with constant background velocity $\mathbf{u}$ on that grid node determines the equilibrium distribution in the i-th discrete velocity direction by [17]

$$f_i^{\text{eq}} = w_i \left(1 + \frac{\mathbf{u} \cdot \mathbf{e}_i}{c_s^2}\right) \Phi, \tag{5}$$

which defines the factor k_i for the collision matrix used by the QLBM algorithm

$$k_i \equiv w_i \left(1 + \frac{\mathbf{u} \cdot \mathbf{e}_i}{c_s^2}\right). \tag{6}$$

Here, the diffusivity D enters the equilibrium distribution via the speed of sound c_s that determines the discretization weights w_i. This allows to calculate the change of the distribution function $\hat{f}_i$ for the next time step due to *collision*

$$\hat{f}_i(\mathbf{x}, t) = \left(1 - \frac{\Delta t}{\tau}\right) f_i(\mathbf{x}, t) + \frac{\Delta t}{\tau} f_i^{\text{eq}}(\mathbf{x}, t), \tag{7}$$

and the *streaming* to neighbouring nodes for the next time step $t + \Delta t$ for every updated velocity direction $\hat{f}_i$

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) = \hat{f}_i(\mathbf{x}, t). \tag{8}$$

With the updated distribution functions on each node, the *macroscopic* scalar quantity on each node can by re-calculated by the sum of the distribution functions f_i [17]

$$\Phi = \sum_i f_i. \tag{9}$$

2.2.1. Diffusion and lattice boltzmann weights

For the velocity distribution functions in different velocity directions, the equilibrium function of each discrete velocity direction f_i^{eq} scales with a weighting factor w_i. These weighting factors are obtained by evaluating moment equations of different orders to ensure conservation properties [17]. The resulting equations relate the weighting factors with the speed of sound, and therefore indirectly the diffusion constant. Using a Chapman-Enskog expansion, a relation between diffusion D and relaxation time τ with the speed of sound c_s for discretization parameters $\mathrm{d}x$ and $\mathrm{d}t$ within the LBM can be derived [18], which leads to a diffusion of

$$D = c_s^2\left(\tau - \frac{\Delta t}{2}\right). \tag{10}$$

The diffusion is typically modeled with the relaxation time τ, but the algorithm design only allows for a full relaxation within one time step yet, i.e. $\tau = \Delta t = 1$ as the literature proposes [8,14,15]. For a simulation that is scaled in such a way that the distribution functions relax fully to their equilibrium within one time step, the diffusion is either fixed or can only be modeled by scaling the speed of sound c_s^2 of the simulation. To scale the speed of sound, the weighting factors w_i are adjusted accordingly. With sets of weighting factors other than the standard set, the moment equations may only be fulfilled up to a certain order. Depending on the physical system, this may or may not violate conservation symmetries, depending on the given order of symmetry that our system shows [17]. Whether a set of weights, other than the standard set is physically justified has to be decided by the end user for the specific problem of interest. Independently of that choice, our algorithm is capable of utilizing arbitrary sets of weights.

For our linear, isotropic advection-diffusion problem, we test on the standard weight set and additionally on weight sets that fulfill the moment equations up to order four, which allows us to model different diffusion constants.

2.3. Quantum state encoding and initial state

To encode the flow field with N grid points in Q velocity directions, amplitude encoding is used to generate the quantum state that the QLBM algorithm will operate on. In order to make the algorithm work for multiple time steps with collision, streaming and calculation of the macroscopic quantities, we build on an initial state vector that contains the scalar grid information only in the first velocity direction f_1 while all other velocity directions have zero state amplitudes. So we explicitly choose an initial state, where all the scalar grid values are stored in the first velocity component, so for a grid node this means $\Phi = f_1$, and the full grid states is

$$F = \begin{pmatrix} f_1^1 \\ \vdots \\ f_1^N \\ f_2^1 \\ \vdots \\ f_2^N \\ f_Q^1 \\ \vdots \\ f_Q^N \end{pmatrix} = \begin{pmatrix} \Phi^1 \\ \vdots \\ \Phi^N \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{11}$$

which will also be referred in short notation grouping the grid into separate *velocity direction subspaces*:

$$F = (f_1,\ f_2,\ \ldots,\ f_Q)^T = (\Phi,\ 0,\ \ldots,\ 0)^T. \tag{12}$$

Note that for N grid points, we need at least $\lceil \log_2(N) \rceil$ qubits, generating $\lceil \log_2(N) \rceil^N$ grid points, where $\lceil \cdot \rceil$ is the ceiling function. Similarly, for a stencil with Q velocity directions, we need as many direction qubits $\#q_{\mathrm{dir}}$, such that they generate as many states as is at least the number of velocity directions needed, so

$$N_Q = 2^{\#q_{\mathrm{dir}}} \geq Q, \tag{13}$$

Table 1
Arrangement of the D1Q2 velocity direction subspaces with one direction qubit.

$\lvert q\rangle_{\mathrm{dir},x}$	$\lvert 0\rangle$	$\lvert 1\rangle$	=	$\lvert 0\rangle$	$\lvert 1\rangle$
direction f_i	f_1	f_2	=	→	←

Table 2
Arrangement of the D1Q3 velocity direction subspaces with two direction qubits.

$\lvert q_2 q_1\rangle_{\mathrm{dir},x}$	$\lvert 00\rangle$	$\lvert 01\rangle$	$\lvert 10\rangle$	$\lvert 11\rangle$	=	$\lvert 00\rangle$	$\lvert 01\rangle$	$\lvert 10\rangle$	$\lvert 11\rangle$
direction f_i	f_1	–	f_2	f_3	=	rest	–	→	←

Table 3
Arrangement of the D2Q9 velocity directions for the two x-direction qubits denoted by $\lvert q_2 q_1\rangle_x$ and the two y-direction qubits denoted by $\lvert q_4 q_3\rangle_y$, respectively.

	$\lvert 00\rangle_x$	$\lvert 01\rangle_x$	$\lvert 10\rangle_x$	$\lvert 11\rangle_x$		$\lvert 00\rangle_x$	$\lvert 01\rangle_x$	$\lvert 10\rangle_x$	$\lvert 11\rangle_x$
$\lvert 00\rangle_y$	f_1	-	f_2	f_3		rest	-	→	←
$\lvert 01\rangle_y$	-	-	-	-	=	-	-	-	-
$\lvert 10\rangle_y$	f_4	-	f_6	f_7		↑	-	↗	↖
$\lvert 11\rangle_y$	f_5	-	f_8	f_9		↓	-	↘	↙

which will expand the state vector from dimension $N \cdot Q$ to $N \cdot N_Q$, i.e. $(f_1,\ f_2,\ \ldots,\ f_{N_Q})^T = (\Phi,\ 0,\ \ldots,\ 0)^T$, with $N_Q - Q$ zero entries in the short velocity direction subspace notation.

In this paper, D1Q2 and D1Q3 as well as D2Q9 stencils are used. They are arranged as shown in the following equations, where each f_i represents a group of N gird points in direction i. For D1Q2, the state es encoded by one velocity qubit to span the two velocity directions

$$F_{\mathrm{D1Q2}} = (f_1, f_2)^T, \tag{14}$$

for D1Q3, 2 velocity qubits generate the state given by

$$F_{\mathrm{D1Q3}} = (f_1, 0, f_2, f_3)^T \tag{15}$$

and for D2Q9, 4 velocity qubits (2 for the x- and 2 for the y-direction) generate the space for the 9 velocity directions arranged by

$$F_{\mathrm{D1Q3}} = (f_1, 0, f_2, f_3, 0, 0, 0, 0, f_4, 0, f_6, f_7, f_5, 0, f_8, f_9)^T. \tag{16}$$

The arrangements are represented in Tables 1, 2 and 3.

2.4. The quantum lattice boltzmann method

To solve the advection-diffusion Eq. (1) with a quantum algorithm, the individual steps of the lattice Boltzmann method are solved by quantum algorithms, i.e. collision, streaming and updating the macroscopic values as shown in Fig. 1.

The data is encoded in the complex amplitude coefficients of the states of the grid qubits $\lvert q_{\mathrm{grid}}\rangle$, which for N grid qubits generate space for 2^N scalar values. For multiple dimensions, it is separated into groups of grid qubits for each dimension. For the collision operation, additional qubits $\lvert q_{\mathrm{dir}}\rangle$ are added, to enable to duplicate the grid in order to operate for the different LBM directions, given by the specific choice of the velocity set (LBM stencil). Also the streaming is then performed conditioned on the different direction qubits to transport in the corresponding directions. The macroscopic step finally merges the different directions to recalculate the new macroscopic quantities to update for a new local scalar quantity and local equilibrium. This QLBM routine generally requires all velocity subspaces but the first one to have zero probability amplitude in order to obtain the correct assignment by the collision step, as will be discussed in further subsections. Hence, the *reinitialization QLBM* algorithms reinitialize the state vector accordingly in each time step.

Our goal is to construct a scheme for a QLBM step that does not require a state extraction and reinitialization in between the time steps. Therefore, we propose a new extended scheme with a RE-PREP block

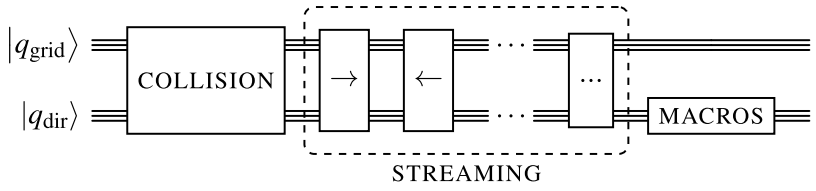


Fig. 1. Quantum circuit for a single quantum lattice Boltzmann time step.

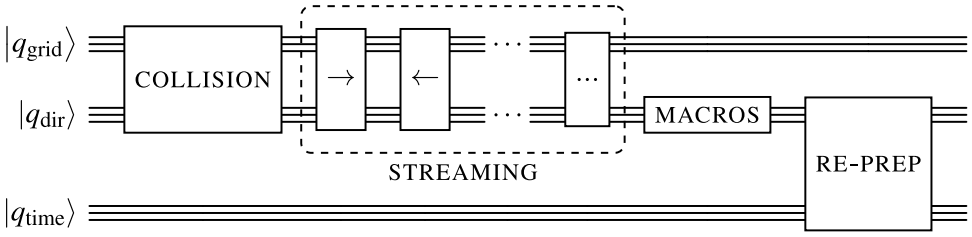


Fig. 2. Our quantum circuit extension of the QLBM routine for multiple quantum lattice Boltzmann time steps without state extraction or reinitialization.

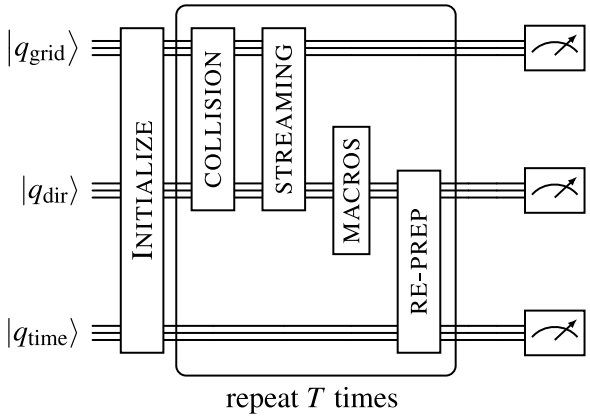


Fig. 3. Full quantum circuit of the QLBM simulation with the extension of the QLBM routine for T time steps without state extraction or reinitialization from Fig. 2.

and additional time qubits $|q_{\text{time}}\rangle$ as shown in Fig. 2, to enable a fully quantum algorithm for all time steps from start to end without a measurement of the state at any time in between. The full quantum circuit of all building block, including initialization, time loop and measurement, is shown in Fig. 3. Corresponding pseudocode referring to the building blocks is shown in the appendix in Section A.1.

2.4.1. The collision step

The collision step in the context of the LBM determines how each of the velocity probability distribution function f_i relaxes towards its local equilibrium for that specific velocity direction i. Assuming a full relaxation within a time step, i.e $\Delta t/\tau = 1$, the collision step is directly determined by the equilibrium distribution f^{eq} in Eq. (5). Although the diffusion usually dictates the relaxation time τ, this does not restrict the diffusion to a fixed value because we can model the diffusion by the modeled speed of sound c_s and weights w_i as described in Section 2.2.1.

For N grid points and Q velocity directions, our total state has dimension $N \cdot Q$, for which we have to calculate the relaxation for each entry. Collecting this in a state vector $F = (f_1^1, \ldots, f_1^N, \ldots, f_Q^1, \ldots, f_Q^N)^T$, the collision operation for each f_i^n, $i \in [1, Q]$, $n \in [1, N]$ as dictated by Eq. (7) can be written as the linear system

$$\hat{F} = AF, \tag{17}$$

where A is the diagonal collision matrix. The collision matrix is generally not unitary and thus can not be decomposed into a series of available unitary operations of a quantum computer. Therefore, Budinski [8] separates A into a sum of diagonal unitary matrices $A = (B_1 + B_2)/2$ with $B_{1,k,k} = A_{k,k} + i\sqrt{1 - A_{k,k}^2}$ and $B_{2,k,k} = A_{k,k} - i\sqrt{1 - A_{k,k}^2}$, following a linear combination of unitaries (LCU) approach. The idea of the LCU approach is to add an additional ancilla qubits $|a\rangle$ and duplicate the grid coefficients up to a norm factor into the new ancilla generated subspace. Now, B_1 and B_2 can operate separately on the two subspaces by conditioning the operations on the ancilla state $|a\rangle$. To do so, B_1 operates conditioned on $|a\rangle = |0\rangle$ and while B_2 is performed conditioned where $|a\rangle = |1\rangle$ and the results in the two subspaces are summed in a last step (for summation, cf. the summation in Section 2.4.3).

After the LCU summation step, there remains a significant amount of amplitude of the complex amplitude coefficient of the states in the additional LCU ancilla subspace, that results in a loss of probability that we want to avoid for our algorithm. So instead of using the LCU, we want to mimic the distribution into the multiple directions dictated by A by a unitary collision operation U_{coll} in the first place.

The idea is to start a time step in a state that has all the scalar information in one velocity direction subspace (cf. Section 2.3) and then mimic the distribution of the collision operation along the velocity directions by a unitary matrix. To achieve starting in this state, the state is re-prepared at the end of each time step by the RE-PREP block as described in Section 2.4.4 and as sketched in Fig. 2. The choice of this state is made possible by our choice of $\Delta t/\tau = 1$ because after the collision the state relaxes fully into the equilibrium state, which only depends on Φ of the node, independent of how Φ is distributed along the different directions in the first place (recap $\Phi = \sum_i f_i$ for a specific grid point). So if Φ is fully captured by one velocity direction, we can make sure that our desired operation U_{coll} operates on these states according to the corresponding entries in A, while the remaining entries can be chosen arbitrarily, but in particular such that our desired matrix U_{coll} becomes unitary.

To find a general collision description, we start with the simplest QLBM stencil and generalize to arbitrary stencils further in the subsection. The simplest case for a QLBM collision is given by a **D1Q2** stencil, so by one spacial dimension and two velocity directions $Q = 2$, leading to a state vector $F = (f_1^1, \ldots, f_1^N, f_2^1, \ldots, f_2^N)^T$. Here, the collision matrix A has the following form:

$$A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}, \tag{18}$$

where the A_i are diagonal $N \times N$ matrices. With the state prepared as in Eq. (11), A_2 and A_4 do not contribute to the collision calculation. Additionally, according to Eq. (5), all the diagonal entries a_i inside an

A_i are all equal and we can write:

$$A = \left(\begin{array}{ccc|ccc} a_1 & & & a_2 & & \\ & \ddots & & & \ddots & \\ & & a_1 & & & a_2 \\ a_3 & & & a_4 & & \\ & \ddots & & & \ddots & \\ & & a_3 & & & a_4 \end{array}\right) = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \otimes \mathbb{1}_N. \tag{19}$$

This can be represented by a quantum operation using a 2×2 rotational gate, the $RY(\theta)$-gate, on one qubit and no operation ($\equiv$ identity operation $\mathbb{1}$) on N qubits. So the rotation operation has to fit the coefficients a_i, but only for a_1 and a_3, since our state is prepared in such a way that a_2 and a_4 do not contribute

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \equiv RY(\theta), \tag{20}$$

leading to the conditions for choosing θ such that the a_i correspond to map the vector Φ from Eq. (11) to f^{eq} in Eq. (5) of the corresponding velocity direction:

$$a_1 = k_1 \equiv w_1\left(1 + \frac{u \cdot e_1}{c_s^2}\right) \overset{!}{=} \cos(\theta/2), \tag{21}$$

$$a_3 = k_2 \equiv w_2\left(1 + \frac{u \cdot e_2}{c_s^2}\right) \overset{!}{=} \sin(\theta/2). \tag{22}$$

So applying $RY(\theta)$ to the state as prepared in Eq. (11), this operation effectively shifts a certain fraction from all Φ in the first velocity direction to the second velocity direction, such that the ratio of $k_1/(k_1 + k_2)$ is kept in the first velocity subspace while moving $k_2/(k_1 + k_2)$ into the second subspace. So essentially, we want to choose θ, such that we distribute our state $(\Phi, 0)^T$ to $(k_1\Phi, k_2\Phi)^T$, but this can not be achieved by unitary operation in general. So instead, we choose θ, such that we keep the proportion of k_1/k_2 for $(\cos(\theta/2)\Phi, \sin(\theta/2)\Phi)^T$. This leads to our condition and single solution for the argument θ:

$$\frac{k_1}{k_2} = \frac{\cos(\theta/2)}{\sin(\theta/2)} \iff \theta = 2\arccos\underbrace{\left(\frac{k_1}{\sqrt{k_1^2 + k_2^2}}\right)}_{\equiv n} = 2\arcsin\left(\frac{k_2}{\sqrt{k_1^2 + k_2^2}}\right). \tag{23}$$

Note, that instead of calculating the ratio of k_1 and k_2, only a normalized ratio with $n \equiv k_1/\sqrt{k_1^2 + k_2^2}$ and $k_2/\sqrt{k_1^2 + k_2^2}$ is calculated, which will lead to a certain decay discussed in Section 2.4.5.

Generalization

Extending this idea to **D1Q3** scheme, three velocity directions are modeled ($Q = 3$) using $N_Q = 2$ direction qubits that span four velocity direction subspaces. Here, the first velocity direction is the resting node direction $(f_1^1, \ldots, f_1^N)$, while the second and third direction are the left and right streaming direction, respectively, which are arranged as shown in Table 2. In order to include the additional direction, the same idea is used, starting with an initial state vector as in Eq. (11) and sequentially distributing to the additional direction subspaces. Now two steps are modeled: in the first step, an angle θ_1 is determined, such that the amount n_1 is kept in the first (resting) direction while all remaining fraction n_1' is shifted to the second direction subspace $(\Phi, 0, 0, 0)^T \to (n_1\Phi, 0, n_1'\Phi, 0)^T$. Note that n_1 is the normalized correct amount of the first subspace while the second subspace now contains all remaining information, so for the second and third velocity direction. This remaining information still needs to be further distributed. Now by a second RY operation, an angle θ_2 is determined to distribute between the second and third subspace, according to the LBM weights: $(n_1\Phi, 0, n_1'\Phi, 0)^T \to (n_1\Phi, 0, n_2\Phi, n_3\Phi)^T$. This results in two rotation operations with angles

$$\theta_1 = 2\arccos\underbrace{\left(\frac{k_1}{\sqrt{k_1^2 + k_2^2 + k_3^2}}\right)}_{\equiv n_1}, \quad \theta_2 = 2\arccos\underbrace{\left(\frac{k_2}{\sqrt{k_2^2 + k_3^2}}\right)}_{\equiv n_2}. \tag{24}$$

For a 1D stencil, we shift all the information sequentially through each of the subspaces while keeping a fraction according the LBM weights in the corresponding subspace $(\Phi, 0, \ldots, 0)^T \to (n_1\Phi, n'\Phi, \ldots, 0)^T \to \ldots \to (n_1\Phi, n_2\Phi, \ldots, n_Q\Phi)^T$, where each operation is performed with a general angle for each of the rotation gates of

$$\theta_i = 2\arccos\left(\frac{k_i}{\sqrt{\sum_{j=i}^{Q} k_j^2}}\right). \tag{25}$$

For multiple spacial dimensions, the sequential distribution of the signal through the velocity subspaces may have to be adjusted. If the signal may have to be kept for multiple directions k_j in a certain subspace (as will be necessary in our D2Q9 distribution procedure), the value of k_i has to be replaced by the Euclidean norm of all the directions that we want to keep in the subspace:

$$\theta_i = 2\arccos\left(\frac{\sqrt{\sum_{\text{keep}} k_{\text{keep}}^2}}{\sqrt{\sum_{\text{inv}} k_{\text{inv}}^2}}\right), \tag{26}$$

where k_{keep} are the k_i of the directions that (temporarily) remain in the direction subspace and k_{inv} are all the subspaces involved in the operation, so those which are shifted to another subspace and those which will remain in the respective subspace. This defines the normalized k_i factor n_i used as the argument of the $\arccos(\cdot)$ by

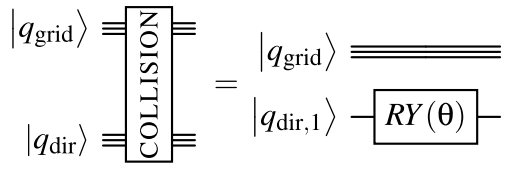
$$n_i \equiv \frac{\sqrt{\sum_{\text{keep}} k_{\text{keep}}^2}}{\sqrt{\sum_{\text{inv}} k_{\text{inv}}^2}}. \tag{27}$$

In this work, the D1Q2, D1Q3 and D2Q9 stencils are used. The details of the quantum circuit of the collision block from Fig. 2 is shown in Figs. 4(a) and (b) for the D1Q2 and the D1Q3 scheme, respectively, and for the D2Q9 scheme it is shown in Fig. 5. A spatially constant diffusion and flow velocity is assumed, so there is no dependence on the grid qubits. For a spatially varying diffusion, additional RY gates with controls on the grid qubits need to be used.

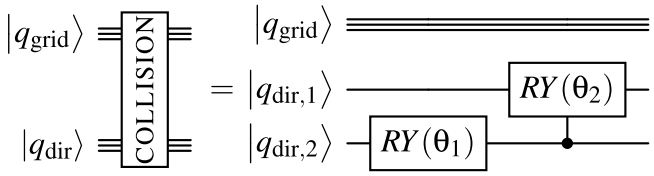
To represent the two velocity directions in the D1Q2 scheme, we use one direction qubit in the $|q_{\text{dir}}\rangle$ register to span the state vector in Eq. (11) of length $2N$, where we already use $\log_2(N)$ qubits for the representation of the N grid points.

For the D1Q3 scheme, we need a state vector in Eq. (11) of length $3N$, so we need an second qubit in the direction register, as explained in Section 2.3. This will span our state space to $4N$.

For the D2Q9 scheme, it is not feasible to distribute the velocity directions in a sequential way, always shifting all the signal to the next direction subspace. Specifically, for the D2Q9 scheme used in this work, first the amplitude from the resting subspace will be distributed along the x-direction subspaces as for D1Q3, but in a second series of steps, the direction subspaces superposing the x-direction (including the resting direction) with up and down directions will get their corresponding signal. The details of the order on how the directions are distributed to their corresponding subspace is shown in the appendix in Section A.2 in Table A.5. For the distribution procedure, a set of eight $RY(\theta)$-gate operations are used and $N_Q = 4$ direction qubits generating 16 subspaces are used to accommodate for the $Q = 9$ velocity directions. To perform the operations only on the desired subspace, several controls are set to the RY-gates, which are shown in the collision circuit in Fig. 5.



(a) D1Q2 collision block circuit



(b) D1Q3 collision block circuit

Fig. 4. Quantum circuit for the collision step in Fig. 2 for the D1Q2 scheme (Fig. 4(a)) and a the D1Q3 scheme (Fig. 4(b)) for spatially constant diffusion and flow velocity. For spatially varying diffusion or flow velocities, multiple RY gates with controls on the grid qubits $|q_{\text{grid}}\rangle$ need to be used.

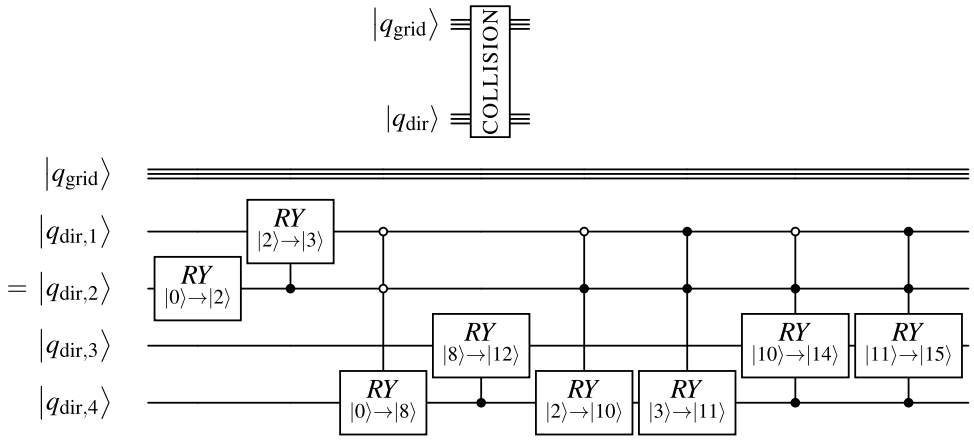


Fig. 5. Quantum circuit for the collision step in Fig. 2 for the D2Q9 scheme for spatially constant diffusion and flow velocity. Each RY-gate distributes the velocity distribution functions into their corresponding velocity direction subspace according to the procedure in Table A.5 in Section A.2 of the appendix. The argument θ for each RY-gate has to be chosen according to Eq. (26). For additionally spatially varying diffusion or flow velocities, multiple RY gates with controls on the grid qubits $|q_{\text{grid}}\rangle$ need to be used.

2.4.2. *The streaming step*

The streaming for the stencils used in this work as indicated in the circuit in Fig. 2 are basic periodic right (→) and left (←) shift operations in 1D and additional up (↑) and down (↓) operations including diagonal superpositions of them, respectively, in 2D. This is done by a binary $+1$ and -1 operation with binary integer overflow, conditioned on the direction qubits. The quantum circuit for the binary shift operations is adapted from Todorova et al. [6] as presented in Figs. 6(a) and (b), where instead of changing the condition state of the control qubits for -1 compared to -1, we instead reverse the order of the gates of $+1$ to represent the -1 operation.

To perform the correct streaming direction in the corresponding direction subspace, the $+1$ and -1 streaming operations are conditioned on the direction qubits, which depends on the velocity set chosen by the QLBM stencil. The conditions for our choice of direction subspaces as described in Section 2.3 are shown in Fig. 7, so every gate of the $+1$ and -1 operation in Fig. 6 has to have additional controls on the direction qubits. The controls are chosen in such a way that they shift the correct subspace of the state, which for the D1Q2 scheme is chosen to $(f_1, f_2)^T = (f_\rightarrow, f_\leftarrow)^T$ and for the D1Q3 is chosen to be $(f_1, 0, f_2, f_3)^T = (f_{\text{rest}}, 0, f_\rightarrow, f_\leftarrow)^T$ (cf. Tables 1 and 2).

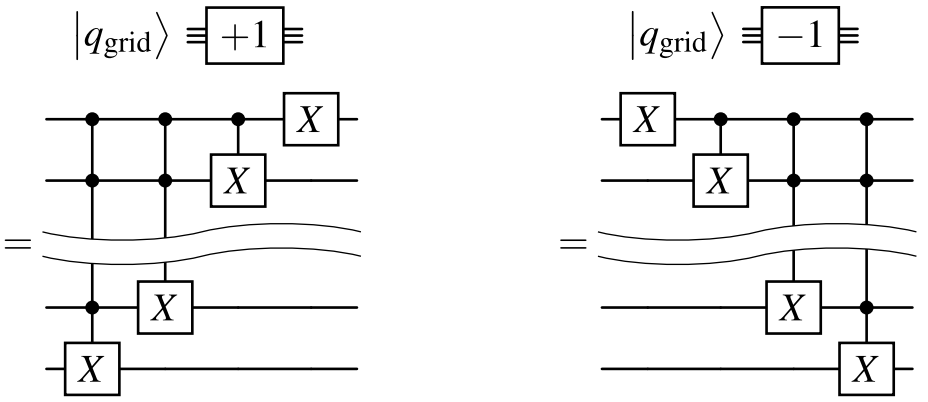
For the D2Q9 scheme, additional up and down streaming operations are performed on y-direction grid qubits with corresponding control states on the direction qubits to select the correct subspace like arranged in Table 3. The diagonal streaming directions are performed by sequentially performing a left or right with an up or down operations on the same direction subspace, so with similar control states on the direction register. The quantum circuits for the streaming of the directions in D2Q9 are shown in Fig. 8.

2.4.3. *Updating macroscopic variables*

The summation of the distribution functions in the different subspaces is basically done by reversing the steps of distributing the distribution functions along the subspaces as is done in the collision steps, so by collecting instead of distributing subspaces. Assuming a total statevector of dimension $2N$, which can be subdivided into two distributions of dimension N, such that $|\Psi\rangle = (\Psi_1^1, \ldots, \Psi_1^N, \Psi_2^1, \ldots, \Psi_2^N)^T$, this can be represented by the two distributions $|\Psi_1\rangle, |\Psi_2\rangle$ for different states of the last, the ancilla, qubits:

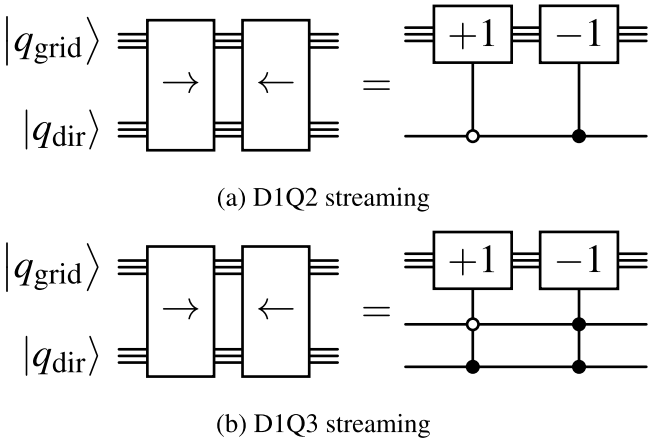
$$|\Psi\rangle = |0\rangle \otimes |\Psi_1\rangle + |1\rangle \otimes |\Psi_2\rangle \equiv (\Psi_1, \Psi_2)^T. \tag{28}$$

The Hadamard transformation H can distribute from a distribution in one subspace to another subspace with zero probability $(\Phi, 0)^T$ to two equal parts in both subspaces $(\Phi_{1/2}, \Phi_{1/2})^T$, while having to conserve the norm of the vector. The Hadamard transformation operation H is its own inverse, so while distributing from one subspace to two subspaces, it also reverses the operation and collects information back into the first subspace $(\Phi, 0)^T \overset{H}{\longleftrightarrow} (\Phi_{1/2}, \Phi_{1/2})^T$ when applied again. The Hadamard



(a) Binary +1 operation with overflow (b) Binary −1 operation with overflow

Fig. 6. Binary +1 and −1 operations with overflow as the basis of the right, left, up and down operations in the QLBM circuit in Fig. 2.



(a) D1Q2 streaming

(b) D1Q3 streaming

Fig. 7. Arrangement of the controls on the direction qubits for the streaming operations of the D1Q2 scheme (7)(a) and D1Q3 scheme (7)(b).

transformation calculates the equally weighted sum in the first subspace, while at the same time, the difference of the distributions is kept in the second subspace

$$H|a\rangle \otimes |\Psi\rangle = H|0\rangle \otimes |\Psi_1\rangle + H|1\rangle \otimes |\Psi_2\rangle$$
$$= \sqrt{\frac{1}{2}}\left(|0\rangle \otimes |\Psi_1 + \Psi_2\rangle + |1\rangle \otimes |\Psi_1 - \Psi_2\rangle\right), \tag{29}$$

where we notice, that we calculate the sum in the first subsection up to a normalization factor of $\sqrt{1/2}$, which will account for another loss of signal, as also further taken into account in Section 2.4.5.

The Hadamard transformation as used here calculates the equally weighted sum and difference up to a normalization of the two distributions. The same holds in general for the RY gate with corresponding choice if θ, which we understand as mixing in between the subspaces as has been explained in Section 2.4.1. Note that in order to maintain the sum and the difference in the corresponding subspaces $|0\rangle$ and $|1\rangle$, we need to choose θ negative, such that the cosine remains the same but the sine flips its sign and the difference shifts to the bottom line in the RY matrix

$$RY(\theta) \equiv \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} = \begin{pmatrix} \cos(-\theta/2) & \sin(-\theta/2) \\ -\sin(-\theta/2) & \cos(-\theta/2) \end{pmatrix}. \tag{30}$$

The result of the difference state is of no further interest. With choosing different angles of θ, the sum can be calculated with different weights. It is important to avoid including zero subspaces that do not contain any distribution function because this would lead to a significant additional decay of the solution. For the sequential summation of different subspaces we have to account for different weightings in each summation step, due to the general norm preservation of a quantum state. The weighted summation of two subspaces $|\Psi_1\rangle$ and $|\Psi_2\rangle$ using the RY-gates results in

$$RY(\theta)|a\rangle \otimes |\Psi\rangle = |0\rangle \otimes |\cos(-\theta/2)\Psi_1 + \sin(-\theta/2)\Psi_2\rangle + |1\rangle \otimes |\cos(-\theta/2)\Psi_2 - \sin(-\theta/2)\Psi_1\rangle. \tag{31}$$

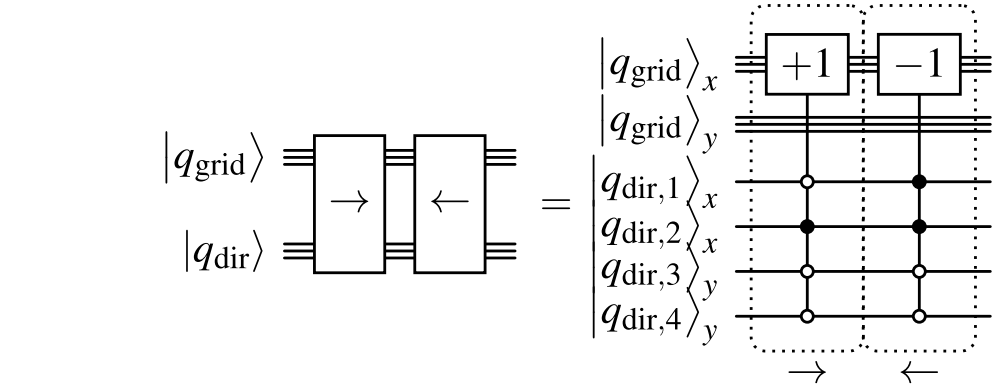
For the **D1Q2** scheme, one equally weighted sum is needed. This is done with the Hadamard gate as demonstrated in Eq. (29). The circuit is shown in Fig. 9(a).

For the **D1Q3** scheme, at first, only the third and fourth subspaces, so $f_{\rightarrow}$ and $f_{\leftarrow}$ in $\Psi = (f_{\text{rest}}, 0, f_{\rightarrow}, f_{\leftarrow})^T$, are summed equally weighted by a Hadamard gate, which leaves its sum in the third subspace. Afterwards, a RY gate is used to sum the first and third subspace where the sum results in the first subspace. We need the final sum to be an equally weighted sum of the first, third and fourth subspace, so $f_{\text{rest}} + f_{\rightarrow} + f_{\leftarrow}$, meaning a weighting of the statevector with $(1, 0, 1, 1)^T$. Due to normalization, we need the weights to be $(\sqrt{1/3},\ 0, \sqrt{1/3}, \sqrt{1/3})$, which is fine as long as the weights remain equal. To apply the gates as summation operations on these isolated subspaces, controls need to placed accordingly. The quantum circuit with corresponding controls are shown in Fig. 9(b). For these two controlled operations, the total summation operation on the two direction subspaces is:

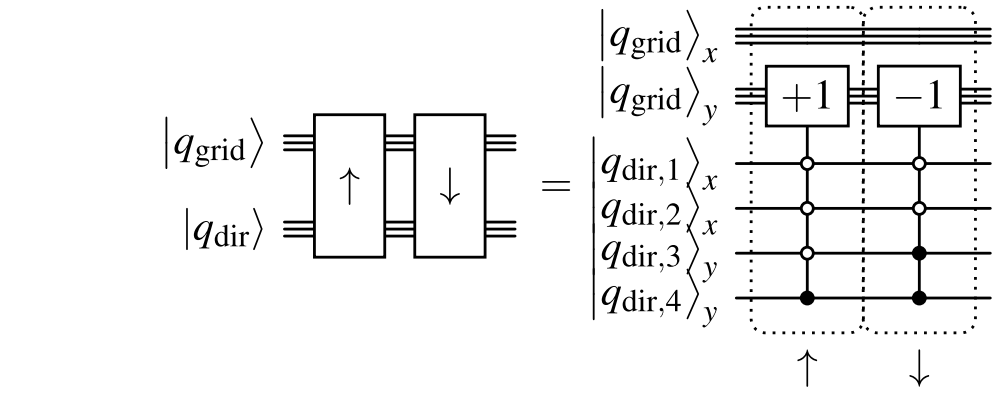
$$RY(\theta)\Big|_{|q_{\text{dir},1}\rangle=|0\rangle} H\Big|_{|q_{\text{dir},2}\rangle=|1\rangle} \Psi$$
$$= \begin{pmatrix} \cos\left(-\frac{\theta}{2}\right) & 0 & \sin\left(-\frac{\theta}{2}\right) & 0 \\ 0 & 0 & 0 & 0 \\ -\sin\left(-\frac{\theta}{2}\right) & 0 & \cos\left(-\frac{\theta}{2}\right) & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{1/2} & \sqrt{1/2} \\ 0 & 0 & \sqrt{1/2} & -\sqrt{1/2} \end{pmatrix} \begin{pmatrix} f_{\text{rest}} \\ 0 \\ f_{\rightarrow} \\ f_{\leftarrow} \end{pmatrix} \tag{32}$$
$$= \begin{pmatrix} \cos\left(-\frac{\theta}{2}\right) & 0 & \sin\left(-\frac{\theta}{2}\right)\sqrt{\frac{1}{2}} & \sin\left(-\frac{\theta}{2}\right)\sqrt{\frac{1}{2}} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} f_{\text{rest}} \\ 0 \\ f_{\rightarrow} \\ f_{\leftarrow} \end{pmatrix},$$

which will fulfill our demand of an equally weighted summation $(\sqrt{1/3}f_{\text{rest}} + 0 + \sqrt{1/3}f_{\rightarrow} + \sqrt{1/3}f_{\leftarrow}, \cdot, \cdot, \cdot)^T$ by the condition:

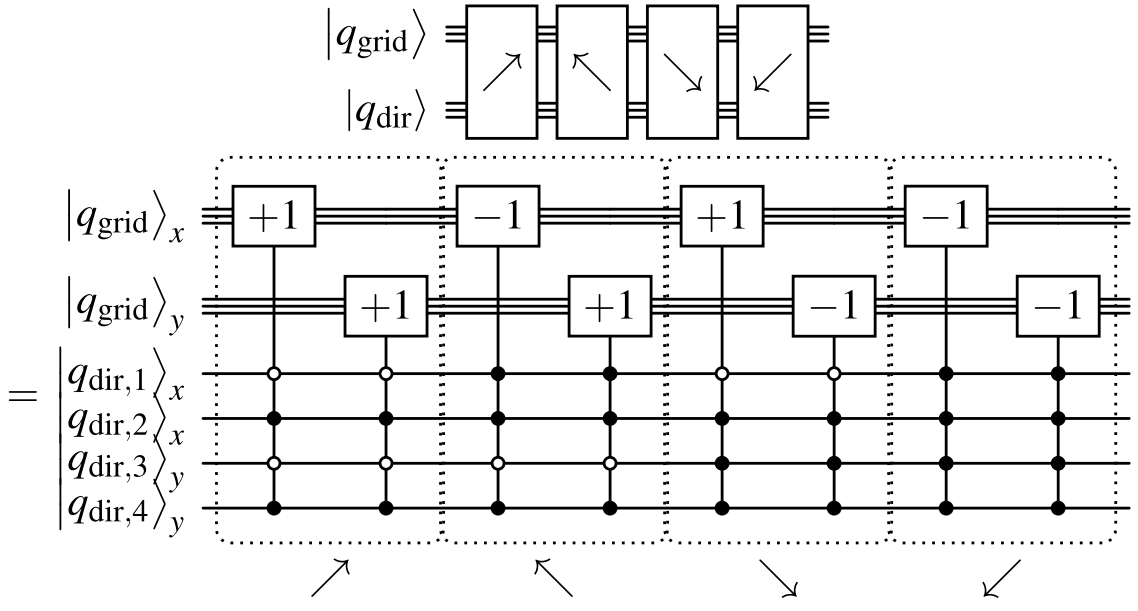
$$\cos\left(-\frac{\theta}{2}\right) \overset{!}{=} \sqrt{\frac{1}{3}} \quad \wedge \quad \sin\left(-\frac{\theta}{2}\right)\sqrt{\frac{1}{2}} \overset{!}{=} \sqrt{\frac{1}{3}} \tag{33}$$



(a) D2Q9 right and left streaming

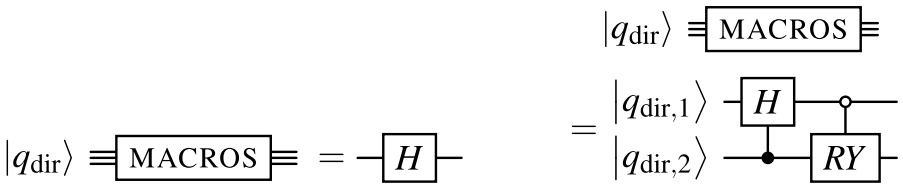


(b) D2Q9 up and down streaming



(c) D2Q9 diagonal streaming

Fig. 8. Implementation of the D2Q9 streaming operations in different directions as a combination of +1 and −1 operations from Fig. 6 with corresponding controls on the direction qubits as in Table 3.



(a) Update for macroscopic variables for D1Q2 (b) Update for macroscopic variables for D1Q3

Fig. 9. Update of the macroscopic variables as the sum of all distributions into the first subspace for D1Q2 (Fig. 9(a)) and D1Q3 (Fig. 9(b)).

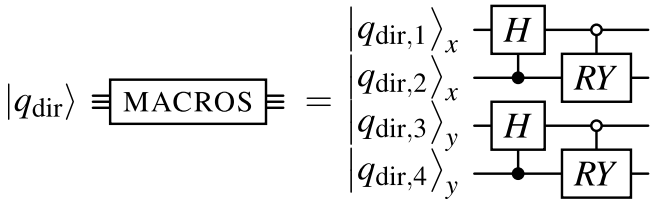


Fig. 10. Update of the macroscopic variables as the sum of all distributions into the first subspace for the D2Q9 scheme using θ from Eq. (34) for the RY gates, following the order schematically shown in Table A.7.

$$\Longleftrightarrow \theta = -2\arccos\left(\sqrt{\frac{1}{3}}\right) = -2\arcsin\left(\sqrt{\frac{2}{3}}\right). \tag{34}$$

So for the calculation of the D1Q3 macroscopic quantities with its result in the first subspace, a Hadamard gate on the first direction qubit with a control on the second direction qubit being in state $|1\rangle$ and a RY rotation gate with $\theta = -\arccos(\sqrt{1/3})$ on the second direction qubit with control on the first direction qubit being in state $|0\rangle$ is used, as shown in Fig. 9(b).

To calculate the macroscopic values in the **D2Q9** scheme, the summation is done in a similar way with the same summation weights as in the D1Q3 scheme, which is shown in the circuit in Fig. 10 with the summation order shown in Table A.7 in the appendix. For the D2Q9 scheme, the summation is done in x- and y-direction, respectively, by the same operations as in the D1Q3 summation step. First, the x-direction summation operations perform the summations of the directions $|\text{rest}\rangle, |\rightarrow\rangle, |\leftarrow\rangle$ into $|\text{rest}\rangle$ directions, the $|\uparrow\rangle, |\nwarrow\rangle, |\nearrow\rangle$ direction into $|\uparrow\rangle$ direction and $|\downarrow\rangle, |\swarrow\rangle, |\searrow\rangle$ directions into $|\downarrow\rangle$ direction with the Hadamard gate H and rotation gate RY as in D1Q3. After that, the final summations of $|\text{rest}\rangle, |\uparrow\rangle, |\downarrow\rangle$ into$|\text{rest}\rangle$ are done similarly but on the y-qubits.

2.4.4. Re-prepare state for next time step

In order to create a fully quantum algorithm for multiple quantum lattice Boltzmann steps without reinitialization, the state vector has to be re-prepared to the state that contains the scalar grid information only in the first direction subspace while all other direction subspaces have to have zero state amplitudes, i.e. $(f_1,\ f_2,\ \ldots,\ f_{N_Q})^T = (\Phi,\ 0,\ \ldots,\ 0)^T$ (cf. Section 2.3). As we see in Eq. (29), with the summation of subspaces also comes a subtraction result. So after the calculation of the macroscopic quantities, there is the sum in the first subspace but generally also left over information in all other used direction subspaces. But in order to make the algorithm work for multiple time steps with collision, streaming and calculation of the macroscopic quantities, the state requires to have only the grid results in the first velocity direction subspace and all other velocity direction subspaces need to have zero probability amplitude, as stated in Eq. (11). So a way is needed to set these probabilities to zero, without destroying the system quantum state.

The idea is to introduce additional qubits as part of the time qubit register, which is shown in Fig. 2. The first additional time qubit doubles the number of direction subspaces, so for N_Q direction subspaces, the statevector expands from $(f_1,\ f_2,\ \ldots,\ f_{N_Q})^T$ to $(f_1,\ f_2,\ \ldots,\ f_{N_Q},\ 0,\ \ldots,\ 0)^T$. This statevector now contains N_Q additional subspaces with states of probability zero, because the additional time qubit, as long as not operated on, remains in state $|0\rangle$, so in the state of the first N_Q direction subspaces and never takes $|1\rangle$ in the second N_Q direction subspaces. The key part is to shift all $2N_Q$ periodically and switching the $N_Q + 1$ subspace with the first subspace again:

$$\text{RE-PREP:} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N_Q} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N_Q} \end{pmatrix} \longrightarrow \begin{pmatrix} f_1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ f_2 \\ \vdots \\ f_{N_Q} \end{pmatrix}, \tag{35}$$

where after the collision, streaming and macroscopic calculation steps, the first subspace contains the gird $f_1 \sim \Phi$ up to a known normalization factor. The norm of f_1 will decrease in every time step due to the left over results of the subtraction in the other subspaces. This decay is further discussed in Section 2.4.5. Since the collision, streaming and macroscopic calculation steps only act within a group of every N_Q subspaces, the state after the re-preparation at the end of Eq. (35) can be used for the next time step. For multiple time steps, further time qubits are added, where each additional time qubits doubles the number of zero-probability-subspaces. Therefore, the number of time qubits needed scales logarithmically with the number of simulated time steps T, so $\#q_{\text{time}} \sim \log(T)$, meaning that $\#q_{\text{time}}$ time qubits can simulate up to $T = 2^{\#q_{\text{time}}}$ time steps. The periodic shifting $+N_Q$ of the entries of state vector can be realized in a similar way as the streaming operation. The quantum circuit is shown in Fig. 11.

For some specific lattice Boltzmann stencils, the RE-PREP circuit as shown in Fig. 11 can be simplified. For the D1Q2 and D1Q3 stencil a simplification of the RE-PREP circuit is shown in the appendix in Section A.4.

2.4.5. Decay of quantum state amplitude of solution per time step

The updated grid values for the next time step in the first subspace $f_1^{t+1} = \Phi^{t+1}$ undergoes a certain decay of its coefficient amplitude at two points in the algorithm: the collision operation and the calculation of the macroscopic variables. The first decay comes from the collision operation, that distributes the full grid in the first subspace f_1 to the velocity distributions f_i. In this step, only the normalized fraction n_1 of Φ is kept in the first subspace and the remaining signal n' is moved

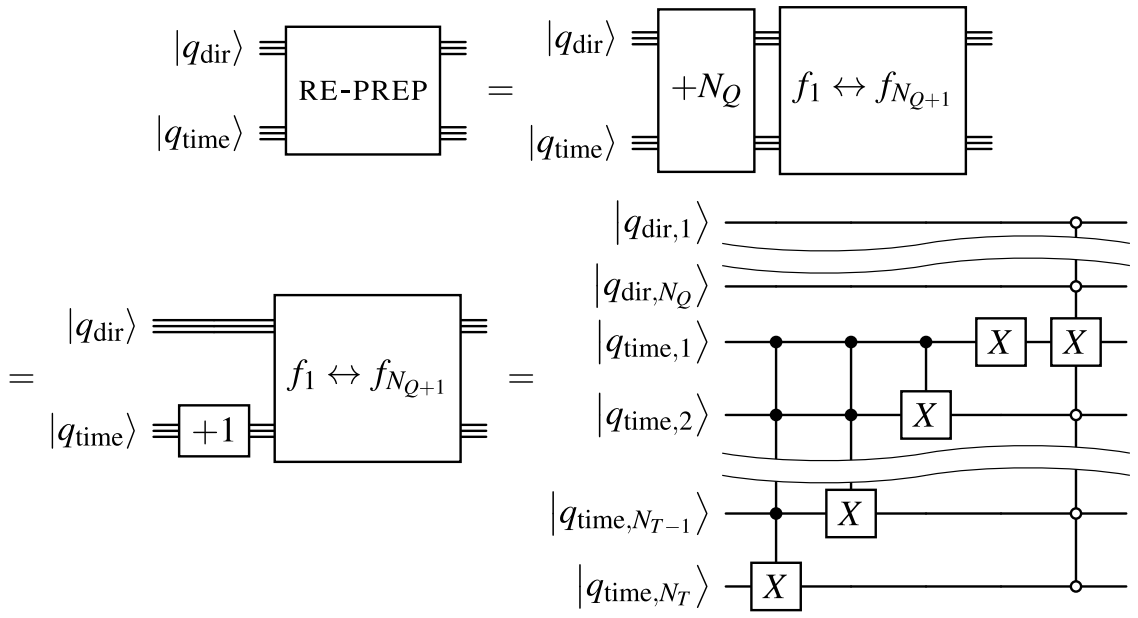


Fig. 11. Quantum circuit of the RE-PREP step in Fig. 2 to prepare the state vector for the next time step in order to allow a fully quantum algorithm for all simulation time steps without the need of reinitialization or mid-circuit measurements.

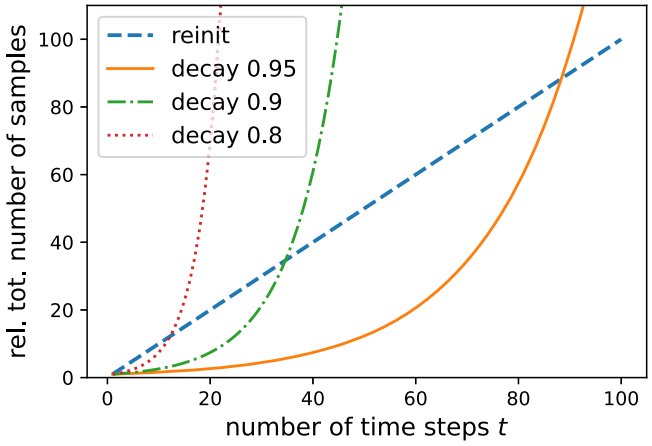


Fig. 12. Comparison of the total number of shots over time for reinitialization methods (dashed line) and our method for different decay factors.

to subspace f_2. Instead of keeping the full amount k_1 in subspace f_1, as would be dictated by the non-unitary collision operation in Eq. (5), only the normalized amount of n_1 is kept. This normalization leads to the first signal decay in f_1 as determined in Eq. (25)

$$\gamma_{\text{coll},f_1} = \frac{1}{\sqrt{\sum_{i=1}^{Q} k_j^2}}. \tag{36}$$

The second decay arises from the last summation step when calculating the macroscopic variables. As shown in the Section 2.4.3, a summation of subspaces comes with a subtraction as well. While there is the sum of two subspaces in one subspace, there is the subtraction result in the other subspace. The resulting subtraction is some left over, unusable signal in the other subspaces. The last summation step is weighted with $\sqrt{1/Q}$ to take into account previous sequentially summations of velocity subspaces. So the decay due to the summation of the last subspaces that includes the f_1 subspace is

$$\gamma_{\text{macro},f_1} = \sqrt{\frac{1}{Q}}. \tag{37}$$

By determining only the decay of the macroscopic value Φ, which is calculated in subspace f_1, only the collision decay and macroscopic variable calculation decay for f_1 need to be taken into account. Further decays due to the collision and summation in further velocity subspaces $f_{i>1}$ do not change the decay in the first subspace. Only when f_1 is included in the operations f_1 decays further. This results in a total decay of

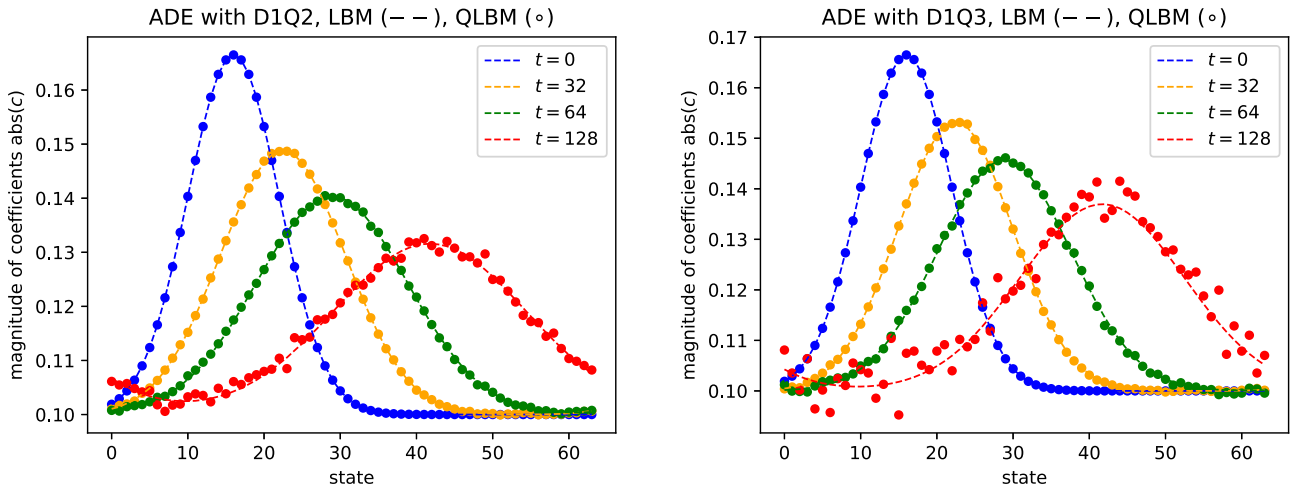
$$\gamma_{\text{tot},f_1} = \frac{1}{\sqrt{\sum_{i=1}^{Q} k_j^2}} \sqrt{\frac{1}{Q}}, \tag{38}$$

which is the signal loss of the complex amplitude coefficients of the macroscopic grid variables in the first velocity subspace per time step. For the decay of probability amplitude, the square of the decay value $\gamma^2_{\text{tot},f_1}$ determines the signal loss. In order to reconstruct the grid after T time steps, that is encoded in the complex amplitude coefficients, the magnitude of the complex amplitude coefficients need to be multiplied by the decay factor for T time steps $\gamma^T_{\text{tot},f_1}$.

2.4.6. Comparison to methods that use reinitialization

Our algorithm avoids the necessity to measure and reinitialize the quantum state after every single time step by adding the RE-PREP block. For a single time step, our method and methods that require reinitialization are similar, i.e. both initialize and perform collisions, streaming and the macroscopic step. Performing many time steps, methods that require reinitialization have to contain the INITIALIZATION block in every time step while our method initializes only once and contains the RE-PREP block in every time step instead. When considering many time steps, the computational cost of the single INITIALIZATION block in our method becomes negligible. Therefore, the difference between a method with reinitialization and our method without reinitialization are the REINITIALIZATION block and the RE-PREP block. Comparing these two blocks to compare methods with and without reinitialization can not be done in general, since the initialization highly depends on the flow field to encode. However, the total number of samplings required to resolve the flow to a certain level after T time steps can be compared for both methods.

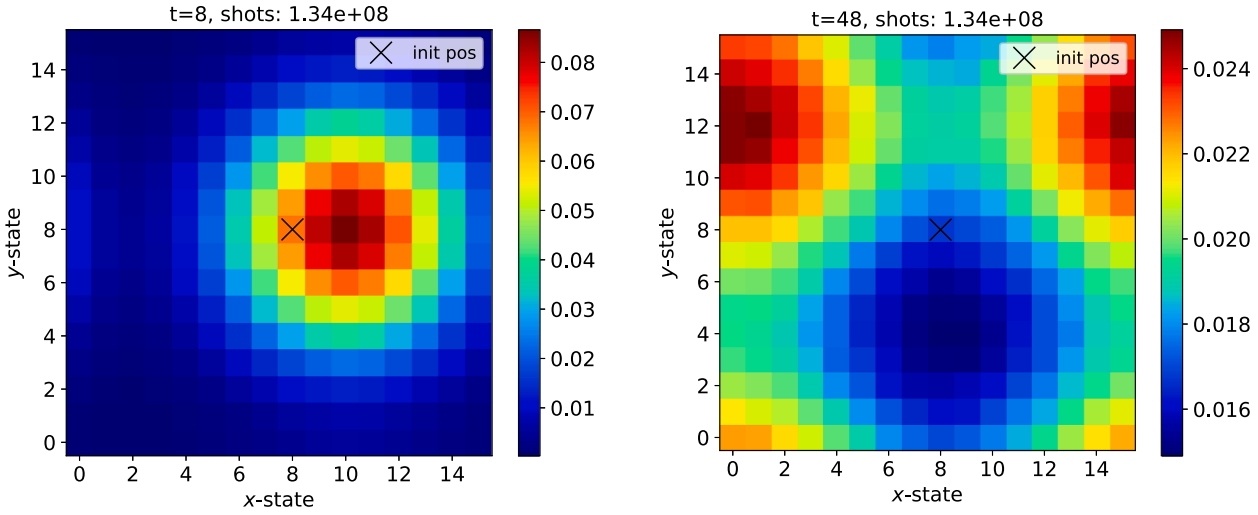
Assuming a requirement of N_{shots} shots for a desired resolution, a reinitialization method needs to resolve the entire flow field with N_{shots} in every time step. So for T time steps, the total number of shots scales



(a) 1D ADE with the D1Q2 scheme

(b) 1D ADE with the D1Q3 scheme

Fig. 13. Fully Quantum Lattice Boltzmann method without reinitialization for the 1D linear advection-diffusion equation solved with a D1Q2 scheme in Fig. 13(a) and a D1Q3 scheme in Fig. 13(b).



(a) D2Q9. The flow velocity is $\mathbf{u} = (1/4, 0)^T$ in lattice units for $t = 8$ time steps.

(b) D2Q9. The flow velocity is $\mathbf{u} = (1/6, 1/12)^T$ in lattice units for $t = 48$ time steps.

Fig. 14. Fully Quantum Lattice Boltzmann method without reinitialization for the 2D linear advection-diffusion equation solved with a D2Q9 scheme with periodic boundary conditions and a diffusion constant of $D = 1/3$. The cross (×) indicates the initial position of the gaussian with a standard deviation of $\sigma = 2.0$. Two different times and velocities are simulated in Figs. 14(a) and (b).

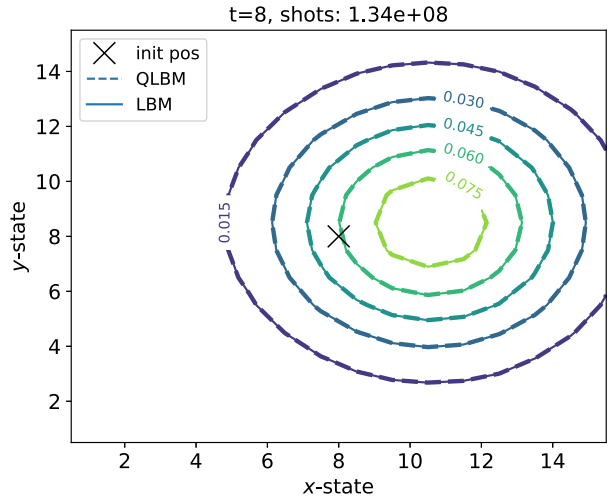
linearly by $T \cdot N_{\text{shots}}$. Our methods decays by the decay factor γ_{tot,f_1} in every time step (cf. Eq. (38), Section 2.4.5), so to compensate the decay, the number of samples need to be $N_{\text{shots}}/\gamma_{\text{tot},f_1}$. For T time steps, this means a total number of $N_{\text{shots}}/\gamma_{\text{tot},f_1}^T$ is required. Removing the arbitrary factor N_{shots}, the total relative number of shots required for a reinitialization method and our method is shown in Fig. 12 for different decay values γ_{tot,f_1}.

3. Verification

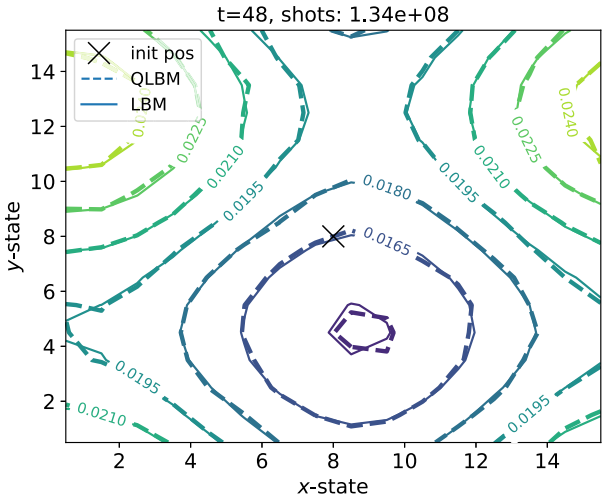
To verify our proposed algorithm, we model the advection-diffusion process of a gaussian distributed concentration in one and two dimensions for multiple time steps without reinitialization. We use our quantum lattice Boltzmann method (QLBM) with a D1Q2, D1Q3 and D2Q9 stencil and compare it to classical lattice Boltzmann (LBM) results. All simulations are performed such that each velocity distribution function fully relaxes to its local equilibrium within one time step, i.e. $\Delta t = \Delta\tau = 1$. Further, a spacial discretization of $\Delta x = 1$ is used. With different sets of lattice Boltzmann weights w_i different diffusion constants D are modeled ensuring conservation of moment equations at least up to order four (cf. Section 2.2.1). For the simulation of our quantum algorithm, the shot method of the *Qiskit* package [16] from *IBM* is used. The number of quits and the number of gates for the *Qiskit* simulations for the different stencils, i.e. D1Q2, D1Q3 and D2Q9, are listed in Table 4.

3.1. 1D Advection-Diffusion equation

The simulation of a one-dimensional gaussian hill following the ADE is done with a D1Q2 stencil and D1Q3. For both 1D test cases, a uniform velocity of $u = 0.2$ and periodic boundary conditions are used. The

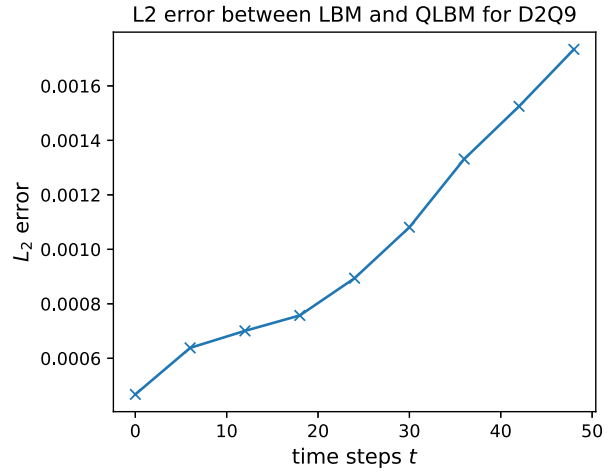


(a) D2Q9. The flow velocity is $\mathbf{u} = (1/4, 0)^T$ in lattice units for $t = 8$ time steps.

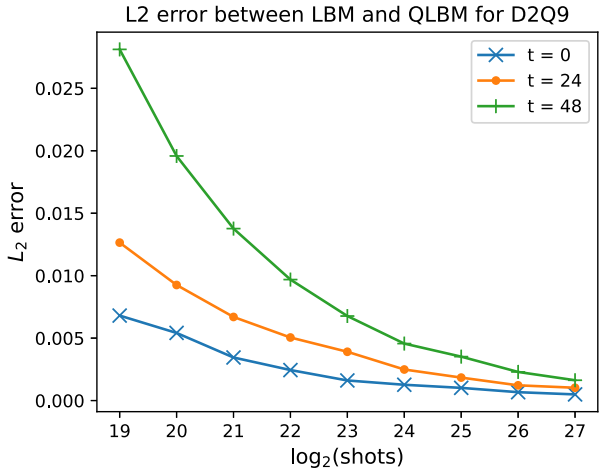


(b) D2Q9. The flow velocity is $\mathbf{u} = (1/6, 1/12)^T$ in lattice units for $t = 48$ time steps.

Fig. 15. Contour plots of the 2D linear advection-diffusion simulation showing results of the fully Quantum Lattice Boltzmann method without reinitialization from Figs. 15(a) and (b) (dashed lines) and the results from the classical Lattice Boltzmann implementation (solid lines).



(a) Evolution of the L_2-error deviation of QLBM to LBM method over time for the QLBM D2Q9 test case in fig. 14b.



(b) Convergence of the QLBM D2Q9 test case towards LBM for more shots at different simulation times.

Fig. 16. Comparison of QLBM and LBM method for a D2Q9 test case in Fig. 14(b) with an initial gaussian distribution of standard deviation of size $\sigma = 2.0$ at a diffusion of $D = 1/3$ for a velocity of $\mathbf{u} = (1/6, 1/12)^T$.

Table 4
Number of qubits (grid, direction, time) and gates (X,) used for the not-transpiled simulation of the results shown for D1Q2 in Fig. 13(a), D1Q3 in Fig. 13(b) and D2Q9 in Figs. 14(a) and (b).

stencil	Figures	qubits	X	CX	MCX	H	CH	RY	CRY	$MCRY$
D1Q2	13(a)	14 (6, 1, 7)	0	3	16	1	0	1	0	0
D1Q3	13(b)	15 (6, 2, 7)	0	1	18	0	1	1	2	0
D2Q9	14(a)	15 (4+4, 4, 3)	1	1	50	0	2	1	4	5
D2Q9	14(b)	18 (4+4, 4, 6)	1	1	53	0	2	1	4	5

gaussian hills are initialized at position $x_0 = 16$ with a standard deviation of $\sigma = 6.0$ and a global offset of $o = 0.1$. The grid is discretized on $N = 64$ grid points by $\#q_{\text{grid}} = 6$ grid qubits. For a maximum simulation time of $t = 128$, a number of $\#q_{\text{time}} = 7$ qubits in the time qubit register are used. The D1Q2 stencil requires $\#q_{\text{dir}} = 1$ direction qubit for its two velocity directions, whereas the D1Q3 stencil requires $\#q_{\text{dir}} = 2$ direction qubits for its three velocity directions. The simulations are performed with $3.4 \cdot 10^7$ shots in total. The results are shown in Figs. 13(a) and (b).

For the D1Q2 stencil, the standard LBM weights are used, i.e. [1/2, 1/2], which results in a squared speed of sound of $c_s^2 = 1$ and therefore a diffusion of $D = 1/2$ is modeled. This results in a decay per time step to $\gamma = 0.98$, which reduces the probability per time step to $\gamma^2 = 0.96$.

The simulation with the D1Q3 stencil uses the non-standard weight set of [1/3, 1/3, 1/3] which results in a squared speed of sound of $c_s^2 = 2/3$ and therefore a simulated diffusion of $D = 1/3$. This setting results in a decay per time step to $\gamma = 0.97$, so a probability reduction to $\gamma^2 = 0.94$ per time step.

The results show that our QLBM is capable of reproducing the LBM results overall very accurately. For more time steps, the probability of measuring the correct states becomes less likely, which is due to the decay of the probability function in our algorithm. This results in an increasing noise of the solution for larger time steps. Performing the simulation with more shots in total reduces the noise, as will be discussed in more detail for the 2D test case in Section 3.2.

3.2. 2D Advection-Diffusion equation

To verify our algorithm for two-dimensions, we again propagate a gaussian hill, now in 2D, following the advection-diffusion equation using the D2Q9 stencil with periodic boundary conditions. The results for two different advection velocities are shown in Figs. 14(a) and (b). Contour plots of the results in Figs. 14(a) and (b) with comparison to the classical Lattice Boltzmann solutions are shown in Figs. 15(a) and (b).

The gaussians are discretized on 16×16 nodes using $\#q_{\text{grid}} = 4 + 4$ grid qubits for the x and y directions. The concentrations have a standard deviation of $\sigma = 2.0$ and no global offset. The D2Q9 stencil requires $Q = 9$ velocity directions, so $N_Q = 16$ velocity direction subspaces need to be generated, which is done using $\#q_{\text{dir}} = 4$ velocity direction qubits. With $\#q_{\text{time}} = 6$ time qubits, up to $T = 48$ time steps are simulated. The flow field is sampled at the end of the simulation with $1.3 \cdot 10^8$ shots.

Since the decay is smaller for smaller difference of the values of the weights, we choose a weight set of equal weights. For the test cases, a set of $w_i = 1/9\ \forall i \in [1, 9]$ is used which results in the minimal decay of our solution. This set of weights simulates a squared speed of sound of $c_s^2 = 2/3$ and thus a diffusion of $D = 1/3$ is simulated. The decay has a dependence on the advection velocity since it is part of the k_i factors in the decay in Eq. (2.4.5). So for the test case in Fig. 14(a), an advection velocity of $\mathbf{u} = (1/4, 0)^T$ is used, resulting in a decay of $\gamma = 0.96$ per time step, so a probability decay to $\gamma = 0.91$ per time step. For the test case in Fig. 14(b), the smaller advection velocity of $\mathbf{u} = (1/6, 1/12)^T$ results in a decay of $\gamma = 0.97$ per time step, so a probability decay to $\gamma = 0.95$ per time step. The results show very good agreement with the expectations, including deviations due to sampling noise. To verify the agreement with the classical LBM solutions, especially regarding expected sampling deviations, a more detailed look comparing QLBM with LBM is done: to quantify the difference the results by our QLBM compared to a classical LBM, we calculate the L_2 error as

$$L_2 = \sqrt{\sum_{i=1}^{N} (\Phi_{\text{QLBM}} - \Phi_{\text{LBM}})^2}. \tag{39}$$

Due to an increasing decay of our solution for more time steps, we expect that for a fixed number of shots, fewer shots resolve our flow field in the correct subspace. Therefore, we expect increase deviation for an increasing number of simulated time steps between QLBM and LBM, since the flow field is effectively resolved with fewer shots. This is confirmed by calculating the L_2 error for more time steps, as is shown in Fig. 16(a). Now this also means, that the error should decrease to zero for a fixed number of time steps with more sampling shots, meaning that the QLBM solution converges towards the LBM solution. This in fact can be verified by Fig. 16(b), which shows the convergence of QLBM solution towards the LBM solution for different simulation time lengths. This means that our QLBM algorithm can approximate the LBM solution arbitrarily close even for longer simulation times if the decay can be compensated with more total simulation shots.

4. Conclusion

In this paper, an extension of the Quantum Lattice Boltzmann Method (QLBM) is proposed and verified, such that multiple time steps can be performed without the need of state measurement or reinitialization in between the time steps. This extension is valid for general lattice Boltzmann velocity stencils and tested on D1Q2, D1Q3 and D2Q9 stencils using the *shot* methods of the *Qiskit* simulation package. The algorithm is proposed and discussed in detail, giving the mathematical description as well as the quantum circuit diagrams. For the extended QLBM algorithm, we discuss our required initialization state, the collision and streaming step, calculation of macroscopic variables and a re-preparation step for the next time step, all as fully quantum algorithm blocks with corresponding quantum circuit gate diagrams.

The extension is tested on a linear advection-diffusion equation (ADE) in one and two dimensions and compared to classical lattice Boltzmann (LBM) reference solutions. We show excellent agreement and a convergence of our QLBM to LBM for any desired accuracy. For very large, highly-resolved grids, a state extraction of the full grid may be infeasible. The main advantage of our algorithm is that there is no need to extract the full flow field at any time. While other algorithms requires measurements and state reinitialization of the full flow field, our algorithm can perform all time steps without any measurements or reinitialization. When only interested in surface integrals or scalar properties, our algorithm would allow to calculate these quantities without ever having to extract the flow field at any time at all. This overcomes the scaling issues of algorithms in the literature that require state extraction where the computational effort, given by the number of shots, scales with the grid resolution. Future work is dedicated to reduce the decay by such techniques like amplitude amplification and to tackle nonlinearities to be able to solve fluid flow equations like the *Burgers* equations. Further, the method is supposed to include the flow around bodies and different boundary conditions than periodic.

4.1. Discussion of our algorithms advantages

Our goal is to investigate possible quantum algorithm approaches and improve these algorithms such that they may be suitable and usable for applications in Aerospace science. Since we are looking for algorithms that can deal with very large grids with extremely high resolution, it is important to find an algorithm where the computational cost scales efficiently with the grid resolution. These computational costs are essentially the number of shots and the number of gates required. An algorithm that requires to sample the fully resolved grid cannot fulfill these requirements. Therefore, we propose a method that can perform the full simulation without the need of state extraction at any time and still obtain the results that we are interested in. These can be mainly reduced quantities like lift, drag, or other body surface properties.

Our algorithm can perform multiple time steps more efficient in terms of the number of shots required than those which require state extraction in between every time step, although this efficiency holds only for a limited number of time steps. In terms of efficiency, one could think of combining our algorithm with state extraction algorithms, but for our purposes, we want to avoid a state extraction for the aforementioned reasons at all and only extract our reduced quantities. How to extract these reduced quantity is completely unclear yet and remains an open question.

A large computational advantage of our algorithm is that we can avoid having to reinitialize a very complex flow field in between the time steps, which generally will require a large number of gates to tune the statevector accordingly. For a fluid flow simulation around a body, we could basically initialize a uniform velocity field which is achieved with very few gates.

While our algorithm can be more efficient in the initialization and number of shots, we do not reduce the gate count compared to

other QLBM algorithms. In fact, our algorithm without state extraction produces a very large circuit depth which may cause coherence time issues. Additionally, the noise induces due to errors of the gate operations significantly limits the number of feasible time steps. This depends highly on the hardware and large improvements are to expect in the upcoming years.

CRediT authorship contribution statement

Aaron Nagel: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization; **Johannes Löwe:** Writing – review & editing, Supervision, Formal analysis, Conceptualization.

Data availability

The data and code cannot be shared due to restrictions within the *ToQuaFlics* project of the *DLR QCI*. However, the content of this paper allows to fully reproduce the algorithm and the results. Data and code may be made available via a license agreement on request.

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.

Acknowledgements

This project was made possible by the DLR Quantum Computing Initiative and the Federal Ministry for Economic Affairs and Climate Action; *qci.dlr.de/projects/toquaflics*. Figures with Quantum circuits are visualized using the *Quantikz* latex package [19].

Appendix A. Appendix

A.1. Pseudocode of simulation blocks

A pseudocode showing the structure of the entire simulation using the corresponding QLBM block from the methods section is shown in Algorithm 1. It shows essentially the structuring of the program corresponding to Fig. 3. Each of the functions representing a QLBM block list the gates of their corresponding quantum circuit figure as explicitly depicted in the respective methods subsection.

A.2. D2Q9 Collision step distribution of velocity direction distribution functions

The distribution of the D2Q9 velocity distribution functions is easier to handle if not distributed directly in a sequential way. The order of the procedure of distributing the velocity distribution function into the corresponding direction subspaces by several RY-gate operations is shown in Table A.5.

A.3. D2Q9 Summation step of velocity direction distribution functions

The order of the summation of the D2Q9 velocity distribution functions is shown in Table A.7 of the corresponding circuit in Fig. 10.

Algorithm 1 Main structure of QLBM simulation program, abbreviate *quantum-circuit* by *qc*, *list of control qubits* by *ctrls* and *list of target qubits* by *targets*.

```
PARAMETER declaration

def collision:                                   ▷ choose for velocity set stencil
    while i in subspaces do
        θ_i ← calculate θ_i from Eq. 26
        qc.append(θ_i, ctrls, targets)
    end while

def streaming:                                   ▷ cf. Figures 6, 7 and 8
    while i in directions do
        stream_direction(qc, ctrls, targets)
    end while

def macros:                                      ▷ cf. Figures 9 and 10
    call_H_or_RY_gates(qc, ctrls, targets)

def re_prep:                                     ▷ cf. Figure11
    streaming_+1(qc, |q_time⟩ )
    bring_back_first_subspace(qc, ctrls, target = |q_time, 1⟩ )

init vector ← flatten initial flow field
init vector ← normalize flattened state
init quantum state ←call Qiskit's initialize(init vector) function

while t ≤ T do                                   ▷ build entire quantum circuit
    collision()
    streaming()
    macros()
    re_prep()
end while
measure
```

A.4. Re-prepare state for D1Q2 and D1Q3

For the D1Q2 and D1Q3 stencil, the RE-PREP circuit shown in Fig. 11 can be simplified. In these stencils, all the velocity subspaces but the first one are located in the second half of all subspaces, so f_1 is in subspace one and all velocity subspaces f_i are in subspace with index $i > \frac{1}{2}N_Q$. This is shown in Eqs. (A.1) and (A.2), where the dashed line indicates the split into half of the velocity direction subspaces and the solid separation line the split to the additional subspaces due to the added time qubits. This results in a RE-PREP step for the D1Q2 scheme of

$$\left(\begin{array}{c} |0\rangle_{\text{time}}\,|0\rangle_{\text{dir}} \\ \hdashline |0\rangle_{\text{time}}\,|1\rangle_{\text{dir}} \\ |1\rangle_{\text{time}}\,|0\rangle_{\text{dir}} \\ |1\rangle_{\text{time}}\,|1\rangle_{\text{dir}} \end{array}\right) : \left(\begin{array}{c} f_1 \\ \hdashline f_2 \\ 0 \\ 0 \end{array}\right) \xrightarrow{\text{D1Q2RE-PREP}} \left(\begin{array}{c} f_1 \\ \hdashline 0 \\ 0 \\ f_2 \end{array}\right) \tag{A.1}$$

and for the D1Q3 scheme of

$$\left(\begin{array}{c} |0\rangle_{\text{time}}\,|00\rangle_{\text{dir}} \\ \hdashline |0\rangle_{\text{time}}\,|01\rangle_{\text{dir}} \\ |0\rangle_{\text{time}}\,|10\rangle_{\text{dir}} \\ |0\rangle_{\text{time}}\,|11\rangle_{\text{dir}} \\ |1\rangle_{\text{time}}\,|00\rangle_{\text{dir}} \\ |1\rangle_{\text{time}}\,|01\rangle_{\text{dir}} \\ |1\rangle_{\text{time}}\,|10\rangle_{\text{dir}} \\ |1\rangle_{\text{time}}\,|11\rangle_{\text{dir}} \end{array}\right) : \left(\begin{array}{c} f_1 \\ \hdashline 0 \\ f_2 \\ f_3 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right) \xrightarrow{\text{D1Q3RE-PREP}} \left(\begin{array}{c} f_1 \\ \hdashline 0 \\ 0 \\ 0 \\ 0 \\ f_2 \\ f_3 \end{array}\right). \tag{A.2}$$

In these arrangements it is sufficient to move only the second half of all velocity subspaces. With this arrangement of the velocity direction subspaces, this re-preparation can be achieved by a $+N_Q$ operation on all subspaces where the most significant qubit is in state $|1\rangle$ or equivalently

Table A.5
Distributing the D2Q9 velocity distribution functions along the velocity direction subspaces. The division of the four direction qubits is done such that the first two qubits form the x-directions and the last two qubits form the y-directions, so $|q_4q_3q_2q_1\rangle_{\text{dir}} = |q_4q_3\rangle_y\,|q_2q_1\rangle_x$. The index of θ contains the state with decimal representation of the binary states. The series of tables continues in Table A.6.

	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	$f_1, \ldots, f_9$	-	-	-
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	-	-	-	-
$\|11\rangle_y$	-	-	-	-
$\xrightarrow{\theta_{\|0\rangle\to\|2\rangle}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1, f_4, f_5	-	$f_2, f_3, f_6, f_7, f_8, f_9$	-
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	-	-	-	-
$\|11\rangle_y$	-	-	-	-
$\xrightarrow{\theta_{\|2\rangle\to\|3\rangle}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1, f_4, f_5	-	f_2, f_6, f_8	f_3, f_7, f_9
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	-	-	-	-
$\|11\rangle_y$	-	-	-	-
$\xrightarrow{\theta_{\|0\rangle\to\|8\rangle}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1	-	f_2, f_6, f_8	f_3, f_7, f_9
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4, f_5	-	-	-
$\|11\rangle_y$	-	-	-	-
$\xrightarrow{\theta_{\|8\rangle\to\|12\rangle}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1	-	f_2, f_6, f_8	f_3, f_7, f_9
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4	-	-	-
$\|11\rangle_y$	f_5	-	-	-

Table A.6
Continuation of Table A.5.

$\xrightarrow{\theta_{\|2\rangle\to\|10\rangle}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1	-	f_2	f_3, f_7, f_9
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4	-	f_6, f_8	-
$\|11\rangle_y$	f_5	-	-	-
$\xrightarrow{\theta_{\|3\rangle\to\|11\rangle}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1	-	f_2	f_3
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4	-	f_6, f_8	f_7, f_9
$\|11\rangle_y$	f_5	-	-	-
$\xrightarrow{\theta_{\|10\rangle\to\|14\rangle}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1	-	f_2	f_3
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4	-	f_6	f_7, f_9
$\|11\rangle_y$	f_5	-	f_8	-
$\xrightarrow{\theta_{\|11\rangle\to\|15\rangle}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1	-	f_2	f_3
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4	-	f_6	f_7
$\|11\rangle_y$	f_5	-	f_8	f_9

a +1 operation on all time qubits conditioned on the most significant direction qubit to be in state $|1\rangle$. This circuit is shown in Fig. A.17.

Table A.7
Summation order of the D2Q9 velocity distribution functions first in x-direction and then in y-direction.

$\xrightarrow{H_{\|11\rangle_x\to\|10\rangle_x}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1	-	f_2	f_3
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4	-	f_6	f_7
$\|11\rangle_y$	f_5	-	f_8	f_9
$\xrightarrow{H_{\|11\rangle_x\to\|10\rangle_x}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1	-	f_2, f_3	-
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4	-	f_6, f_7	-
$\|11\rangle_y$	f_5	-	f_8, f_9	-
$\xrightarrow{RY_{\|10\rangle_x\to\|00\rangle_x}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1, f_2, f_3	-	-	-
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4, f_6, f_7	-	-	-
$\|11\rangle_y$	f_5, f_8, f_9	-	-	-
$\xrightarrow{H_{\|11\rangle_y\to\|10\rangle_y}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	f_1, f_2, f_3	-	-	-
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	f_4, f_6, f_7 f_5, f_8, f_9	-	-	-
$\|11\rangle_y$	-	-	-	-
$\xrightarrow{RY_{\|10\rangle_y\to\|00\rangle_y}}$	$\|00\rangle_x$	$\|01\rangle_x$	$\|10\rangle_x$	$\|11\rangle_x$
$\|00\rangle_y$	$f_1, \ldots, f_9$	-	-	-
$\|01\rangle_y$	-	-	-	-
$\|10\rangle_y$	-	-	-	-
$\|11\rangle_y$	-	-	-	-

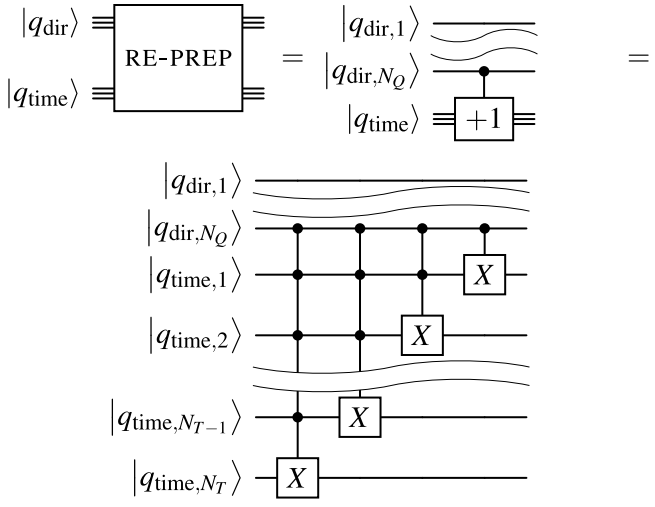


Fig. A.17. Simplified quantum circuit of RE-PREP step from Fig. 11 for the D1Q2 and D1Q3 state arrangements.

References

[1] R.P. Feynman, Quantum mechanical computers, Found. Phys. 16 (6) (1986) 507–532.
[2] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35Th Annual Symposium on Foundations of Computer Science, IEEE, 1994, pp. 124–134.
[3] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, 1996, pp. 212–219.
[4] D.A. Meyer, From quantum cellular automata to quantum lattice gases, J. Stat. Phys. 85 (5) (1996) 551–574.
[5] J. Yepez, Quantum lattice-gas model for computational fluid dynamics, Phys. Rev. E 63 (4) (2001) 046702.
[6] B.N. Todorova, R. Steijl, Quantum algorithm for the collisionless boltzmann equation, J. Comput. Phys. 409 (2020) 109347.
[7] M.A. Schalkers, M. Möller, Efficient and fail-safe collisionless quantum Boltzmann method, arXiv:2211.14269 (2022).
[8] L. Budinski, Quantum algorithm for the advection–diffusion equation simulated with the lattice boltzmann method, Quantum Inf. Process. 20 (2) (2021) 57.
[9] A.M. Childs, N. Wiebe, Hamiltonian simulation using linear combinations of unitary operations, arXiv:1202.5822 (2012).
[10] T. Shinde, L. Budinski, O. Niemimäki, V. Lahtinen, H. Liebelt, R. Li, Utilizing classical programming principles in the intel quantum SDK: implementation of quantum lattice boltzmann method, ACM Trans. Quantum Comput. 6 (1) (2025) 1–18.
[11] W. Itani, S. Succi, Analysis of carleman linearization of lattice boltzmann, Fluids 7 (1) (2022) 24.
[12] W. Itani, K.R. Sreenivasan, S. Succi, Quantum algorithm for lattice boltzmann (QALB) simulation of incompressible fluids with a nonlinear collision term, Phys. Fluid. 36 (1) (2024).
[13] J.-P. Liu, H.Ø. Kolden, H.K. Krovi, N.F. Loureiro, K. Trivisa, A.M. Childs, Efficient quantum algorithm for dissipative nonlinear differential equations, Proc. Natl. Acad. Sci. 118 (35) (2021) e2026805118.
[14] 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.
[15] D. Wawrzyniak, J. Winter, S. Schmidt, T. Indinger, C.F. Janßen, U. Schramm, N.A. Adams, Linearized quantum lattice-Boltzmann method for the advection-diffusion equation using dynamic circuits, Comput. Phys. Commun. (2025) 109856.
[16] 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
[17] A.A. Mohamad, Lattice boltzmann method, 70, Springer, 2011.
[18] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E.M. Viggen, The lattice Boltzmann method, 10, Springer, 2017.