Momentum exchange method for quantum Boltzmann methods
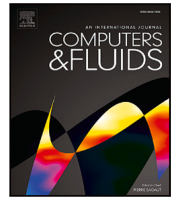
Schalkers, Merel A.; Möller, Matthias

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

Momentum exchange method for quantum Boltzmann methods

Merel A. Schalkers [*], Matthias Möller

Department of Applied Mathematics, Delft University of Technology, The Netherlands

ARTICLE INFO

ABSTRACT

The past years have seen a surge in quantum algorithms for computational fluid dynamics (CFD). These algorithms have in common that whilst promising a speed-up in the performance of the algorithm, no specific method of measurement has been suggested. This means that while the algorithms presented in the literature may be promising methods for creating the quantum state that represents the final flow field, an efficient measurement strategy is not available. This paper marks the first quantum method proposed to efficiently calculate quantities of interest (QoIs) from a state vector representing the flow field. In particular, we propose a method to calculate the force acting on an object immersed in the fluid using a quantum version of the momentum exchange method (MEM) that is commonly used in lattice Boltzmann methods to determine the drag and lift coefficients. In order to achieve this we furthermore give a scheme that implements bounce back boundary conditions on a quantum computer, as those are the boundary conditions the momentum exchange method is designed for.

1. Introduction

Computational fluid dynamics is one of the most frequently applied scientific endeavours, accounting for a large amount of the computational power used every day. As the power of classical computers grows, the demand in precision and scale for computational fluid dynamics increases similarly.

Future fault tolerant quantum computers promise a novel compute technology with an exponential computational power in the amount of qubits, leading to the natural questions of whether and how this novel method of computation can be used to simulate interesting problems of computational fluid dynamics (CFD). The question of a potential use for quantum computers in CFD was first researched by Yepez and his co-workers in the early 1990s, during quantum computing's first boom [1–5]. These papers describe a quantum distributed computing approach in which each grid point is described by six qubits, and the collision step at each grid point can be calculated on a separate machine. This has as a benefit that only small stable quantum computers are required and the collision step can be implemented on a quantum computer, making use of its inherently probabilistic nature. The downside of this approach is that streaming then needs to be done classically implying that complete measurement of the system and reinitialization are required in each time step. On top of that $6N$ qubits, where N is the number of grid points, are required. As the number of qubits grows linearly with the size of the grid, since the grid is typically very large for CFD problems and the number of qubits currently available is very low, this poses a significant problem.

After the initial QCFD research by Yepez et al. the field of Quantum Boltzmann methods became stagnant for over a decade. Whilst other QCFD approaches came to the forefront, quantum Boltzmann methods were largely forgotten until the more recent boom in 2019 starting with the paper by Todorova and Steijl [6]. Most recent are the methods presented in [6–13], that all have their strengths and weaknesses. Due to the heavy computational demands of CFD and precision required, all these methods require future fault tolerant quantum computers. The methods described in [6,10] include detailed quantum primitives for streaming and specular reflection but do not yet include a collision step. The methods described in [7,8,13] include a quantum primitive for collision using the linear combination of unitaries approach [14], as such measurement and reinitialization are required in each time step. Due to the high computational cost of such a 'stop-and-go' strategy caused by the difficulty of initialization and measurement errors, such techniques loose their practical advantage. The methods presented in [12,12,13] make use of Carleman linearization of the lattice Boltzmann equation as presented in [15]. The methods of [12] stand out as they are geared towards quantum simulation rather than the more general quantum computation paradigm.

What all these methods have in common is that after completing the final time step, a quantum state has been created that represents the entire flow field as a probability density distribution, e.g., encoded in the quantum state's amplitudes. So far, however, no efficient measurement strategies for this quantum state representing the flow field

---

* Corresponding author.
E-mail address: m.a.schalkers@tudelft.nl (M.A. Schalkers).

have been suggested. This implies that the current methods require the exponentially expensive reading out of the full quantum state to extract the entire flow field and post-process it on a classical computer afterwards. Consequently, any and all quantum advantages that were gained during the computation are lost. This paper marks the first that offers a quantum observable for the efficient reading out of the force vector acting on an object for the quantum Boltzmann method.

We first introduce the Lattice Boltzmann method in Section 2. In Section 3 we introduce the so-called Momentum Exchange Method (MEM) that can be used in combination with the Lattice Boltzmann method and bounce back boundary conditions to calculate the force acting on an object immersed in the fluid. Subsequently, in Section 4 we provide the reader with the basic ideas of the Quantum Lattice Boltzmann method (QLBM) and its encoding. Using this we introduce bounce back boundary conditions for QLBM in Section 5 and ultimately in Section 6 we introduce the Quantum Momentum Exchange Method. Finally Section 7 is dedicated to explaining how the QMEM can be efficiently implemented in practice and Section 6.1.0.1 gives insight into the computational costs.

2. The lattice Boltzmann method

The lattice Boltzmann method (LBM) is one of multiple widely-used computational approaches to model the behaviour of fluid flow with the aid of computers. It is based on the Boltzmann equation which can be written

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = \Omega(f),\tag{1}$$

where $f(\mathbf{x}, \mathbf{u}, t)$ is the distribution function of the particle density ρ, over space $\mathbf{x}$, with velocity $\mathbf{u}$, at time t. Here, Ω represents the collision term. We furthermore assume that no external force is present.

Since the actual collision term is relatively expensive to implement, in practice typically the BGK collision term is used [16]

$$\Omega(f) = -\frac{1}{-\tau}(f - f^{eq}),\tag{2}$$

where τ is the relaxation time and f^{eq} is the equilibrium function.

The Boltzmann equation can be discretized in both time, physical and velocity space leading to the lattice Boltzmann method. In the lattice Boltzmann method a time step can be denoted as

$$f_i\left(\mathbf{x} + \mathbf{c}_i \delta t, t + \delta t\right) = f_i(\mathbf{x}, t) - \frac{\delta t}{\tau}\left(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)\right),\tag{3}$$

where subscript i denotes the velocity direction.

What sets the Boltzmann method apart from other CFD approaches is that a single time step can be split into two consecutive parts, the so-called streaming and collision steps.

Writing the state of the system after collision as $f_i^\star(\mathbf{x}, t)$ we can get

$$f_i^\star(\mathbf{x}, t) = f_i(\mathbf{x}, t) - \frac{\delta t}{\tau}\left(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)\right),\tag{4}$$

for the collision step. Subsequently the streaming step is written as

$$f_i\left(\mathbf{x} + \mathbf{c}_i \delta t, t + \delta t\right) = f_i^\star(\mathbf{x}, t).\tag{5}$$

This ability to split the equation into two separate physically motivated steps leads to the Boltzmann method being implemented by performing collision and streaming separately and consecutively.

A popular way of classifying different combinations of dimensions and number of possible velocities is the so-called DdQq scheme. Here, d denotes the number of space dimensions considered and q the number of distinct velocities. In Fig. 1 we give four examples of different combinations of DdQq possible.

In this paper we are only considering the D1Q3, D2Q9 and D3Q27 cases.

We furthermore write $\mathbf{e}_i$ to represent the vector in the direction $i \in Q = \{0, 1, \ldots, q-1\}$ of the DdQq scheme. For example, in the D2Q9 system we have

$$\mathbf{e}_i = \begin{cases} (0,0) & \text{for } i = 0 \\ (1,0),(0,1),(-1,0),(0,-1) & \text{for } i = 1,2,3,4 \\ (1,1),(-1,1),(-1,-1),(1,-1) & \text{for } i = 5,6,7,8. \end{cases}\tag{6}$$

Therefore 2 qubits are necessary to represent the speed in each dimension, as the three options 'positive', 'negative' and 'standing still' need to be encoded.

3. Momentum exchange method

The momentum exchange method was proposed by Ladd [17] to determine the force acting on an object in order to calculate the drag and lift coefficients of an obstacle equipped with bounce back boundary conditions when the flow field is modelled by the Boltzmann method. Bounce back boundary conditions differ from the more intuitive specular reflection boundary conditions in that, upon contact with an object, the particle's velocity is reversed entirely instead of just the velocity normal to the object; see Fig. 2. Bounce back boundary conditions are often used in combination with the Lattice Boltzmann method [18]. In Section 5 we give an in-depth explanation of bounce back boundary conditions as well as how to implement them in our QLBM scheme. In this paper we adopt the momentum exchange method as described in [18]. Then the force exerted on the object by the particles can be expressed as

$$\mathbf{F} = \sum_{i \in Q}\left(\mathbf{e}_i f_i(\mathbf{x}_f, t) - \mathbf{e}_{\bar{i}} f_{\bar{i}}(\mathbf{x}_f, t)\right).\tag{7}$$

In the above expression $\mathbf{x}_f$ refers to a point in the fluid space adjacent to the obstacle and $\bar{i}$ represents the velocity of the particles after particles with velocity i have impinged on the object. This expression assumes that there is no fluid inside the object and as such only takes the momentum exchange outside of the object into account. Since we are using bounce back boundary conditions we have $\mathbf{e}_{\bar{i}} := -\mathbf{e}_i$ and $f_i(\mathbf{x}_f, t) = f_{\bar{i}}(\mathbf{x}_f, t)$ by definition, therefore we can rewrite Eq. (7) to

$$\mathbf{F} = \sum_{i \in Q} 2\mathbf{e}_i f_i(\mathbf{x}_f, t).\tag{8}$$

As force is composed of magnitude and direction it is expressed by a d dimensional vector with subscript j denoting its jth dimensional component, i.e.

$$F_j = \left(\sum_{i \in Q} 2\mathbf{e}_i f_i(\mathbf{x}_f, t)\right)_j.\tag{9}$$

4. Quantum lattice Boltzmann method

The quantum lattice Boltzmann method (QLBM) is, as the name suggests, the quantum analog of the lattice Boltzmann method. Similar to the classical lattice Boltzmann method the QLBM consists of the initialization of the problem, methods for streaming and collision, an approach to impose boundary conditions and, finally, a measurement procedure to extract application-specific QoIs. This paper introduces an efficient measurement procedure that can be used in combination with existing QLBMs. As such we abstain from presenting concrete methods for collision, streaming or state preparation. Instead we focus on explaining a set-up for the measurement procedure, which can be used with any QLBM method that uses a similar encoding scheme.

As the measurement procedure in practice should be fitted to the quantum state that it is used on we will present how the density function is encoded in the quantum state for this method. The density function encoding presented below is similar to the ones presented in [6–8,10], as such the measurement procedure presented here can be used with those papers.
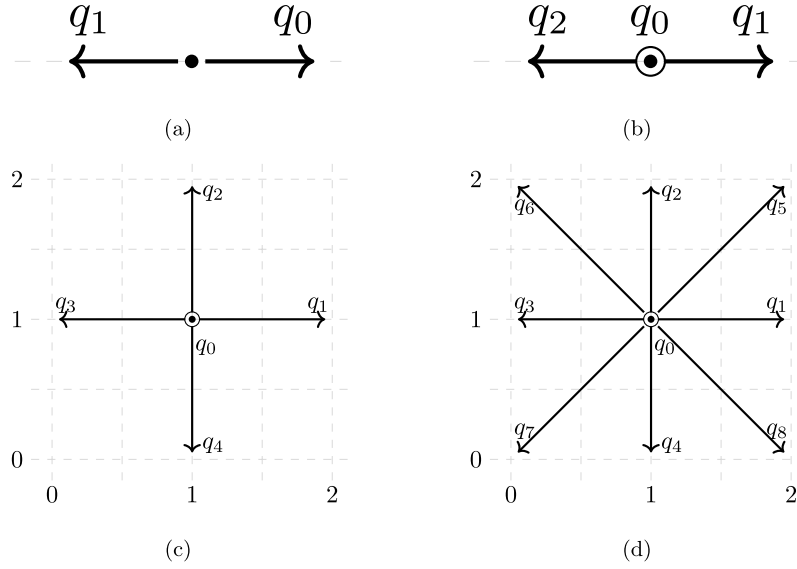
Fig. 1. Four examples of different types of DdQq possible. Fig. 1(a) portrays the D1Q2 setting and Fig. 1(b) portrays the D1Q3 setting (where a stationary particle can be included). Fig. 1(c) portrays the D2Q5 setting and Fig. 1(d) shows the D2Q9 setting.

Flow field encoding. Building on our previous work [10], the quantum encoding of the discretized density function reads

$$|\underbrace{a_{n_a} \dots a_1}_{\text{ancillae}} \overbrace{g_{n_g} \dots g_1}^{\text{position}} \underbrace{v_{n_v} \dots v_1}_{\text{velocity}}\rangle, \qquad (10)$$

whereby the positional and velocity qubits are split into d groups, one for each dimension. More specifically, zooming in on the positional qubits, we get

$$|g_{n_g} \dots g_1\rangle = |g^d_{n_{g_d}} \dots g^d_1 g^{d-1}_{n_{g_{d-1}}} \dots g^{d-1}_1 \dots g^1_{n_{g_1}} \dots g^1_1\rangle, \qquad (11)$$

where $g^j_{n_{g_j}} \dots g^j_1$ encodes the jth dimension of the location of grid points by representing the binary value of the location.

Similarly if we write out the velocity qubits for the encoding explicitly we get

$$|v_{n_v} \dots v_1\rangle = |v^d v^d_{\text{dir}} \dots \dots v^1 v^1_{\text{dir}}\rangle, \qquad (12)$$

where v^j_{dir} expresses the direction (positive or negative) of the particle in dimension j and the v^j qubits express whether a particle has a nonzero velocity in dimension j. Notice that this order is different from the one presented in [10] where the v_{dir} qubits are grouped together, this is done simply to make the observable in Section 6.1 easier to visualize as a matrix. Another difference from the setup presented in [10] is that here we are only considering the D1Q3, D2Q9 and D3Q27 cases leading to exactly two velocity qubits per dimension.

The ancilla qubits are used for several different purposes throughout the QLBM method. In this paper we will only highlight the labels and purposes of the ancillae that are used in the quantum bounce back boundary conditions implementation and the quantum momentum exchange method presented in Sections 5 and 6, respectively. We identify the $a_{v,i}$ ancilla qubits that indicate whether in this time step the associated particles are streamed in dimension i. Furthermore we make use of the a_o which is the ancilla qubit that indicates whether or not a particle is in an object and the bounce back boundary conditions need to be applied.

5. Quantum bounce back boundary conditions

One of the most commonly used boundary conditions in practical LBM is the bounce back boundary condition which amounts to fully reflecting the direction of particles that get into contact with
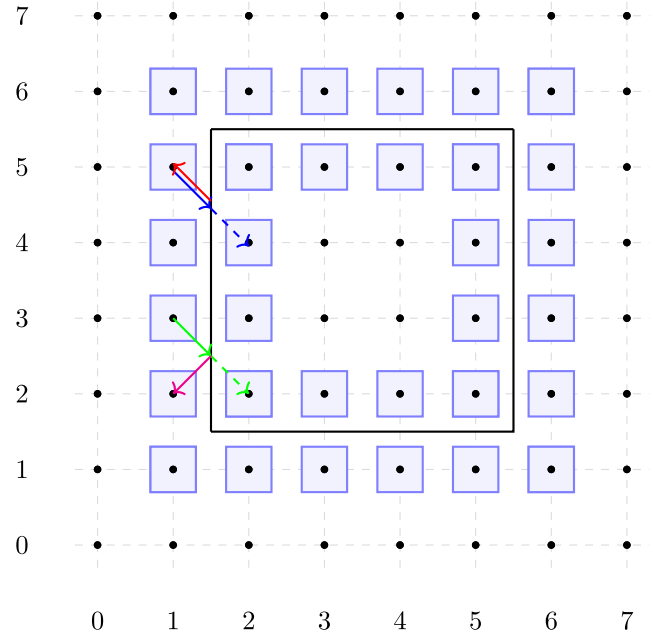

Fig. 2. Illustration of bounce back boundary conditions (top, in red and blue arrows) versus specular reflection boundary conditions (bottom, in green and magenta).

obstacles and resetting them to their original position inside the flow domain [18,19]. This is different from the specular reflection boundary conditions that we adopted in our earlier work [10], which reverses only the normal component of the velocity vector. Fig. 2 illustrates the difference between the two types of boundary conditions.

The algorithm for implementing bounce back boundary conditions in a classical LBM can be summarized as follows. First, the particles that virtually travelled into the obstacle have their velocity direction reversed in all dimensions and subsequently these particles are placed outside of the obstacle. The particles are placed in the correct position outside of the obstacle by moving one grid point in the dimension(s) that they previously moved in.

For implementing the bounce back boundary condition as a quantum primitive we require only one ancilla qubit a_o to indicate whether
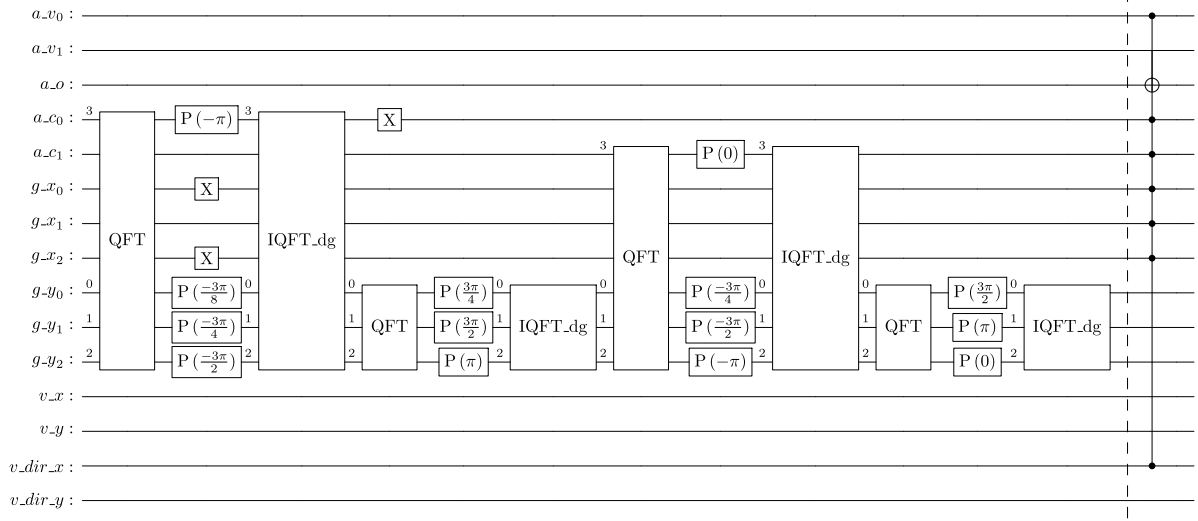
Fig. 3. The first part of the bounce back boundary conditions, applied to the example of Fig. 6 to properly reset the particles moving to the right in the x-direction hitting the particles on the left wall. This part of the algorithm sets the ancilla qubit, indicating that particles have virtually travelled into the object and need to have their velocity reversed and be moved out.
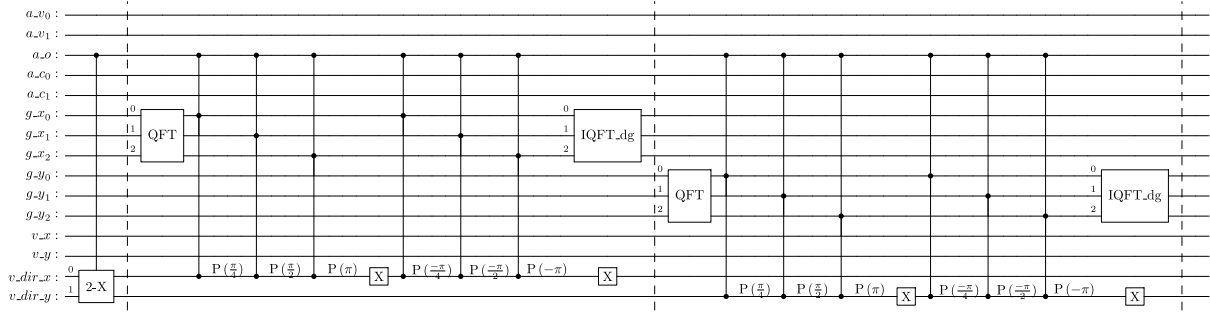
Fig. 4. This figure shows part of the bounceback boundary conditions quantum algorithm where the velocity qubits of the particles hitting the object get reversed and subsequently moved back out of hte object.

a particle has virtually moved into an object. The ancilla a_o is initialized in the $|0\rangle$ state and flipped to $|1\rangle$ when a particle has virtually travelled into one of the points inside the obstacle. We check whether a particle has virtually travelled into the object using the efficient object encoding method as described in Section 5 of [10]. In Fig. 3 we show how this efficient object encoding method can be implemented for the example give in 6. In Fig. 3 we show the quantum comparison operation that can check whether or not a particle has come into contact with the wall from (2,2) to (2,5). This is done by checking whether the location on the x-axis is equal to two as is done by applying an X gate to the g_{x_0} and g_{x_2} qubit. We use two quantum comparison operations to check whether $2 \leq y \leq 5$ as can be seen in the picture by the QFT operations followed by rotations and IQFT. The mathematics behind this procedure is explained in Section 5.4 of [10].

As a next step we flip the state of the v_{dir}^j qubits for all dimensions j controlled on the state of the a_o ancilla. By doing this we make sure that the velocity direction is reversed in all dimensions after contact with an obstacle as is required for bounce back boundary conditions. And subsequently the particles are moved by one position controlled on the a_v^j, v_{dir}^j, and a_o qubits to ensure that the particles move one step in the correct direction in the dimension(s) that they moved in when they moved into the obstacle and of course to ensure that this only happens after the particles moved into the obstacle.

This is done by the operation controlled double NOT operation shown in the beginning of Fig. 4. Subsequently, as can be seen in the same figure, controlled or whether or not the a_o qubit is in the state 1, we stream in the x and y dimension, thereby making sure we only

stream the particles that just collided with the object. This is done to ensure that the particles are set back outside of the object again.

Finally the a_o qubits need to be reset to $|0\rangle$ before we can start the next time step. As in our previous work [10] the blue and green encircled points outside of the object constitute the trivial case in Fig. 6. We reset the a_o qubits controlled on if we are in one of the blue (green) encircled points, the direction of the x (y) velocity and the ancilla qubit indicating whether we moved in the dimension in this time step a_v^1 (a_v^2). Specifically we reset the ancilla qubit a_o if we are in a blue (green) encircled grid point outside the object and $a_v^1 = 1$ ($a_v^2 = 1$) and v_{dir}^1 (v_{dir}^2) points away from the object. Using this logic the a_o qubits are reset to $|0\rangle$ for each wall separately.

Resetting the a_o qubit is a bit more difficult around the edges as indicated with black and orange encircled points in Fig. 6.

For the black encircled points outside the corner we need to reset the ancilla qubit a_o if and only if both v_{dir}^1 and v_{dir}^2 point in the direction away from the object and $a_v^1 = a_v^2 = 1$ holds.

As for the orange encircled 'side-edge' grid point we first reset the a_o ancilla in the same way as for the blue (green) encircled points described above. Now we only need to note that for the case described by the red arrow in Fig. 6 we have wrongly flipped the a_o ancilla qubit and so we need to verify whether we are in the 'red arrow' case by checking if v_{dir}^1 and v_{dir}^2 pointed in the direction of the arrow and if $a_v^1 = a_v^2 = 1$ holds. If the particle is in a state where $a_v^1 = a_v^2 = 1$ and v_{dir}^1 and v_{dir}^2 are such that the particle is in an red arrow case the a_o ancilla get flipped again, back to the original state of $|0\rangle$.

In Fig. 5 we show how the ancilla qubits are reset. In this picture it can be seen that controlled on the conditions described above, an X
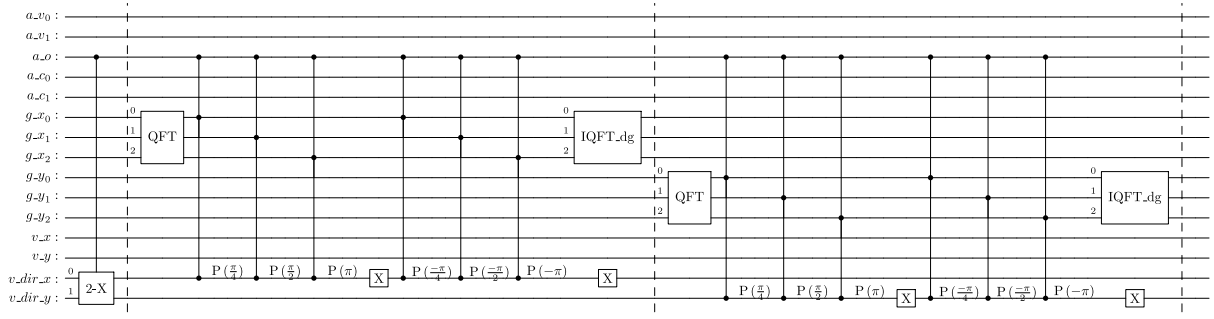
Fig. 5. This figure shows part of the bounceback boundary conditions quantum algorithm where the ancilla qubit a_O qubits of the particles that hit the object and got reversed and subsequently moved back out of hte object gets reset.
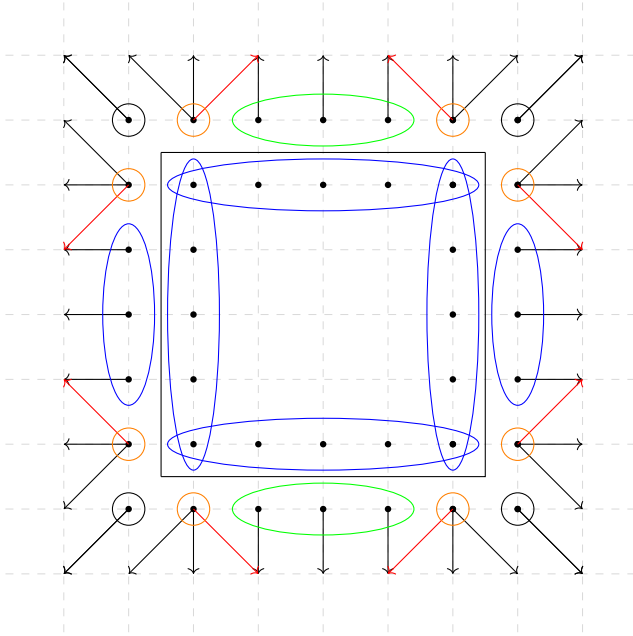

Fig. 6. Illustration of all possible corner cases to be taken into account when particles collide with an obstacle (black box). The arrows and colours indicate the physically correct behaviour for a fail-safe implementation of the boundary conditions.

gate is applied to the ancilla qubits to re-set them to the zero state if and only if they were in the one state to begin with. Resetting the ancilla qubits in this particular use case is non-trivial as the ancilla qubits were originally set based on location and then streamed, so we need to keep track on how all the particles moved in order to be able to reset them properly. The core is to now check if the particles are just outside the object pointing away from the object in such a way that could only have happened if they just reflected from the object. We can safely reset the ancilla qubits based on this, the circuit is given in Fig. 5.

6. Quantum momentum exchange method

In this section we explain how the momentum exchange method can be expressed as an observable for the encoding described in Section 4. In order to do this we will first change the density encoding of the quantum state into a rooted density encoding. Using this rooted density encoding we can subsequently define the observable that calculates the force using Eq. (8) and finally we describe how this method can be implemented as an executable quantum circuit.

Rooted density encoding. Since the momentum exchange method is linear in nature, whereas quantum observables are quadratic, it is advisable to change from encoding the density function $f_i(\mathbf{x}, t)$ in the quantum state $|\psi\rangle$ to an encoding of the square root of the density function $\sqrt{f_i(\mathbf{x}, t)}$. Such a shift to a rooted density encoding can be done without altering any subsequent circuits in the methods [6,10]. In practice the densities should be rooted before the start of the quantum algorithm, therefore the information will be initialized into the quantum state such that the density is already rooted and no quantum method is required for this task.

6.1. Momentum exchange method as an observable

We now derive the observable that calculates the force exerted on the object using the momentum exchange method. As force is described using a vector, we need to calculate the values of the vector for all d dimensions. Here we show how to calculate this vector using $2d$ different observables, where each observable calculates the value of the force vector in one spatial dimension, $d \in \{1, 2, 3\}$, and one velocity direction. This is done to make the resulting observable easier to portray and explain, since all the observables can be expressed as diagonal matrices they do commute and so they can all be measured in the same runs.

Considering the encoding described above, Eq. (8) can be evaluated from the quantum state $|\psi\rangle$ encoding $\sqrt{f_i(\mathbf{x}, t)}$, by the diagonal observable O_{OME} to be specified below. The diagonal of the matrix expressing the observable O_{OME} is built up using B_{OME} matrices, which are placed at grid points in the fluid domain directly adjacent to the obstacle. All the other indices of the matrix expressing O_{OME} will remain zero. An example of this is the matrix

$$O_{\mathrm{OME}} = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & B_{\mathrm{OME}} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}, \tag{13}$$

where the dots represent 4×4 matrices with only 0 indices and B_{OME} is a 4×4 matrix that can be written as

$$B_{\mathrm{OME}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{14}$$

The example given in Eq. (13) represents a 1-dimensional case with 4 grid points and three possible speeds (one in the positive and one in the negative x-direction as well as the zero speed) and the wall adjacent to the second grid point as represented in Fig. 7. Since $B_{\mathrm{OME}} = B_{\mathrm{OME}}^\dagger$, both B_{OME} and O_{OME} are Hermitian and therefore O_{OME} constitutes a quantum observable. It can easily be seen that any other distribution of B_{OME} along the diagonal will also lead to a quantum observable.

For the D1Q3 case with four grid points described above we have the following quantum state encoding

$$|\psi\rangle = \frac{1}{\sqrt{\sum_{x,i} f_i(\mathbf{x}, t)}} \sum_{x,i} \sqrt{f_i(\mathbf{x}, t)} |g_2^1 g_1^1 v^1 v_{\mathrm{dir}}^1\rangle, \tag{15}$$
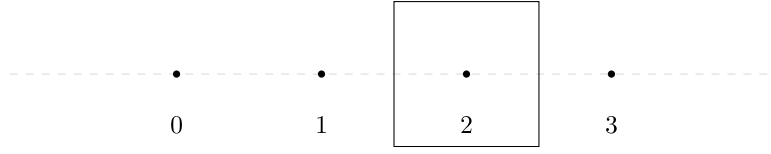
Fig. 7. Example of the D1Q3 case with four grid points and one obstacle located on the third grid point.

where $g_2^1 g_1^1$ represent the binary value of the location of the x-axis, $|v^1 v_{\text{dir}}^1\rangle = 10$ indicates streaming in the positive x-direction, $|v^1 v_{\text{dir}}^1\rangle = 11$ indicates streaming in the negative x-direction and $|v^1 v_{\text{dir}}^1\rangle = 01$ as well as $|v^1 v_{\text{dir}}^1\rangle = 00$ indicates that the particle is not streaming in the x-direction. Here and in the remainder of this Section we do not take into account the ancilla qubits as they play no role in the final density function and as such will not be part of the measurement process.

With the above convention, the quantum state can be written as the coefficient vector relative to the computational basis as follows:

$$|\psi\rangle = \sum_{x,v} \alpha_{x,v} |g_2^1 g_1^1 v^1 v_{\text{dir}}^1\rangle = \frac{1}{\sqrt{\sum_{x,i} f_i(\mathbf{x},t)}} \begin{bmatrix} 0 \\ \sqrt{f_0(0,t)} \\ \sqrt{f_1(0,t)} \\ \sqrt{f_2(0,t)} \\ 0 \\ \sqrt{f_0(1,t)} \\ \sqrt{f_1(1,t)} \\ \sqrt{f_2(1,t)} \\ 0 \\ \sqrt{f_0(2,t)} \\ \sqrt{f_1(2,t)} \\ \sqrt{f_2(2,t)} \\ 0 \\ \sqrt{f_0(3,t)} \\ \sqrt{f_1(3,t)} \\ \sqrt{f_2(3,t)} \end{bmatrix}. \quad (16)$$

Using expression (16) and some basic linear algebra it follows that

$$\langle\psi|O_{\text{OME}}|\psi\rangle = \frac{2f_1(1,t)}{\sum_{x,i} f_i(\mathbf{x},t)}. \quad (17)$$

Since the value of $\sum_{x,i} f_i(\mathbf{x},t)$ is known when starting the algorithm we can simply multiply Eq. (17) by $\sum_{x,i} f_i(\mathbf{x},t)$ to find the value of the force we wish to calculate as described in Eq. (8), which can subsequently be used to calculate the drag and lift coefficient [17].

Extension to more dimensions. This method can easily be extended to more dimensions by noticing that the B_{OME} matrix is of size $2^{n_v} \times 2^{n_v}$ and should consist of only one non-zero element. This non-zero element will always be placed on the diagonal at the position of the basis state $|v^i\rangle$.

6.1.0.1. Complexity analysis. The number of measurements required to determine the force with an ϵ precision using our proposed approaches depends on multiple factors. As long as the total number of grid points located inside and adjacent to the boundary of the obstacle is polynomial in the total number of grid points, the number of diagonal elements that are non-zero in the observable is polynomial in the size of the grid. This means that the number of non-zero elements in the observable is not exponentially small in the total size of the system. Therefore, in this case, we wish to measure a subspace that is only polynomially small in the total size of the system which is feasible without exponential overhead.

7. Practical implementation of the momentum exchange method on a quantum computer

Realizing an observable on a real-world quantum computer amounts to implementing a quantum circuit that translates the observable to

measurements in the computational Z-basis. We will now present the quantum circuit that translates the observable described in Section 6.1 for determining the force in one dimension in one direction to measuring one qubit in the Z-basis, making the process clear and easily implementable on a quantum device.

We will first describe how this operation can be applied by using an already implemented circuit for the bounce back boundary condition and we will subsequently show that this operation indeed transforms the described observable to one that consists of a Z measurement on one qubit.

7.1. Implementation using ancilla qubits for bounce back boundary conditions

We have implemented a method to measure the expectation value of the described observable by measuring only one qubit. This is done using the implementation of the bounce back boundary conditions. In this implementation a qubit a_o gets flipped to indicate that a particle is inside an object. To determine the expectation value of the observable we will use these ancilla qubits differently. We will from now on call this a_o ancilla qubit that was used for the bounce back boundary conditions $a_{o,+}$ and we define a second ancilla qubit $a_{o,-}$. These ancilla qubits will be flipped if a force was exerted on the object in a positive or negative direction, respectively. In order to do this we apply a multi-controlled NOT operation controlled on the qubits to determine whether we are in the object and the qubit indicating the direction of the particles in the dimension to the $a_{o,+}$ ($a_{o,-}$) qubits.

By doing this we are extracting the relative density of particles that come into contact with an obstacle in the positive and negative direction for the considered dimension. Subsequently we measure the qubits $a_{o,+}$ and $a_{o,-}$ and subtract the expectation value of $a_{o,-} = 1$ from $a_{o,+} = 1$. The resulting value expresses the relative pressure in the positive/negative direction. Fig. 8 shows the quantum circuit that implements this quantum momentum exchange method for the example of Fig. 6.

7.2. Proof of method

In this section we show that using the method described above, we indeed calculate the force exerted on an object in one dimension as expressed in Eq. (8).

We flip the $a_{o,+}$ ancilla qubit in the case that particles have impinged on the object and the particles have a positive velocity in the x-direction. Therefore, after re-arranging some qubits, we can write

$$\left(\sqrt{f_{v_0}(x_0,t)} |\left(g_{n_g} \dots g_1\right)_0 \left(v_{n_v} \dots v_1\right)_0\rangle + \dots + \right.$$
$$\left. \sqrt{f_{v_1}(x_1,t)} |\left(g_{n_g} \dots g_1\right)_1 \left(v_{n_v} \dots v_1\right)_1\rangle \right) |0\rangle_{a_{o,+}} +$$
$$\left(\sqrt{f_{v_2}(x_2,t)} |\left(g_{n_g} \dots g_1\right)_2 \left(v_{n_v} \dots v_1\right)_2\rangle + \dots + \right.$$
$$\left. \sqrt{f_{v_3}(x_3,t)} |\left(g_{n_g} \dots g_1\right)_3 \left(v_{n_v} \dots v_1\right)_3\rangle \right) |1\rangle_{a_{o,+}}, \quad (18)$$

where for v_i and x_i the subscript is simply used to indicate that a specific value for the location and velocity is considered and similarly for the qubits $\left(g_{n_g} \dots g_1\right)_i$, $\left(v_{n_v} \dots v_1\right)_i$ the subscript is used to indicate the velocity and grid point qubits are in a specific state. Here the states $|\left(g_{n_g} \dots g_1\right)_0 \left(v_{n_v} \dots v_1\right)_0\rangle + \dots + |\left(g_{n_g} \dots g_1\right)_1 \left(v_{n_v} \dots v_1\right)_1\rangle$
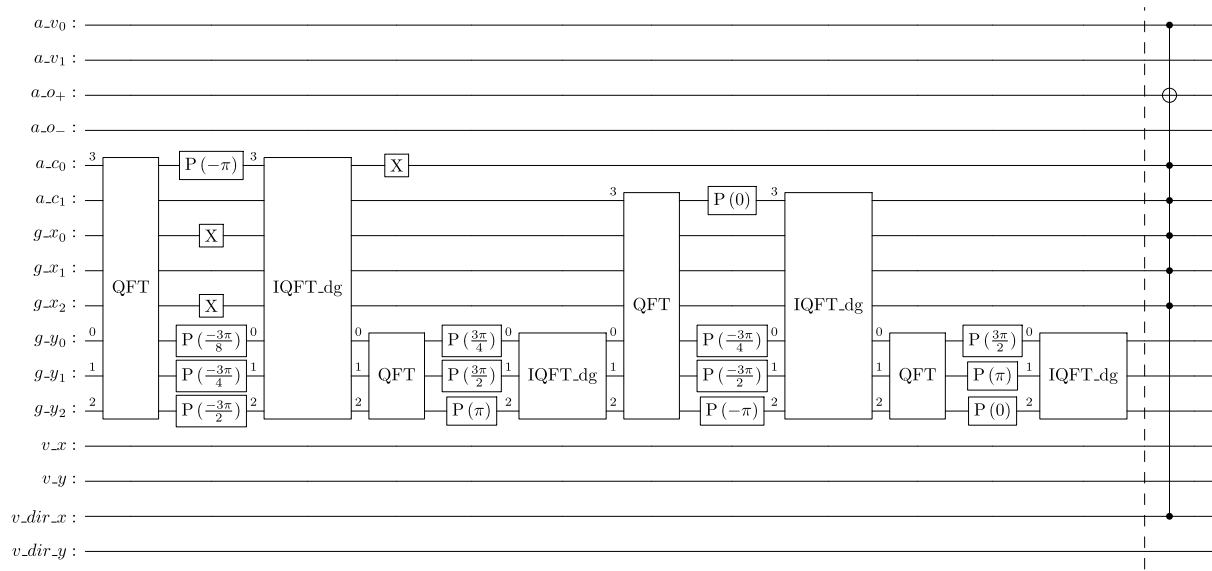
Fig. 8. The quantum algorithm for the QMEM, applied to the example of Fig. 6 to determine the force of the particles moving to the right in the *x*-direction hitting the particles on the left wall. This part of the algorithm sets the ancilla qubit, indicating that particles have virtually travelled into the object. Subsequently the ancilla qubit $a_{o,+}$ is measured to determine the force.

describe exactly the particles that do not impinge on the object in the positive *x*-direction in the current time step, as the $a_{o,+}$ qubit is in the $|0\rangle$ state. Similarly the states $|\left(g_{n_g} \ldots g_1\right)_2 \left(v_{n_v} \ldots v_1\right)_2\rangle +$ $\cdots + |\left(g_{n_g} \ldots g_1\right)_3 \left(v_{n_v} \ldots v_1\right)_3\rangle$ represent the relative densities of the particles that have impinged the object in the positive *x*-direction. From this we can conclude that the total probability of finding $|a_{o,+}\rangle = |1\rangle$ upon measurement is equal to

$$f_{v_2}\left(x_2, t\right) + \cdots + f_{v_3}\left(x_3, t\right), \tag{19}$$

which is precisely equal to the relative density of particles hitting the object with a positive velocity and which is precisely what we wish to measure.

8. Conclusion

In this paper we have presented a quantum approach to determine the force of the flow field acting on an object immersed in the fluid via an efficient and easily implementable measurement procedure for the quantum lattice Boltzmann method. To the best of our knowledge, this is the first time that efficient measurement strategies are addressed in the QLBM literature. Previous works are limited to reading out the entire flow field which cannot be realized efficiently on a quantum computer, thereby destroying any quantum advantage.

Our approach represents the quantum analog of the momentum exchange method and consists of a quantum primitive for implementing bounce back boundary conditions at the end of each time step and an observable that can be easily implemented as measurements in the computational basis to obtain the forces exerted by the fluid on an internal object.

CRediT authorship contribution statement

Merel A. Schalkers: Writing – original draft, Investigation, Formal analysis, Conceptualization. **Matthias Möller:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they received funding to conduct their research under the joint research program "Quantum Computational

Data availability

No data was used for the research described in the article.

References

[1] Yepez Jeffrey. Quantum computation of fluid dynamics. In: Quantum computing and quantum communications: lecture notes in computer science, 1998, URL https://www.phys.hawaii.edu/~yepez/papers/publications/pdf/1999LectNotesCompSciVol1509Pg35.pdf.

[2] Yepez Jeffrey. Quantum lattice-gas model for computational fluid dynamics. Phys Rev E 2001. URL https://journals.aps.org/pre/abstract/10.1103/PhysRevE.63.046702.

[3] Yepez Jeffrey, Boghosian Bruce. An efficient and accurate quantum lattice-gas model for the many-body Schrödinger wave equation. Comput Phys Comm 2001. URL https://www.phys.hawaii.edu/~yepez/static/papers/pdf/2002CompPhysCommVol146No3%2C15Pg280.pdf.

[4] Yepez Jeffrey. Quantum lattice-gas model for the Burgers equation. J Statist Phys 2002. URL https://link.springer.com/article/10.1023/A:1014514805610.

[5] Pravia Marco A, Chen Zhiying, Yepez Jeffrey, Cory David G. Experimental demonstration of quantum lattice gas computation. Quantum Inf Process 2003. URL https://link.springer.com/article/10.1023/A:1025835216975.

[6] Todorova BN, Steijl R. Quantum algorithm for the collisionless Boltzmann equation. J Comput Phys 2020;409. http://dx.doi.org/10.1016/j.jcp.2020.109347.

[7] Budinski Ljubomir. Quantum algorithm for the advection-diffusion equation simulated with the lattice Boltzmann method. Quantum Inf Process 2021 2020. URL https://link.springer.com/article/10.1007/s11128-021-02996-3.

[8] Budinski Ljubomir. Quantum algorithm for the Navier-Stokes equations by using the streamfunction-vorticity formulation and the lattice Boltzmann method. Int J Quantum Inf 2021. URL https://arxiv.org/abs/2103.03804.

[9] Moawad Y, Vanderbauwhede W, Steijl R. Investigating hardware acceleration for simulation of CFD quantum circuits. Front Mech Eng 2022. http://dx.doi.org/10.3389/fmech.2022.925637, URL https://www.frontiersin.org/articles/10.3389/fmech.2022.925637/full.

[10] Schalkers Merel A, Möller Matthias. Efficient and fail-safe quantum algorithm for the transport equation. J Comput Phys 2024;502:112816. http://dx.doi.org/10.1016/j.jcp.2024.112816.

[11] Steijl René. Quantum circuit implementation of multi-dimensional non-linear lattice models. MDPI: Appl Sci 2023. http://dx.doi.org/10.3390/app13010529, URL https://www.mdpi.com/2076-3417/13/1/529.

[12] Succi Sauro, Itani Wael, Sreenivasan Katepalli, Steijl René. Ensemble fluid simulations on quantum computers. 2023, URL https://arxiv.org/pdf/2304.05410.pdf.

[13] Sanavio Claudio, Succi Sauro. Lattice Boltzmann-carleman quantum algorithm and circuit for fluid flows at moderate Reynolds number. 2023, URL https://arxiv.org/pdf/2310.17973.pdf.

[14] Childs Andrew M, Wiebe Nathan. Hamiltonian simulation using linear combinations of unitary operations. Quantum Inf Comput 2012. URL https://arxiv.org/pdf/1202.5822.pdf.

[15] Itani Wael, Succi Sauro. Analysis of carleman linearization of lattice Boltzmann. Fluids 2022;7(1). http://dx.doi.org/10.3390/fluids7010024, URL https://www.mdpi.com/2311-5521/7/1/24.

[16] Bhatnagar PL, Gross EP, Krook M. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. Phys Rev 1954;94:511–25. http://dx.doi.org/10.1103/PhysRev.94.511, URL https://link.aps.org/doi/10.1103/PhysRev.94.511.

[17] Ladd Anthony J. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. J Fluid Mech 1993;271:285–309.

[18] Krüger Timm, Kusumaatmaja Halim, Kuzmin Alexandr, Shardt Orest, Silva Goncalo, Viggen Erlend Magnus. The lattice Boltzmann method. Springer; 2017, URL https://link.springer.com/book/10.1007/978-3-319-44649-3.

[19] Schiller Ulf D. Thermal fluctuations and boundary conditions in the lattice Boltzmann method [Ph.D. thesis], Johannes Gutenberg-Universität; 2008.