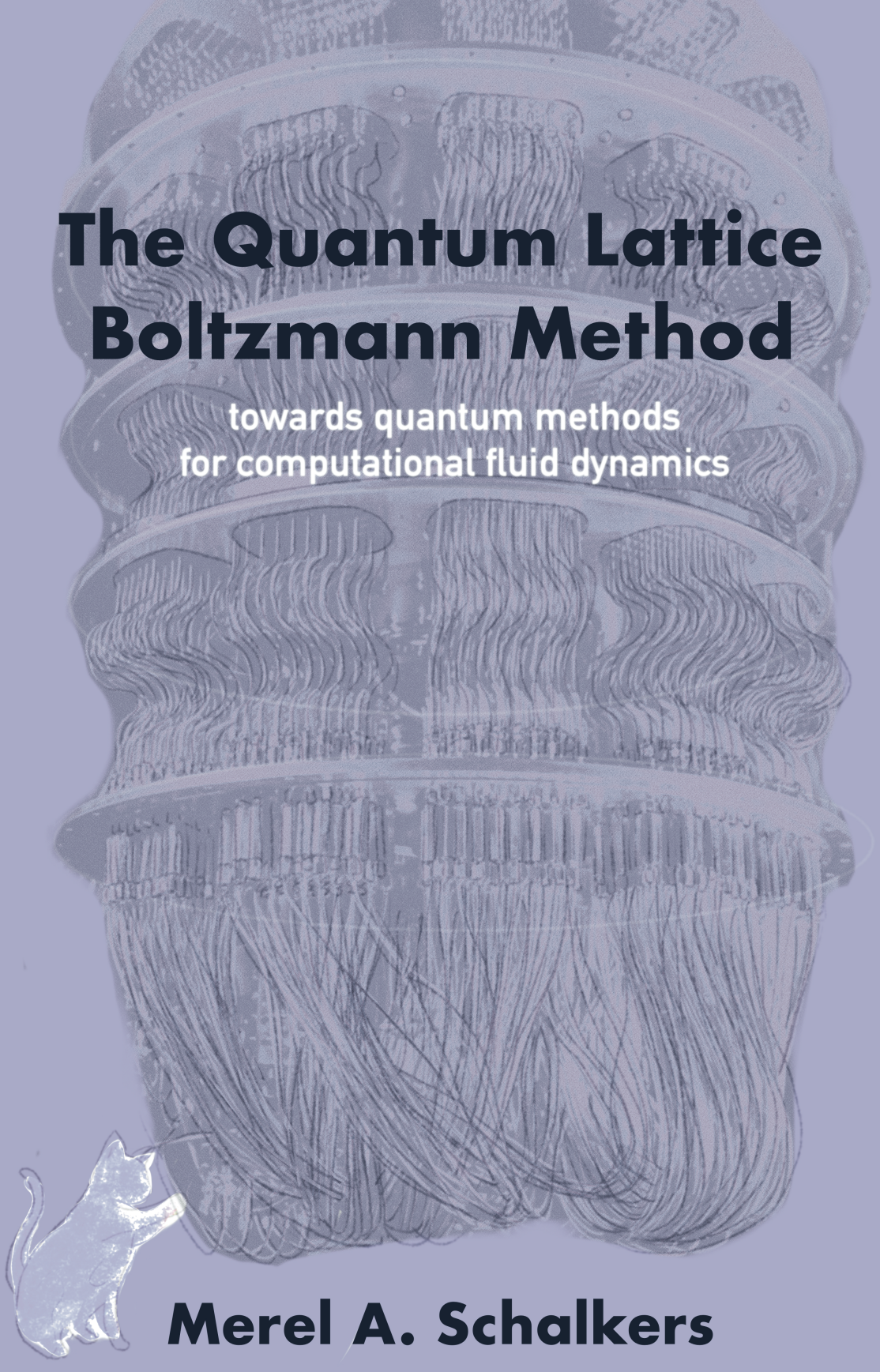
The Quantum Lattice Boltzmann Method

towards quantum methods
for computational fluid dynamics



Merel A. Schalkers

THE QUANTUM LATTICE BOLTZMANN METHOD

TOWARDS QUANTUM METHODS FOR COMPUTATIONAL FLUID DYNAMICS

THE QUANTUM LATTICE BOLTZMANN METHOD

TOWARDS QUANTUM METHODS FOR COMPUTATIONAL FLUID DYNAMICS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. H. Bijl,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 3 februari 2026 om 15:00 uur

door

Merel Annelise SCHALKERS

Master of Science in Applied Mathematics,
Technische Universiteit Delft,
geboren te Voorhout, Nederland

Dit proefschrift is goedgekeurd door de promotors.

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter	
Em. Prof. dr. ir. C. Vuik,	Technische Universiteit Delft	Promotor
Dr. rer. nat. M. Möller,	Technische Universiteit Delft	Promotor
Dr. D. de Laat,	Technische Universiteit Delft	Copromotor
Onafhankelijke leden:		
Prof. dr. ir. J.T. Padding	Technische Universiteit Delft	
Prof. dr. J. Kowalski	Rheinisch-Westfälische Technische Hochschule Aachen, Duitsland	
Prof.dr. J. Schumacher	Technische Universität Ilmenau, Duitsland	
Dr. R. Steijl	University of Glasgow, Verenigd Koninkrijk	
Prof. dr. ir. M.B. van Gijzen	Technische Universiteit Delft	reservelid

Het onderzoek in dit proefschrift is mede gefinancierd door Fujitsu ltd en de Rijksdienst voor Ondernemend Nederland onder projectnummer PPS23-3-03596728.

Keywords: Quantum Computing, The Boltzmann Equation, The lattice Bolzmann Method, Quantum Computational Fluid Dynamics

Cover Design by E.M. Schalkers

ISBN 978-94-6536-019-5

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

Voor mijn zus, moeder, oma en
alle vrouwen die mij voorgegaan zijn.

CONTENTS

SUMMARY

There are certain computational problems that a quantum computer is expected to be able to solve more efficiently than a classical compute device. However, due to their specific structure and properties, quantum computers will not be beneficial for all types of computations. One potential use-case of interest is the so-called lattice Boltzmann method, which is a computational approach that can be used to model the flow of fluid. The lattice Boltzmann method models fluid flow by dividing the process into its two natural steps of streaming and collision and simulating them separately. This means that the method is naturally split up into a linear and a local step, this structure is potentially interesting for quantum computers as they portray naturally linear behavior and local computations can be done in parallel. The aim of this thesis is to determine how quantum computers could be used to implement a quantum Lattice Boltzmann method and if such an implementation would constitute a speed-up over classical methods. In Chapter 1 we give a brief introduction to quantum computing, computational fluid dynamics (CFD), the Boltzmann method and the field of quantum CFD.

Chapter 2 presents our quantum implementation of a lattice Boltzmann inspired approach to solve the transport equation. The transport equation is essentially a Boltzmann equation without external forces or a collision term. The transport equation is sometimes referred to as the Vlasov equation or the collisionless Boltzmann equation. Due to the inherent linearity of this equation we can efficiently solve it on a quantum computer. In this chapter we present a more efficient method of streaming than used by former approaches by making use of the quantum Fourier transform. We furthermore present efficient and fail-safe methods for simulating the interaction of particle densities with the boundaries of an object.

In Chapter 3 we develop a measurement strategy that can be used to determine the force exerted on an object by the particle flow. The measurement strategy is based the momentum exchange method and therefore called the quantum momentum exchange method. This strategy can be used to efficiently determine the force exerted on an object in certain use cases.

Chapter 4 discusses the problem of extending quantum methods for the transport equation to include a collision step and quantum collision methods to include a streaming step. This is done be realizing that transport methods make use of an amplitude based encoding and collision only methods make use of a computational basis state encoding. We use this realization to show that in fact, for those encodings, it is not possible to add a collision or streaming step in a way that maintains unitarity. The reason we cannot add a collision step to an amplitude based encoding is because the operator implementing a collision step that makes physical sense cannot be a unitary operator.

Similarly we cannot design a streaming step for computational basis state encodings as streaming is inherently non-local and in order to determine the encoding at each grid point we need information from all the adjoining grid points to do this correctly, this again runs into trouble with the fact that only unitary operations are possible on a quantum computer.

In Chapter 5 we present the space-time encoding. The space-time encoding is designed to circumvent the non-unitarity issues presented in Chapter 4. This is done by an extended computational basis state encoding, which takes into account a larger part of the grid at each point in space. The downside of this approach is that the number of qubits required to encode the grid grows polynomially.

Chapter 6 summarizes our results and presents three open questions in the field of quantum algorithms for the Boltzmann equation that need to be answered in order to determine whether or not quantum computing can present a speed-up over classical methods for the Boltzmann method in certain use cases.

SAMENVATTING

Er bestaan problemen die kwantumcomputers efficiënter kunnen oplossen dan klassieke computers. Het is echter zo dat, door de specifieke eigenschappen van kwantumcomputers, deze niet voor alle type problemen geschikt zullen zijn. Een potentiële toepassing voor kwantumcomputers is de Lattice Boltzmann methode. De lattice Boltzmann methode is een techniek die gebruikt wordt in de numerieke stromingsleer, om vergelijkingen zoals de Navier-Stokes vergelijkingen op te lossen. Lattice Boltzmann methodes kunnen stromingen simuleren door het proces op te delen in twee delen, bestaande uit stromen en botsen. Die stromings- en botsingsstap kunnen vervolgens los van elkaar gesimuleerd worden. Dit betekent dat de methode natuurlijk op te delen is in een lineaire en een lokale stap, deze eigenschap is interessant voor kwantumcomputers aangezien zij van nature lineaire operaties kunnen uitvoeren en daarnaast lokale operaties kunnen paralleliseren. Het doel van dit proefschrift is om te bepalen hoe kwantumcomputers gebruikt kunnen worden om de lattice Boltzmann methode te implementeren en of zo'n implementatie een versnelling kan betekenen ten opzichte van klassieke methodes. In Hoofdstuk 1 geven we een korte inleiding over kwantumcomputers, numerieke stromingsleer, de Boltzmann methode en het nieuwe vakgebied van quantum numerieke stromingsleer.

Hoofdstuk 2 presenteert onze kwantumimplementatie van een lattice Boltzmann geïnspireerde techniek om de transport vergelijking op te lossen. De transport vergelijking is in principe gelijk aan de boltzmann-vergelijking zonder krachtterm waarvan tevens de botsingsterm is weggelaten. De transport vergelijking wordt soms de vlasov-vergelijking genoemd. Door de lineariteit van deze vergelijking kunnen wij hem efficiënt op een kwantumcomputer oplossen. In dit hoofdstuk presenteren we een efficiëntere manier om de stromingsfunctie te implementeren dan eerdere methodes gebruikt hebben. Onze implementatie van de stromingsfunctie is geïnspireerd door de Draper adder en maakt zodoende gebruik van de fouriertransformatie. Daarnaast presenteren we een efficiënte en correcte methode om de interactie tussen deeltjes en oppervlaktes van een object te simuleren.

In Hoofdstuk 3 presenteren we een metingsstrategie die gebruikt kan worden om de kracht die op een object wordt uitgeoefend door de stromende deeltjes te bepalen. De metingsstrategie is gebaseerd op de momentum uitwisselingsmethode en hebben wij daarom de quantum momentum uitwisselingsmethode genoemd. In bepaalde gevallen kan deze strategie gebruikt worden om efficiënt de kracht die op een object wordt uitgeoefend door de stromende deeltjes te bepalen.

Hoofdstuk 4 bekijkt in hoeverre het mogelijk is om het kwantumalgoritme voor de transport vergelijking uit te breiden zodat er een botsingsstap aan toegevoegd wordt

en het kwantumalgoritme voor de botsing stap uit te breiden zodat er een stromingsstap aan toegevoegd wordt. Hierbij merken wij eerst op dat kwantumalgoritmes voor de transportvergelijking gebruik maken van encoding technieken die gebaseerd zijn op de amplitudes van de kwantumstaten terwijl kwantumalgoritmes voor de botsingsstap gebruik maken van de basisvectoren encoding. We gebruiken deze realisatie om te laten zien dat het voor deze encoderingen niet mogelijk is om een botsing- of stromingsstap toe te voegen op een manier die de unitariteit van de kwantum operatie behoudt. De reden dat we geen botsingsstap kunnen toevoegen aan een amplitude gebaseerde encodering is dat de botsingsoperatie in deze encodering per definitie niet unitair kan zijn. Evenzo kunnen we geen stromingsstap toevoegen aan een basisvectoren encodering omdat stromen per definitie een niet lokale operatie is en de enige manier om de encodering op elk gridpunt na het stromen te bepalen is door informatie die uit alle omringende gridpunten komt te combineren.

In Hoofdstuk 5 presenteren we de zogenaamde space-time encodering. Space-time encodering is ontwikkeld om de problemen met unitariteit beschreven in het vorige hoofdstuk te voorkomen. Deze encodering is in wezen een uitbreiding van de basisvector encodering waarin het idee van lokaliteit wordt uitgebreid om alle gridpunten mee te nemen die in het totale aantal tijdstappen bereikt kunnen worden. Op deze manier groeit het aantal qubits dat nodig is om de gridpunten te beschrijven met een macht die gelijk is aan het aantal dimensies die we willen simuleren.

Hoofdstuk 6 bespreekt drie open problemen in het vakgebied van kwantum algoritmes voor de Boltzmann vergelijking die nog beantwoord moeten worden om te kunnen bepalen of quantum algoritmes een speed-up kunnen betekenen ten opzichte van klassieke methodes voor de lattice Boltzmann method.

1

INTRODUCTION

Quantum computing is a novel compute technology, with the promise of solving certain problems more efficiently than classical computers can. One question that remains is if quantum computers can be of help in solving problems of fluid dynamics. Whilst it is of great importance to many industries such as aviation, medicine and the automotive industry, certain types of fluid flow remain difficult to predict. The main bottleneck is computational power. As classical computers are becoming more and more powerful, more realistic fluid flow simulations can be performed, but with stagnation of Moore's law this increase will also grind to a halt without the development of novel algorithms. This leads to the question if new compute technologies, such as quantum computing, can be used to reach a speed-up in the field of computational fluid dynamics. Specifically we consider the Boltzmann method which, due to its characteristic where fluid flow is modeled by considering collision and streaming separately, makes an interesting candidate.

Parts of this chapter have been published in the Journal of Computational Physics, Quantum Information Processing, Computers and Fluids and the VKI Lecture notes [31, 33, 32]

1

1.1. GOALS OF THIS THESIS

Quantum computers are theoretically extremely powerful but specific compute devices. Currently there are some known theoretical advantages of quantum computers over classical computers, such as the algorithms presented in [34, 13]. Whilst quantum computing will prove advantageous for solving some problems, it will not prove beneficial for other types of problems. The goal of this thesis is to investigate how quantum computers can be used to solve the Boltzmann equation, and whether they can create some computational speed-up over classical computers in doing so. In order to achieve this we first present an algorithm designed for solving the transport equation. The transport equation is also referred to as the collisionless Boltzmann equation and as such this can be used in settings where for physical reasons the collision term can be neglected. We then extend our research by investigating ways in which we can add a collision term. The aim of this part of the thesis is two-fold, we first analyzed the commonly used data encoding techniques in quantum computing. We subsequently show that for these data encoding techniques there is no way to solve the full Boltzmann equation without restarting techniques, as this clashes with the unitarity constraints. We then propose a third data encoding technique for which we show that it is possible to perform a collision operation to reach a method that resembles the lattice gas models of the 1970's [15]. This implementation, unfortunately, comes at a qubit cost growing polynomially with the number of timesteps taken.

1.2. STRUCTURE OF THIS THESIS

We first give a general introduction to the fields of quantum computing and fluid dynamics, to the extent that is necessary to understand this thesis. We end the first chapter with an overview of related research that has been done in this field. Chapter 2 is based on the published paper [31] and shows how one can implement an algorithm for solving the transport equation. The next chapter, Chapter 3, is based on the paper [32] and provides an efficient method of reading out a quantity of interest from the resulting quantum state. In Chapter 4 we investigate the two main encoding methods known for quantum computational fluid dynamics to show that for both methods it is not possible to extend them to include both collision and streaming without breaking the unitarity requirement of quantum computers. In Chapter 5 we subsequently show that by looking at the proofs of why we cannot include a collision and streaming step in the existing encodings, one can find an encoding method that does allow a collision and streaming operation to take place. Both Chapter 4 and Chapter 5 are based on the paper [33]. In the conclusion 6 we summarize the results of this thesis and identify three open questions that remain to be answered in the field of quantum Boltzmann methods.

1.3. INTRODUCTION TO QUANTUM COMPUTING

Quantum theory was largely developed in the 1920s, whilst computers had their first big boom in the 1940s and 1950s. For many years these disciplines remained completely separated until the 1980s when Paul Benioff proposed the idea of what he called a quantum Turing machine [2, 1]. Around the same time Feynman [10] and Manin [24, 25] also suggested the idea of computers based on quantum theory, their ideas being motivated

by the exponential costs faced when modeling quantum systems on classical computing devices. From this moment on slowly but surely algorithms for these new compute devices were being developed. The most noteworthy of these algorithms is Shor's algorithm [34], which can be used to break the commonly used RSA encryption in polynomial time. This sparked a huge interest in the field of quantum algorithms and potential hardware from many stakeholders such as industry and government.

In this section we introduce the fundamentals of quantum computing necessary to understand quantum fluid dynamics for the non-expert. We will first introduce the mathematical theory behind quantum computing and in later sections we will discuss the practical elements of writing programs to run on physical quantum computers. In this thesis we only consider gate-based quantum computing, as such we will not give an introduction for a different type of computing or consider methods for different types of quantum computers. Note that this section is not meant to provide an extensive introduction to the entire field, for those interested in further reading we suggest [25, 37].

1.3.1. QUBITS

Quantum computers are built up using the quantum counterparts of classical bits, namely quantum bits or qubits. Quantum computers make use of the quantum mechanical properties of nature, as such qubits have different properties than their classical counterparts and different operations can be applied to them. Where a bit can be either 0 or 1, a qubit can hold a complex linear combination of basis states. First of all, we express the states of a qubit using Dirac's braket notation. As such the quantum version of the state 0 will be written as $|0\rangle$ and of the state 1 as $|1\rangle$. More importantly, however, quantum states are not restricted to being either in the state $|0\rangle$ or $|1\rangle$, but can be in a so-called superposition of the two. Mathematically this superposition can be thought of as a complex linear combination, or a unit vector in complex space. A single qubit can hold any value of the form

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle, \quad \alpha_0, \alpha_1 \in \mathbb{C}, \quad |\alpha_0|^2 + |\alpha_1|^2 = 1, \tag{1.1}$$

whereby the α_i are referred to as the probability amplitudes.

We can interpret the computational basis states $|0\rangle$ and $|1\rangle$ as vectors in a 2-dimensional Hilbert space, this gives

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle = \alpha_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix}, \tag{1.2}$$

where $|0\rangle$ and $|1\rangle$ represent the computational basis states and α_0 and α_1 are the coefficients.

When describing a two qubit quantum state, we take the so-called Kronecker product between the two single qubit states.

$$
\begin{aligned}
|\psi_0\rangle |\psi_1\rangle &= (\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes \left(\beta_0 |0\rangle + \beta_1 |1\rangle\right) \\
&= \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle \\
&= \gamma_0 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \gamma_1 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \gamma_2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \gamma_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix},
\end{aligned} \tag{1.3}
$$

where we again have that $\gamma_i \in \mathbb{C}$ and $\sum_i |\gamma_i|^2 = 1$ hold.

We can simplify this notation by noting that we can replace the binary expressions of the qubit state by their decimal values. This leads to the expression

$$
\begin{aligned}
|\psi_0\rangle |\psi_1\rangle &= \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle \\
&= \alpha_0\beta_0 |0\rangle + \alpha_0\beta_1 |1\rangle + \alpha_1\beta_0 |2\rangle + \alpha_1\beta_1 |3\rangle \\
&= \sum_{k=0}^{3} \gamma_k |k\rangle .
\end{aligned} \tag{1.4}
$$

Adding a qubit results in the total space being doubled when the Kronecker product is taken. This means that for an n qubit quantum state a vector of size 2^n is required to express it. Another way to interpret this is by noting that an n qubit state lives in a 2^n dimensional complex vector space. This large dimensionality results in one of the main powers of quantum computing. When properly exploited, we can work in an exponentially large space compared to classical compute technologies with n bits.

We write an n qubit quantum state as

$$
|\psi\rangle = \sum_{b\in\{0,1\}^n} \alpha_b |b\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle , \quad \alpha_b, \alpha_k \in \mathbb{C}, \quad \sum_{b\in\{0,1\}^n} |\alpha_b|^2 = \sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1. \tag{1.5}
$$

As before an n qubit state $|\psi\rangle$ can be interpreted as a 2^n dimensional complex unit vector.

1.3.2. MEASUREMENT

As stated above, an n qubit quantum system can be interpreted as a complex 2^n dimensional unit vector. This implies that we can work in an exponentially large space, but there is a very important practical caveat related to measurement. When measuring a single qubit in a basis we do not extract the full quantum state, we only find one of the two possible basis states that the qubit is in. The probability of finding each basis state is determined by its amplitudes. For instance, consider the state we saw before

$$
|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle . \tag{1.6}
$$

The probability of finding $|0\rangle$ upon measurement in the computational basis is $|\alpha_0|^2$, similarly the probability of finding $|1\rangle$ is $|\alpha_1|^2$. Even though the quantum state itself can have an infinite amount of information encoded in it, this measurement principle implies that we cannot easily retrieve this information.

With multiple qubits the principles of measurement in a basis remain the same. Consider the n qubit state

$$|\psi\rangle = \sum_{b\in\{0,1\}^n} \alpha_b |b\rangle , \tag{1.7}$$

then upon measurement finding the state $|b\rangle$ has a probability of $|\alpha_b|^2$. This shows that even though you can work in an exponentially large space, you cannot retrieve the total state and the larger the space the more measurements are required to get a good description of the total quantum state. This implies that a good quantum algorithm should not rely on reading out a very large and spread out quantum state towards the end.

When measuring a quantum state and finding a certain basis state the quantum system collapses to that state. Consider the example above, the n qubit state as expressed in Equation (1.7). Upon measurement one of the basis states is found with a probability as expressed by the amplitudes. Then, after measurement, the quantum state has changed. If, upon measurement, the quantum system is found to be in the state $|l\rangle$, then from that moment on the system becomes

$$\frac{\alpha_l}{|\alpha_l|} |l\rangle . \tag{1.8}$$

Another way to express this is is that upon measurement the superposition collapses onto a single basis state.

It is also possible to measure just one qubit of a multiple qubit system. Consider the state as presented in (1.7) measuring the l-th qubit and finding it in the state $|0\rangle$ ($|1\rangle$) amounts to the system ψ collapsing to the states for which the l-th digit is equal to 0 (1) in its binary expression. That means that after measurement the state collapses to

$$\frac{1}{\sum_{b\in C} |\alpha_b^2|} \sum_{b\in C} \alpha_b |b\rangle , \tag{1.9}$$

where $C \subset \{0,1\}^n$ s.t. $\forall b \in C$ the l-th bit is equal to 0 (1). This is related to the concept of entanglement as further explained below.

ENTANGLEMENT

Entanglement refers to the states of two separate qubits somehow being dependent on each other. Consider the two qubit quantum state

$$|\psi\rangle = \sum_{k=0}^{3} \alpha_k |k\rangle . \tag{1.10}$$

Now consider that the only states with nonzero amplitude are $|00\rangle$ and $|11\rangle$ with amplitude $\frac{1}{\sqrt{2}}$, this gives the Bell state

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) , \tag{1.11}$$

which is a typical example of a state where the qubits are entangled. Consider the first qubit. We can see from the state that whenever the first qubit is in the state $|i\rangle$ with

$i \in \{0,1\}$ the second qubit must also be in the state $|i\rangle$. This property of the Bell state becomes most interesting upon measurement. Consider the Bell state and consider measuring only the first qubit. If the first qubit is measured to be in the state $|0\rangle$, the system collapses to that state. Therefore the part of the quantum state where the first qubit was $|1\rangle$ no longer exists. Since the two qubits were entangled, that means that now the second qubit must also be in the state $|0\rangle$. This is what entanglement means, the measurement of one qubit in a particular state determines the state of the other qubit.

SUPERPOSITION

Superposition is another principle inherent to quantum computing. Superposition is the term that describes that a qubit, unlike a classical bit, does not need to be in the state $|0\rangle$ or $|1\rangle$ but can be in this complex linear combination of the two.

Consider a quantum state

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle . \tag{1.12}$$

This state $|\psi\rangle$ is in a so-called superposition of the $N = 2^n$ computational basis states $|k\rangle$.

1.3.3. QUANTUM COMPUTATION AND GATES

In order to perform some sort of computation with the qubits, we need to be able to manipulate them. Mathematically, any unitary operation can be used to alter the quantum state. Consider the state

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle . \tag{1.13}$$

Applying a unitary $U \in \mathbb{C}^{2^n \times 2^n}$ to this state can be written as

$$U|\psi\rangle = U \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} U|k\rangle . \tag{1.14}$$

In the above equation we make use of the linearity of quantum computing. Which means that we can apply the operation U to all basis states in a single operation. The ability to apply an operation to multiple basis states at once is sometimes referred to as quantum parallelism.

Such a unitary operation U could be interpreted as a quantum circuit. In order for a quantum circuit to be implementable on a quantum computer we must express it in terms of quantum gates. Quantum gates are small logical operators usually applied to one to three qubits for which the direct implementation on a quantum computer is known.

1.3.4. PRACTICAL QUANTUM COMPUTING

Current and future quantum computers are somewhat different from the theoretical concepts of quantum computing we have described above. There are several key differences.

The first key difference is the existence of noise and decoherence. Quantum computers are inherently noisy, meaning that the state will not remain stable for long periods of time and the gate operations performed on them might slightly differ from their

mathematical expression. In practice it is therefore not possible to perform reliable computations that consist of many quantum gates on current quantum technologies often referred to as Noisy Intermediate Scale Quantum (NISQ) devices. Limiting the noise of quantum computers and coming up with methods for stable quantum computing is a large and active area of research [7] that goes beyond the scope of this thesis. In this thesis we will assume to be working with fault tolerant quantum computers[1].

Another important practical caveat is that of compiling. When describing quantum computing in terms of linear algebra, a suitably sized unitary matrix does not suffice for a practically implementable quantum operation. In practice we need to decompose the quantum operation into so-called native gates. Native gates are those directly implementable on the quantum hardware. Most quantum computers have a similar set of native gates, usually consisting of one or two two-qubit gates and multiple single-qubit gates [9, 17]. Bringing this back to the interpretation of linear algebra a two-qubit gate can be seen as a unitary matrix $U \in \mathbb{C}^{4x4}$ and a single qubit gate as a unitary matrix $U \in \mathbb{C}^{2x2}$. An example of a native gate set is {CZ, ID, RZ, RZZ, SX, X}, this is the native gate set of the IBM Fez, IBM's largest quantum computer to date (as seen on the IBM quantum website on November 5-th 2024). The matrix expressions for these gates are

$$\mathrm{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \tag{1.15}$$

$$\mathrm{RZZ} = \begin{bmatrix} e^{\frac{-i\lambda}{2}} & 0 & 0 & 0 \\ 0 & e^{\frac{i\lambda}{2}} & 0 & 0 \\ 0 & 0 & e^{\frac{i\lambda}{2}} & 0 \\ 0 & 0 & 0 & e^{\frac{-i\lambda}{2}} \end{bmatrix}, \tag{1.16}$$

$$\mathrm{ID} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{1.17}$$

$$\mathrm{RZ}(\lambda) = \begin{bmatrix} e^{\frac{-i\lambda}{2}} & 0 \\ 0 & e^{\frac{i\lambda}{2}} \end{bmatrix}, \tag{1.18}$$

$$\mathrm{SX}(\lambda) = \frac{1}{2}\begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}, \tag{1.19}$$

$$\mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{1.20}$$

Any other matrix would have to be decomposed in terms of these before being practically implementable on the quantum computer. There is no non-exponential algorithm to compile an arbitrary unitary matrix in terms of any finite native gate set [25]. This means that even though theoretically quantum computing allows for any unitary operation to be applied to the qubits, this is not always efficiently possible. Therefore, when

[1] Fault tolerant quantum computers refer to quantum hardware that are able to perform reliable computations involving of a sizable number of qubits and quantum gates.

developing quantum algorithms, it does not suffice to show that a unitary operation can calculate what is required. One must also provide a quantum algorithm expressed in terms of a native gates, or gates for which an efficient decomposition is known in order to convincingly provide an efficient quantum algorithm. A similar challenge is encountered when it comes to state initialization. Even though any unit vector of size 2^n can be expressed as an n-qubit quantum state, there might not exist an efficient sequence of native gates to prepare such a state on a quantum computer. Therefore, in this thesis, we will only consider something an efficient quantum algorithm if it only contains building blocks consisting of native gates or for which a polynomial native gate decomposition is known. This is to ensure that the algorithms complexity can be reliably determined and that the algorithm can be efficiently implemented.

1.4. INTRODUCTION TO FLUID DYNAMICS AND THE LATTICE BOLTZMANN METHOD

Fluid dynamics is a branch of physics that describes the behavior of fluids over time. The equations that describe fluid mechanics typically do not have a known analytical solution, which led to the modern field of computational fluid dynamics where such equations are solved numerically. A large part of the information in this section is based on the book [20] which is recommended to the reader as a more extensive introduction to the lattice Boltzmann method.

1.4.1. NAVIER-STOKES

The Navier-Stokes equations can be used to model a Newtonian fluid[2]. The Navier-Stokes equations describe the behavior of fluids on the macroscopic scale, which means they describe it in terms of density, pressure and velocity. Assuming the density ρ is constant, the incompressible Navier-Stokes equation arises and is given by

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \eta\Delta\boldsymbol{u} + \boldsymbol{F}, \tag{1.21}$$

where t is time, $\boldsymbol{u}$ fluid velocity, η shear viscosity and $\boldsymbol{F}$ the external forces acting on the fluid. As the density ρ is constant the continuity equation reduces to

$$\nabla\cdot\boldsymbol{u} = 0, \tag{1.22}$$

which describes mass conservation. Assuming that the density is known, the equations contain four unknowns, namely the pressure p and the three velocity components.

There are several popular methods with which these equations can be solved numerically, such as Finite Difference, Finite Volume and Finite Element Methods. These methods all have their own pros and cons in terms of implementability and precision [20].

1.4.2. LATTICE GAS AUTOMATA

We can also model the behavior of fluids by considering the particles rather than the macroscopic phenomena. One such way to do that is via so-called lattice gas automata.

[2] A Newtonian fluid is a fluid

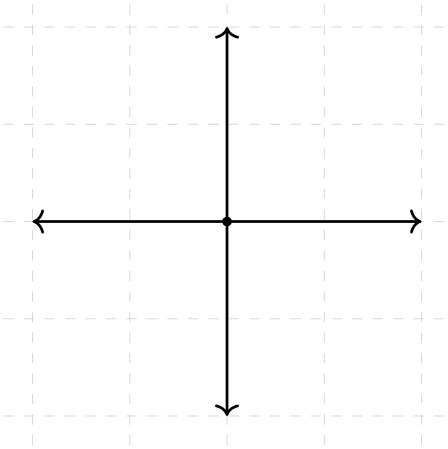
The first example of a lattice gas automaton was the HPP model introduced by Hardy, Pomeau and de Pazzis in 1973 [15]. The HPP model they developed modeled fluid flow on a two-dimensional domain by allowing separate particles to flow in 4 different directions [15, 14]. The idea is that each point in space can have a maximum of 4 particles, where each particle travels in the positive or negative x or y direction with speed equal to one. It is important to note that no two particles at the same grid point can have the same velocity. The velocity distribution at a specific point in space is subsequently labeled using a binary label indicating whether or not a particle is present in each position. We define $n_i(\mathbf{x}, t)$ to be the occupation number of the particle at position in space $\mathbf{x}$ at time t with velocity $\boldsymbol{c}_i$. As we restrict our model to a maximum of one particle with a specific velocity $\boldsymbol{c}_i$ at each point in space and time we have $n_i(\boldsymbol{x}, t) \in \{0, 1\}$. We can represent the particles being present at a given point in space $\boldsymbol{x}$ with velocity i using a binary variable n_i, where 0 indicates no particle with velocity i is present and 1 indicates that such a particle is present.

The velocity vectors are given by

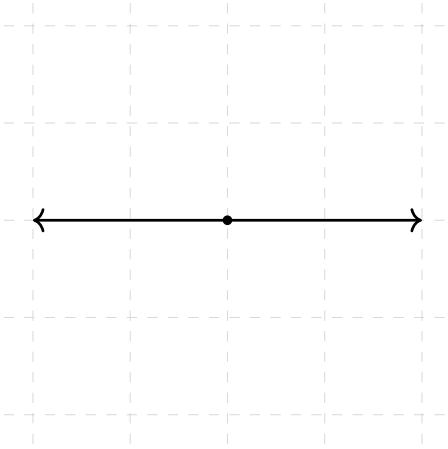
$$\boldsymbol{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{c}_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \boldsymbol{c}_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}. \tag{1.23}$$

We can order the occupation numbers as $n_4 n_3 n_2 n_1$. These occupation numbers can be combined with a pictorial representation as shown in Figure 1.1.

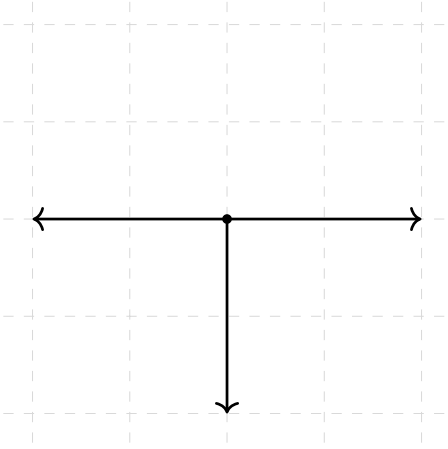
In each time step the particles stream through space based on their velocity. If after streaming, multiple particles end up in the same grid point some sort of collision operation will take place. For the very simple two dimensional example with four possible velocities we are currently considering, a collision operation will only take place if two particles are present that stream in opposite directions. In this case their velocities are rotated with an angle $\frac{\pi}{2}$; see Figure 1.2. This operation models a collision operation which conserves mass and momentum as is physically required. Even though this model can simulate an equation that has some similarities to the Navier-Stokes equations, it does not yet solve them. This is due to the equations this model simulates not being invariant under global spatial rotations [4].



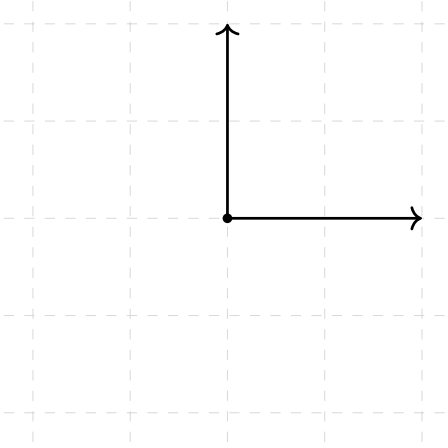
(a) Example of 1111 particle distribution, where four particles are present at the center grid point each flowing in a different direction.



(b) Example of 0101 particle distribution, where two particles are present in the center grid point. One particle is flowing in the positive and one particle is flowing in the negative direction on the x-axis.

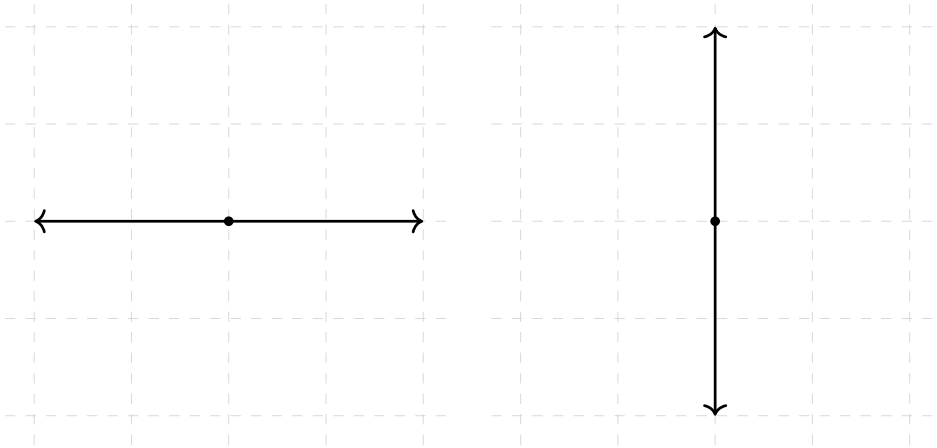


(c) Example of 1101 particle distribution, where three particles are present in the center grid point. One particle is flowing in the negative direction on the y-axis and one particle is flowing in the positive and one particle is flowing in the negative direction on the x-axis.



(d) Example of 0011 particle distribution, where two particles are present in the center grid point. One particle is flowing in the positive direction on the x-axis and one particle is flowing in the positive direction on the y-axis.

Figure 1.1: Different configurations of velocities present at a point in space $\boldsymbol{x}$ for the HPP model.



(a) Example of 0101 particle distribution, where two particles are present in the center grid point. One particle is flowing in the positive and one particle is flowing in the negative direction on the x-axis. This picture represents the state of the system before the described collision has taken place.

(b) Example of 1010 particle distribution, where two particles are present in the center grid point. One particle is flowing in the positive and one particle is flowing in the negative direction on the y-axis. This picture represents the state of the system after the described collision has taken place.

Figure 1.2: In the first HPP model collision is modeled by mapping the 1010 distribution to 0101 and vice versa.

Later, lattice gas models with a hexagonal grid were developed for two dimensions [11]. This model does preserve the isotropy[3] required for the Navier-Stokes equations [4]. Around the same time a three dimensional lattice gas automata was proposed. Since there is no clear extension of a hexagonal grid to three dimensions this model uses a cube shaped grid [8]. Having more possible streaming directions also means more interactions are possible when considering a collision. This is usually modeled using a large look-up table which maps possible input states to possible output states. As it is difficult to give a simple expression to represent the collision operation, we simply write it as $\Omega_i(\boldsymbol{x}, t)$.

Algorithm 1 Lattice gas automata

while $t \leq T$ **do**
  Perform collision: $n_i^*(\mathbf{x}, t) = n_i(\mathbf{x}, t) + \Omega_i$
  Perform streaming: $n_i(\mathbf{x} + \Delta t \boldsymbol{c}_i, t + \Delta t) = n_i^*(\boldsymbol{x}, t)$
  Update time: $t = t + \Delta t$
end while

1.4.3. THE BOLTZMANN EQUATION

With the growth of the power of the available computational devices, slowly the research into Lattice Gas Automata has grinded to a halt and has been replaced by lattice Boltzmann methods. The idea of lattice Boltzmann methods naturally flows from Lattice Gas

[3] Isotropy refers to being invariant under rotation.

Automata, only a more complicated function is used to model the collision operation. This collision operation is derived from the Boltzmann equation. Much like the Navier-Stokes equations the Boltzmann equation can be used to model fluid flows, only instead of considering the macroscopic scale the Boltzmann equation describes the behavior on the so-called mesoscopic scale. The mesoscopic scale lies between the microscopic scale, where separate particles are considered and the macroscopic scale, where measurable physical phenomena are considered and the continuum assumption holds. The Boltzmann equation instead considers the behavior of particle densities over space in time.

The Boltzmann equation can be derived by considering the function $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$, which describes the relative density at the position $\boldsymbol{x}$ of particles with velocity $\boldsymbol{\xi}$ at time t. In order to describe the behavior of fluid flow over time we want to find the equation that expresses how this function changes over time. Taking the derivative of f over time and writing $\frac{df}{dt}$ as the collision operator $\Omega(f)$ leads to the Boltzmann equation

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi}\frac{\partial f}{\partial \boldsymbol{x}} + \frac{\boldsymbol{F}}{\rho}\frac{\partial f}{\partial \boldsymbol{\xi}} = \Omega(f). \tag{1.24}$$

In the above $\boldsymbol{F}$ represents the external force exerted on the particles and ρ represents the particle density.

For computational efficiency Bhatnagar, Gross and Krook [3] suggested replacing the collision operator by the BGK collision operator

$$\Omega(f) = -\frac{f - f^{\text{eq}}}{\tau}, \tag{1.25}$$

where f^{eq} represents the equilibrium distribution function and τ the relaxation time. The equilibrium distribution function, often referred to as the Maxwell-Boltzmann distribution, can be given by

$$f^{\text{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, t) = \rho\left(\frac{1}{2\pi RT}\right)^{\frac{3}{2}} e^{\frac{-|\boldsymbol{\xi}-\boldsymbol{u}|^2}{2RT}}, \tag{1.26}$$

note that ρ, $\boldsymbol{u}$ and T can be calculated using the moments of the Boltzmann equation [20]

$$\rho(\mathbf{x}, t) = \int f(\mathbf{x}, \boldsymbol{\xi}, t)\, d^3\boldsymbol{\xi}, \tag{1.27}$$

$$\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) = \int \boldsymbol{\xi} f(\mathbf{x}, \boldsymbol{\xi}, t)\, d^3\boldsymbol{\xi}, \tag{1.28}$$

$$\rho(\mathbf{x}, t)\, e(\mathbf{x}, t) = \frac{1}{2}\int |\boldsymbol{\xi} - \boldsymbol{u}|^2 f(\mathbf{x}, \boldsymbol{\xi}, t)\, d^3\boldsymbol{\xi}, \tag{1.29}$$

in combination with the equation for the specific internal energy density for three dimensions

$$e = \frac{3}{2}RT. \tag{1.30}$$

In the above ρ represents the particle density, $\boldsymbol{u}$ the fluid velocity and e the internal energy.

Combining Equation (1.25) and (1.24) leads to

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi}\frac{\partial f}{\partial \boldsymbol{x}} + \frac{\boldsymbol{F}}{\rho}\frac{\partial f}{\partial \boldsymbol{\xi}} = -\frac{f - f^{\mathrm{eq}}}{\tau}, \tag{1.31}$$

as a commonly used expression for the Boltzmann equation. In this thesis we consider only cases without a force term F, which leaves

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi}\frac{\partial f}{\partial \boldsymbol{x}} = -\frac{f - f^{\mathrm{eq}}}{\tau}. \tag{1.32}$$

In Chapter 2 of this thesis we consider the transport equation

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi}\frac{\partial f}{\partial \boldsymbol{x}} = 0, \tag{1.33}$$

which is sometimes referred to as the collisionless Boltzmann equation [36] as it is equal to the forceless Boltzmann equation without a collision term.

In order to efficiently solve the forceless Boltzmann equation on a computer it is first discretized into the discrete Boltzmann equation. The discrete Boltzmann equation considers the relative particle density as before, but the input variables $\boldsymbol{x}$, $\boldsymbol{\xi}$ and t are discretized. Discretizing in space means we are considering a grid on which the relative particle densities move. Therefore instead of being able to move anywhere in space, they now can only move from grid point to grid point similar to the particles in the lattice gas automata described in Section 1.4.2. When discretizing in time it is important to choose timesteps that are small enough such that none of the particles can overshoot a grid point. In Chapter 2 we explain exactly how the timesteps can be chosen using the so-called CFL counter. Finally we discretize the velocity space. This means creating a finite set Ξ of possible velocities the particles can have. A final change when working with the discrete Boltzmann equation is that the considered function changes. Instead of considering $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ we now consider $f_i(\boldsymbol{x}, t)$ where the subscript i represents the velocity $i \in \Xi$. This leads to the discrete Boltzmann equation without a force function as derived in [20]

$$f_i(\boldsymbol{x} + \boldsymbol{c_i}\Delta t, t + \Delta t) - f_i(\boldsymbol{x}, t) = -\frac{\Delta t}{\tau}\left(f_i(\boldsymbol{x}, t) - f_i^{\mathrm{eq}}(\boldsymbol{x}, t)\right). \tag{1.34}$$

Note that here we have also included a discretized version of the equilibrium function which is given by

$$f_i^{\mathrm{eq}}(\boldsymbol{x}, t) = w_i \rho\left(1 + \frac{\boldsymbol{u}\cdot\boldsymbol{c}_i}{c_s^2} + \frac{(\boldsymbol{u}\cdot\boldsymbol{c}_i)^2}{2c_s^4} - \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{2c_s^2}\right), \tag{1.35}$$

where w_i are the weights[4], ρ is the density and $\boldsymbol{u}$ represents the fluid velocity. The density and fluid velocity can be calculated by

$$\rho(\boldsymbol{x}, t) = \sum_i f_i(\boldsymbol{x}, t), \tag{1.36}$$

[4] These weights are pre-determined depending on the dimensions and possible velocities considered. They can be found in various sources such as page 88 of [20].

$$\rho(\boldsymbol{x},t)\,\boldsymbol{u}(\boldsymbol{x},t)=\sum_i \boldsymbol{c}_i f_i(\boldsymbol{x},t)\,. \tag{1.37}$$

One interesting feature of the discrete Boltzmann equation is that it can naturally be broken up in two distinct steps. The first step is to set

$$f_i^*(\boldsymbol{x},t)=f_i-\frac{\Delta t}{\tau}\left(f_i(\boldsymbol{x},t)-f_i^{\mathrm{eq}}(\boldsymbol{x},t,)\right), \tag{1.38}$$

which represents the collision operation, where the state of $f_i(\boldsymbol{x},t)$ is altered by the collision term. The second step is given by

$$f_i(\boldsymbol{x}+\boldsymbol{c}_i\Delta t,t+\Delta t)=f_i^*(\boldsymbol{x},t)\,, \tag{1.39}$$

which represents the streaming operation, where the relative densities are moved through space depending on the velocity of the particles and the time step taken. This natural way to split the discrete Boltzmann equation up in a streaming and collision step means that we can implement it by using one non-local but linear step (streaming) and one non-linear but local step (collision).

This discretized version of the Boltzmann equation is typically used in lattice Boltzmann methods to model fluid dynamics. The idea of the workflow is similar to that of lattice gas automata, where on a discretized grid particles can flow with a finite number of possible velocities, and after streaming is performed for one time-step, a collision operation takes place. Since we are now working with relative particle densities we are no longer required to only model a binary version of collision but we can use Equation (1.38), which allows for a version of collision where some relaxation towards equilibrium is actually taking place.

Algorithm 2 Lattice Boltzmann Method

1: **while** $t \le T$ **do**
2: Compute fluid density: $\rho(\boldsymbol{x},t) \leftarrow \sum_i f_i(\boldsymbol{x},t)$
3: Compute fluid velocity: $\boldsymbol{u}(\boldsymbol{x},t) \leftarrow \frac{1}{\rho(\boldsymbol{x},t)}\sum_i \boldsymbol{c}_i f_i(\boldsymbol{x},t)$
4: Compute equilibrium function: $f_i^{\mathrm{eq}}(\boldsymbol{x},t) \leftarrow w_i\rho\left(1+\frac{\boldsymbol{u}\cdot\boldsymbol{c}_i}{c_s^2}+\frac{(\boldsymbol{u}\cdot\boldsymbol{c}_i)^2}{2c_s^4}-\frac{\boldsymbol{u}\cdot\boldsymbol{u}}{2c_s^2}\right)$
5: Perform collision: $f_i^*(\boldsymbol{x},t) \leftarrow f_i(\boldsymbol{x},t)-\frac{\Delta t}{\tau}\left(f_i(\boldsymbol{x},t)-f_i^{\mathrm{eq}}(\boldsymbol{x},t)\right)$
6: Perform streaming: $f_i(\boldsymbol{x}+\boldsymbol{c}_i\Delta t,t+\Delta t) \leftarrow f_i^*(\boldsymbol{x},t)$
7: Update time: $t = t+\Delta t$
8: **end while**

In the discretized Boltzmann method it is common to adopt the DdQq scheme for denoting dimension and velocity. For example, D2Q5 denotes a method with two dimensions and five different velocity directions.

1.5. INTRODUCTION TO QUANTUM COMPUTATIONAL FLUID DYNAMICS

With the dawn of quantum computers and the computational complexity of computational fluid dynamics, the question arises if quantum computers can be of any use in

solving the problems of fluid dynamics. Since the 1990s a variety of ideas of how to solve the problems of computational fluid dynamics on a quantum computer has been researched.

The first quantum computational fluid dynamics (QCFD) algorithms were proposed by Yepez around the turn of the century [38, 39, 41, 40, 28], these algorithms contain a quantum distributed computing approach based on the lattice gas models as presented in [11]. The idea is that each grid point of position-space gets its own 6 qubit quantum computer associated to it (note that here a 600 qubit quantum computer could encode 100 such separate computers). The benefit of this approach is that the possible quantum circuit depth and stable entanglement required remains very low, making it a realistic and relatively near-term approach given the power of current devices. The downside of this approach is that to encode a grid of size N a total of $6N$ qubits are required, which means that the number of qubits required grows linearly with the number of grid points. Given the limited number of quantum devices available and large number of grid points considered in modern Boltzmann methods [35], this proves a significant drawback. Furthermore, as we will prove in this thesis, their computational basis state encoding of the velocity vector does not allow for streaming on a quantum computer, therefore after each timestep a measurement and re-initialization of the quantum system needs to take place. After these results were reached by Yepez et al. the QCFD field became stagnant for over a decade. Recently, however, there has been a resurgence in the field of QCFD and in particular there has been a rise in quantum Boltzmann methods.

In 2020 Gaitan published a quantum algorithm that can be used to solve the Navier-Stokes equations [12]. In this article the author shows that there is a quantum speed-up for non-smooth flows and identifies a regime where the speed-up is quadratic over classical random algorithms. Later, in 2021, Oz et al. adopted the algorithm presented in [12] for solving partial differential equations to Burgers' equation [26]. In this work the authors present solutions for flow problems with and without shock waves thereby achieving similar speed-up as for the Navier-Stokes equations.

A hybrid quantum classical reservoir computing model was suggested by Pfeffer et al. in 2022 [27]. With their method the authors are able to simulate nonlinear chaotic dynamics of Lorenz type models. They show that by using just a few highly entangled qubits they can achieve similar prediction and reconstruction capabilities as classical reservoirs using thousands of perceptrons.

Another interesting approach that targets noisy intermediate quantum computers was proposed by Kyriienko et al. in [21]. In essence, a quantum neural network is trained to learn the solution values of a partial differential equation complemented by boundary and possibly initial values. The classical counterpart of this concept has become popular in recent years under the name physics-informed machine learning.

In 2021, Liu et al. presented a quantum algorithm for solving non-linear differential equations [22]. The authors suggest to use Carleman linearization and subsequently perform time integration by the forward Euler method in combination with the quantum linear system algorithm by Harrow et al. [16] to find a solution.

Todorova and Steijl proposed a quantum algorithm for what they call the collisionless Boltzmann equation, where they propose quantum primitives for the streaming and specular reflection step [36].

In the same year, Budinski suggested a quantum lattice Boltzmann method for a one-dimensional and two-dimensional lattice structure that does include a simplified collision term, but does not model the specular reflection step [5]. The collision term is realized using the linear approximation of unitary approach [23], which requires a measurement at the end of each time step to check whether the previous computations have been meaningful or ended up in a so-called 'orthogonal state of no interest'. In the former case, the algorithm can proceed to the next time step. In the latter case the simulation must be restarted. Post-selection of valid results after (partial) measurement as used in the approximation of unitary approach is a common practice in quantum algorithms but it becomes problematic if used repeatedly within a time-stepping loop. It is obvious that the probability of failure increases in a multiplicative manner with the number of timesteps. Therefore the number of timesteps that can be feasibly modeled using this approach highly depends on the amplitudes of the 'orthogonal states of no interest' and is not realistically implementable for an interesting number of timesteps , unless the probability of failure can be shown to be very small. Unfortunately, the paper [5] does not provide any bound on this probability.

In the subsequent paper [6], Budinski proposed an implementation of the specular reflection step using the quantum linear approximation of unitary approach, which brings the same problems as before. Furthermore, the paper does not provide a clear procedure for decomposing the required unitary into quantum gates for which adecomposition into native gates is known, which hinders straightforward implementation.

Another approach to simulate fluid flow using a quantum computer is by making use of Carlemann linearization. The theoretical set-up for this was presented in [18] and the subsequent papers [29, 30] make use of the Carlemann linearization method to linearize the Boltzmann and other equations and subsequently solve them on a quantum computer. In [29] the authors linearize the Boltzmann equation and solve the subsequent linear system on a quantum computer. The method works, however is not yet ready for large scale use due to the necessary decomposition of a large unitary in the process for which no efficient decomposition has been found. In [30] the authors use Carleman linearization on Grad's generalized hydrodynamics, which is another method that can be used for simulating fluid. Here they end up with a linear system for which, due to its properties, no efficient quantum circuit can be set up to solve it according to the current state of the art. Both these methods do show some interesting structural properties of the resulting matrix, so further investigation is required to determine the future feasibility of this approach.

In the spirit of the lattice gas model presented by Yepez, the paper [42] presents its own versions of a lattice gas model. They present two possible implementations, one superposition-based and one binary-based, both requiring a stop-and-go set-up to circumvent the non-unitarity issues as presented in [33]. Two other lattice gas inspired models are presented in [19]. Here the authors present one method that can be run on a quantum computer without requiring a stop-and-go method by making use of Quantum Phase Estimation. This method requires a number of qubits that scales linearly with the number of velocities required and grid points.

In this thesis we present two novel quantum lattice Boltzmann methods. First, in Chapter 2, we present a quantum algorithm for the collisionless lattice Boltzmann method,

also known as the transport equation, as published in [31]. In Chapter 3 we introduce an efficient measurement method for the collisionless quantum lattice Boltzmann method using the Quantum Momentum Exchange Method. Subsequently, in Chapter 4, we present proofs that the commonly used encoding methods for quantum Boltzmann do not allow streaming and collision to both take place as a unitary operation. This implies that novel encoding methods are required to achieve a full quantum Boltzmann algorithm that does not require a stop-and-go strategy. Here we use the term stop-and-go strategy to refer to methods that require measurement between timesteps. One reason such a measurement might be required is that the operation is not unitary and is therefore applied to a larger space and some post selection must take place. Another reason for this is that some information encoded in the qubits is required for the next timestep, and therefore some measurement and reinitialization is performed. In Chapter 5 we present space-time encoding, which was specifically developed to circumvent the issues in the commonly used encoding methods which caused streaming and collision to not be possible as a unitary operation. Using our space-time encoding we are able to simulate multiple timesteps, whilst having the full operation be a unitary. Furthermore, the operation for our space time encoding is easily and inexpensively expressed as quantum gates without any decomposing of unitaries into quantum gates necessary.

The collisionless quantum Boltzmann method presented in this thesis improves on the aforementioned quantum Boltzmann methods in various ways. Compared to [5, 6], our approach enables a higher variety in the particle velocities. Moreover, our realization of the quantum specular reflection step is based on simple one- and two-qubit quantum gates, without the need to adopt the troublesome linear approximation of unitaries approach. This also allows us to ensure the straightforward implementability of our approach into quantum programming frameworks like Qiskit, without the need to decompose unknown unitaries or multi-control operations into native quantum gates.

Our quantum space-time encoding marks the first quantum algorithm that can perform multiple timesteps of the lattice Boltzmann method as a unitary operation without the requirement of intermediate restarts. On top of that it sets itself apart from other quantum methods which allow for collision to take place in that it does not require the decomposition of large unitary matrices into possibly exponentially many native gates. This makes our method much more realistic to be implemented on a quantum computer, as the decomposition of a unitary operation can be exponentially expensive.

BIBLIOGRAPHY

[1] Paul Benioff. "Quantum Mechanical Hamiltonian Models of Turing Machines". In: *Journal of Statistical Physics* (1982). DOI: https://doi.org/10.1007/BF01342185.

[2] Paul Benioff. "The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines". In: *Journal of Statistical Physics* (1980). DOI: https://doi.org/10.1007/BF01011339.

[3] P. L. Bhatnagar, E. P. Gross, and M. Krook. "A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems". In: *Phys. Rev.* 94 (3 May 1954), pp. 511–525. DOI: 10.1103/PhysRev.94.511. URL: https://link.aps.org/doi/10.1103/PhysRev.94.511.

[4] Bruce M. Boghosian. "Lattice gases and cellular automata". In: *Future Generation Computer Systems* (1999). URL: https://doi.org/10.1016/S0167-739X(99)00045-X.

[5] Ljubomir Budinski. "Quantum algorithm for the advection-diffusion equation simulated with the lattice Boltzmann method". In: *Quantum Information Processing 2021* (2020). URL: https://link.springer.com/article/10.1007/s11128-021-02996-3.

[6] Ljubomir Budinski. "Quantum algorithm for the Navier-Stokes equations by using the streamfunction-vorticity formulation and the lattice Boltzmann method". In: *International Journal of Quantum information* (2021). URL: https://arxiv.org/abs/2103.03804.

[7] Earl Campbell. "A series of fast-paced advances in Quantum Error Correction". In: *Nature Reviews Physics* (2023). URL: https://doi.org/10.1038/s42254-024-00706-3.

[8] D. d'Humières, P. Lallemand, and U. Frisch. "Lattice Gas Models for 3D Hydrodynamics". In: *Europhysics letters* 2 (4 1986), pp. 291–297. DOI: 10.1209/0295-5075/2/4/006.

[9] G.R. Di Carlo and L. DiCarlo. "Quantum Inspire Starmon-5 Fact Sheet". In: (2023). URL: https://qutech.nl/wp-content/uploads/2024/03/Starmon5_FactSheet_240316.pdf.

[10] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* (1982). DOI: https://doi.org/10.1007/BF02650179.

[11] U. Frisch, B. Hasslacher, and Y. Pomeau. "Lattice-Gas Automata for the Navier-Stokes Equation". In: *Phys. Rev. Lett.* 56 (14 Apr. 1986), pp. 1505–1508. DOI: 10.1103/PhysRevLett.56.1505. URL: https://link.aps.org/doi/10.1103/PhysRevLett.56.1505.

[12] Frank Gaitan. "Finding flows of a Navier–Stokes fluid through quantum computing". In: *npj Quantum Information* (2020). URL: https://www.nature.com/articles/s41534-020-00291-0.

[13] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of* 28^{th} *ACM STOC* (1996), pp. 212–219. URL: https://arxiv.org/pdf/quant-ph/9605043.pdf.

[14] J. Hardy, O. de Pazzis, and Y. Pomeau. "Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions". In: *Phys. Rev. A* 13 (5 May 1976), pp. 1949–1961. DOI: 10.1103/PhysRevA.13.1949. URL: https://link.aps.org/doi/10.1103/PhysRevA.13.1949.

[15] J. Hardy, Y. Pomeau, and O. de Pazzis. "Time Evolution of a Two-Dimensional Classical Lattice System". In: *Phys. Rev. Lett.* 31 (5 July 1973), pp. 276–279. DOI: 10.1103/PhysRevLett.31.276. URL: https://link.aps.org/doi/10.1103/PhysRevLett.31.276.

[16] Aram Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* (2009). URL: https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.103.150502.

[17] IBM. *IBM Quantum Hardware*. 2025. URL: https://quantum.ibm.com/services/resources. Accessed: 14.01.2025.

[18] Wael Itani and Sauro Succi. "Analysis of Carleman Linearization of Lattice Boltzmann". In: *Fluids* 7.1 (2022). ISSN: 2311-5521. DOI: 10.3390/fluids7010024. URL: https://www.mdpi.com/2311-5521/7/1/24.

[19] Sriharsha Kocherla et al. "Fully quantum algorithm for mesoscale fluid simulations with application to partial differential equations". In: *AVS Quantum Sci* (2024). DOI: https://doi.org/10.1116/5.0217675.

[20] Timm Krüger et al. *The lattice Boltzmann method*. Springer, 2017. URL: https://link.springer.com/book/10.1007/978-3-319-44649-3.

[21] Oleksandr Kyriienko, Annie E. Paine, and Vincent E. Elfving. "Solving nonlinear differential equations with differentiable quantum circuits". In: *Phys. Rev. A* 103 (5 May 2021), p. 052416. DOI: 10.1103/PhysRevA.103.052416. URL: https://link.aps.org/doi/10.1103/PhysRevA.103.052416.

[22] Jin-Peng Liu et al. "Efficient quantum algorithm for dissipative nonlinear differential equations". In: *Proceedings of the National Academy of Sciences* (2021). URL: https://www.pnas.org/doi/epdf/10.1073/pnas.2026805118.

[23] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". In: *Quantum* (2019). URL: https://arxiv.org/abs/1610.06546v3.

[24] Yuri Manin. "Vychislimoe i nevychislimoe (Computable and uncomputable)". In: *Sovetskoye Radio* (1980).

[25] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information. 10th Anniversary Edition.* 4th ed. Cambridge University Press, 2016. ISBN: 9781107002173.

[26] Furkan Oz et al. "Solving Burgers equation with quantum computing". In: *quantum information processing* (2021). URL: https://link.springer.com/article/10.1007/s11128-021-03391-8.

[27] Philipp Pfeffer, Florian Heyder, and Jörg Schumacher. "Hybrid quantum-classical reservoir computing of thermal convection flow". In: *Phys. Rev. Research* (2022). DOI: https://doi.org/10.1103/PhysRevResearch.4.033176.

[28] Marco A. Pravia et al. "Experimental demonstration of Quantum Lattice Gas Computation". In: *Quantum Information Processing* (2003). URL: https://link.springer.com/article/10.1023/A:1025835216975.

[29] Claudio Sanavio and Sauro Succi. "Lattice Boltzmann-Carleman quantum algorithm and circuit for fluid flows at moderate Reynolds number". In: (2023). URL: https://arxiv.org/pdf/2310.17973.pdf.

[30] Claudio Sanavio, Sauro Succi, and Enea Mauri. "Carleman-Grad approach to the quantum simulation of fluids". In: *arXiv pre-print* (2024). URL: https://doi.org/10.48550/arXiv.2406.01118.

[31] Merel A. Schalkers and Matthias Möller. "Efficient and fail-safe quantum algorithm for the transport equation". In: *Journal of Computational Physics* 502 (2024), p. 112816. DOI: https://doi.org/10.1016/j.jcp.2024.112816.

[32] Merel A. Schalkers and Matthias Möller. "Momentum exchange method for quantum Boltzmann methods". In: *Computers & Fluids* 285 (2024), p. 106453. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2024.106453.

[33] Merel A. Schalkers and Matthias Möller. "On the importance of data encoding for quantum Boltzmann methods". In: *Quantum Information Processing* (2024). DOI: https://doi.org/10.1007/s11128-023-04216-6.

[34] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: (1994). URL: https://doi.org/10.1109/SFCS.1994.365700.

[35] A. Tiftikci and C. Kocar. "Lattice Boltzmann simulation of flow across a staggered tube bundle array". In: *Nuclear engineering and design* (2015). URL: https://www.sciencedirect.com/science/article/pii/S0029549316000315.

[36] B. N. Todorova and R. Steijl. "Quantum algorithm for the collisionless Boltzmann equation". In: *Journal of Computational Physics, 409, 109347* (2020). DOI: http://dx.doi.org/10.1016/j.jcp.2020.109347.

[37] Ronald de Wolf. *Quantum Computing: Lecture Notes.* 2022. URL: https://arxiv.org/abs/1907.09415.

[38] Jeffrey Yepez. "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.

[39] Jeffrey Yepez. “Quantum Lattice-Gas Model for computational fluid dynamics”. In: *Physical Review E* (2001). URL: https://journals.aps.org/pre/abstract/10.1103/PhysRevE.63.046702.

[40] Jeffrey Yepez. “Quantum Lattice-Gas Model for the Burgers Equation”. In: *Journal of statistical physics* (2002). URL: https://link.springer.com/article/10.1023/A:1014514805610.

[41] Jeffrey Yepez and Bruce Boghosian. “An efficient and accurate quantum lattice-gas model for the many-body Schrödinger wave equation”. In: *Computer Physics Communications* (2001). URL: https://www.phys.hawaii.edu/~yepez/static/papers/pdf/2002CompPhysCommVol146No3%2C15Pg280.pdf.

[42] Antonio David Bastida Zamora et al. “Efficient Quantum Lattice Gas Automata”. In: *Computers & Fluids* 286 (2025). URL: https://doi.org/10.1016/j.compfluid.2024.106476.

2

QUANTUM ALGORITHM FOR THE TRANSPORT EQUATION

In this chapter we present a scalable algorithm for fault-tolerant quantum computers for solving the transport equation in two and three spatial dimensions for variable grid sizes and discrete velocities, where the object walls are aligned with the Cartesian grid, the relative difference of velocities in each dimension is bounded by 1 and the total simulated time is dependent on the discrete velocities chosen. We provide detailed descriptions and complexity analyses of all steps of our quantum transport method (QTM) and present numerical results for 2D flows generated in Qiskit as a proof of concept.

Our QTM is based on a novel streaming approach which leads to a reduction in the number of CNOT gates required in comparison to state-of-the-art quantum streaming methods.

As a second highlight of this chapter we present a novel object encoding method, that reduces the complexity of the number of CNOT gates required to encode walls, which now becomes independent of the size of the wall. Finally we present a novel quantum encoding of the particles' discrete velocities that enables a linear speed-up in the costs of reflecting the velocity of a particle, which now becomes independent of the number of velocities encoded.

Our main contribution consists of a detailed description of a fail-safe implementation of a quantum algorithm for the reflection step of the transport equation that can be readily implemented on a physical quantum computer. This fail-safe implementation allows for a variety of initial conditions and particle velocities and leads to physically correct particle flow behavior around the walls, edges and corners of obstacles.

Combining these results we present a novel and fail-safe quantum algorithm for the transport equation that can be used for a multitude of flow configurations and leads to physically correct behavior.

This chapter is based on the publication Efficient and fail-safe quantum algorithm for the transport equation by Schalkers and Möller [18].

We finally show that our approach only requires $\mathcal{O}\left(n_w n_g^2 + d\, n_t^v\, n_{v_{\max}^2}\right)$ *CNOT gates, which is quadratic in the number of qubits necessary to encode the grid and the number of qubits necessary to encode the discrete velocities in a single spatial dimension. This complexity result makes our approach superior to state-of-the-art approaches known in the literature.*

2.1. INTRODUCTION

Computational Fluid Dynamics (CFD) has become an indispensable third pillar in modern engineering sciences complementing theoretical and experimental analysis. Its broad applicability has led practitioners to constantly pushing the limits of numerical simulations for at least four decades. However, stagnation of Moore's law requires a radical rethinking of the way CFD codes and their underlying mathematical algorithms need to be designed in the future. An emerging compute technology that has the potential to become a game changer for future CFD applications is quantum computing as it offers breakthrough solutions for the two major challenges of CFD today: large memory consumption and excessive need for computing power.

In a nutshell, the advantage of quantum computers comes from their capability of encoding an exponentially large amount of data in linearly many quantum bits (qubits) and performing operations on all data simultaneously. This type of quantum parallelism is impossible with classical computers that either need to process data one by one, which results in an exponential growth in sequential run time, or duplicate the hardware resources and process multiple data in parallel, leading to an up to exponential growth in hardware resources. However, the advantage of quantum computing does not come for free. Extracting *all* data from the quantum register requires an exponential amount of runs and measurements, thereby foiling any potential quantum advantage. The art of designing quantum algorithms with practical advantage consists of reducing the amount of necessary read-outs, e.g., by performing some post-processing of field data into scalar quantities of interest on the quantum computer itself.

CFD falls into the category of applications that have a good match with the capabilities and limitations of quantum computers, i.e., large amount of data to be worked with, high computational intensity, and, at the same time, users' primary interest in scalar-valued quantities of interest such as drag and lift coefficients, rather than the visual inspection of entire flow fields.

In what follows, we propose a novel quantum algorithm for the transport equation that surpasses the preceeding approach by Todorova and Steijl [20] in several key manners. First of all our quantum specular reflection step, even though seemingly more complicated, performs the correct reflection behavior in all corner cases. This is not the case for the original transport equation method, which shows incorrect reflection behavior for particles hitting the corners of an obstacle on a non-axis-aligned trajectory. On top of that, our specular reflection approach also ensures that the particles are set back into the flow domain before the start of a new timestep if they virtually traveled into an object. This also avoids incorrect behavior present in the original transport equation method. Furthermore our approach for identifying the internal grid points of obstacles at which particles need to be reflected is more efficient than the method used by Todorova and Steijl. This leads to an additional polynomial speed-up.

We finally show that our quantum implementation of the streaming step outperforms the method of [20] in terms of the the number of CNOT gates required to implement them. We intentionally choose for the latter as performance measure as they are either available as native gates or can be efficiently emulated. Hence, complexity estimates in terms of CNOT gates give a more realistic picture of the costs on a real quantum computer, whereas complexity estimates in terms of multi-controlled gates conceal the costs for decomposing multi-controlled gates into native ones.

Last but not least our method outperforms the current state-of-the-art approach in the encoding of the particles' velocity. We propose a novel encoding of the velocity vector, due to which velocities can be flipped with a single NOT operation performed on a single qubit, whereas before n NOT operations were required to flip the direction of the velocity encoded using n qubits [20].

The rest of this chapter is structured as follows. In Section 2.2 we give a brief introduction to the transport equation. Section 2.3 shows how the data required for our QTM can be encoded efficiently in a quantum register. Here, we define the grid-set up, use of ancillae and the novel velocity-vector encoding that enables the flipping of the velocities being possible in constant time. Subsequently, Section 2.4 provides an efficient quantum incrementation (decrementation) method, which provides a significant speed-up when decomposed into native gates over the incrementation (decrementation) method used in other quantum implementations. We then show how this novel quantum incrementation step can be used to implement a quantum streaming operation. The fail-safe quantum specular reflection step is presented in Section 2.5. Here, we provide an extensive and implementable method to avoid deviant behavior around the corners of objects while ensuring unitarity. We furthermore propose a new approach to ensure that the particles will be repositioned into the flow domain before the start of the next timestep. On top of that we design an efficient method to identify whether or not a grid point is located inside an internal obstacle. Section 2.6 shows the the functionability of our approach by giving the result of preliminary simulation runs. Finally, Section 2.7 compares the complexity of our approach to that of other known methods.

2.2. THE TRANSPORT EQUATION

In this chapter we consider the transport equation to describe the evolution of the relative particle density through time. We consider the distribution function $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ to describe the density of molecules at a position $\boldsymbol{x} = (x^{(1)}, x^{(2)}, x^{(3)})$ with velocity $\boldsymbol{\xi} = (\xi^{(1)}, \xi^{(2)}, \xi^{(3)})$ at time t. The transport equation is given by

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla f = 0. \tag{2.1}$$

This equation describes the transport of particles with a given velocity field but neglects the interaction of particles which is much more difficult to realize with a quantum algorithm.

We assume that the possible velocities are discrete, i.e., $\xi^{(i)} \in \Xi^{(i)} = \{\xi_1, \ldots, \xi_{N_\xi}\}$. In this chapter we only consider velocities $\boldsymbol{\xi} = (\xi^{(1)}, \xi^{(2)}, \xi^{(3)})$, for which the relative differ-

ence in speed in the different dimensions is bounded by

$$c_{\text{rel}} = \max \frac{|\xi^{(i)}|}{|\xi^{(j)}|} \leq 1. \tag{2.2}$$

As detailed in Section 2.2.3 this is not a conceptual limitation of our approach, but helps us to keep the number of corner cases to be considered relatively small.

2.2.1. GRID DEFINITION AND OBSTACLE PLACEMENT

In order to keep track of the temporal evolution of the distribution function $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$, we first define a computational grid with equidistant grid spacing in all two or three spatial dimensions and assign densities to the volumes centered around the grid points. The grid is set up in a straightforward fashion, and grid points can be identified using the Cartesian coordinates system. Obstacles are subsequently placed in between grid points as depicted in Figure 2.1.

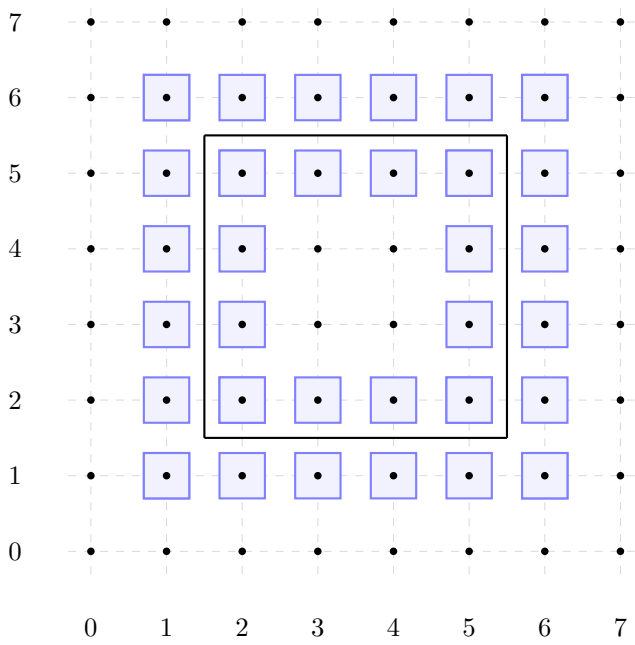


Figure 2.1: Illustration of an obstacle (black box) placed in a computational grid in two dimensions. The grid points ('·' surrounded by blue boxes) surrounding the obstacle placement require special attention in our implementation.

2.2.2. STREAMING

In what follows we describe how to move the particles through space. In the transport method not all particles move in each time step. Furthermore, when a particle moves it does not necessarily move in all dimensions in the same time step. The velocity of each

particle consists of a speed in each spatial dimension. Whether or not a particle will move in a given dimension depends on the speed in that dimension and whether or not a particle traveling at that speed will reach the next grid point in the current time step. This is due to the fact that we choose the size Δt_m of time step m to be such that some particles make it to the next grid point, but none overshoot. To make this idea more precise we define $\xi_{\max} = \max_k |\xi_k|$ and $\xi_{\min} = \min_k |\xi_k|$, with $\xi_k \in \Xi := \bigcup_{i=1}^{d} \Xi^{(i)}$ the set of possible speeds at which a particle can travel. Then, we can normalize the distance between the grid points Δx to enforce it will always be equal to 1. Subsequently, the first time step Δt_0 will have size $\frac{1}{\xi_{\max}}$. All particles that travel with speed $\pm\xi_{\max}$ in a dimension will move one grid point in the first time step in that dimension. To determine the size Δt_m of any subsequent time step m, we keep track of the distances a particle traveling at any of the possible speeds $\xi_k \in \Xi$ would be removed from reaching the next grid point and use this to find the smallest time step necessary for any particle to reach the next grid point in any dimension.

To ensure that particles can travel no further than the neighboring grid points, and thereby adhere to the CFL (Courant-Friedrichs-Lewy) condition, we use a so called CFL counter. This counter keeps track of the distances the particles are located from the next grid point based on their speed in each dimension and thereby determines the size of the next time step.

CFL COUNTER

In what follows, let c_k^m represent how large of a fraction from the current grid point to the next a particle with speed ξ_k has to travel.

We use the following expression

$$c_k^{m+1} = c_k^m + |\xi_k| \frac{\Delta t_m}{\Delta x}, \tag{2.3}$$

where $\xi_k \in \Xi$ and m represents the number of time steps that have been taken so far. Furthermore, Δx represents the equidistant spacing between the grid points as before. It then follows that we can express the size of the time step taken in each iteration as

$$\Delta t_m = \min_{k \in N_\xi} \left(\left[1 - c_k^m \right] \frac{\Delta x}{|\xi_k|} \right). \tag{2.4}$$

In the above equation we minimize over k to ensure that at least one speed reaches the next grid point, and none overshoots.

In a classical transport method c_k^m can be used to interpolate the results when the simulation terminates in a time step that is not a complete cycle. This is not possible in the quantum counterpart and thus if we terminate the simulation in a time step that is not a complete cycle, some small oscillations will occur unless a post-smoothing method is applied. In this thesis we restrict ourselves to running the QTM algorithm for a total time $T = \sum_{m=0}^{N_t - 1} \Delta t_m$ such that $\frac{T}{\xi_m} \in \mathbb{Z}\ \forall m$, where N_t is the total number of time steps.

To complete the quantum transport method we need to describe the behavior when a particle impinges on an obstacle.

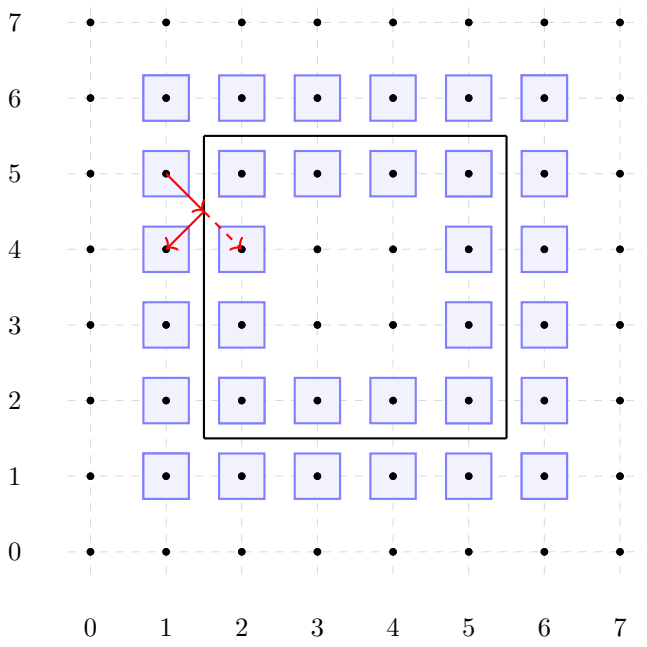


Figure 2.2: Illustration of an obstacle (black box) in the grid with the red arrow representing one possible specular reflection operation. The dashed arrow represents the trajectory in case no reflection had taken place.

2.2.3. REFLECTION BY AN OBSTACLE

In this chapter we consider specular reflection boundary conditions when a particle impinges on an object. For specular reflection boundary conditions the velocity of a particle gets reflected along the axis normal to the object it impinges on. Notice that due to this method we are restricted to modeling objects whose walls are either parallel or perpendicular to each dimension. In order to facilitate the correct reflections we need to keep track of when a particle has come into contact with a wall, this is done differently for different values of c_{rel}.

We first consider the case $c_{\text{rel}} \leq 1$. In this case we know a particle has come into contact with an object if and only if it hits a grid point located in the first layer of the obstacle. We will refer to these grid points as the wall of the object. When a particle reaches a grid point in the wall of an object, its velocity normal to the wall gets reversed and the particle is moved one grid point outside of the object again in the direction normal to the wall(s) that just reflected it. Figure 2.2 gives an example of what such a reflection might look like.

For $c_{\text{rel}} > 1$ the specular reflection step becomes more complicated, as grid points located outside the object can be reached by a particle indicating that the particle has come into contact with an object; See Figure 2.3. Similarly, multiple cases need to be taken into account to determine whether a particle hits a corner point of an object or the wall at a non-corner point. For simplicity we will restrict ourselves to the case $c_{\text{rel}} \leq 1$,

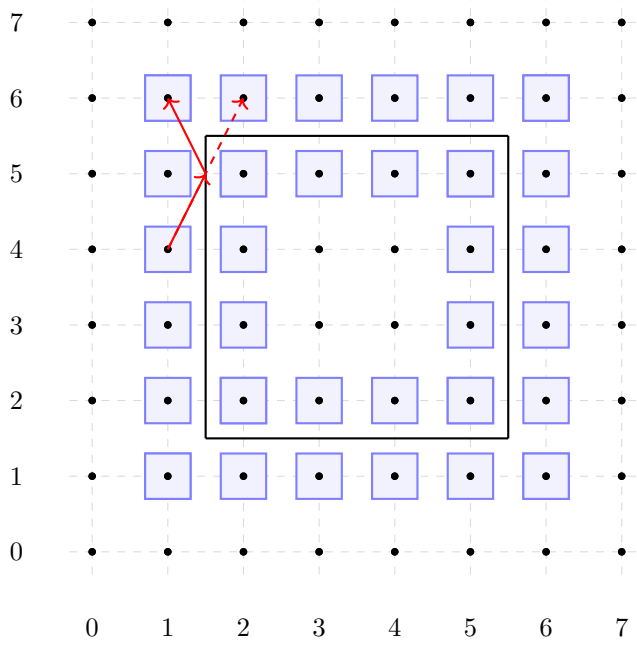


Figure 2.3: Illustration of an obstacle (black box) in the grid with the red arrow representing one possible specular reflection operation when $c_{\text{rel}} > 1$ holds. The dashed arrow represents the trajectory had no reflection taken place.

but we would like to remark that our method can be generalized to any value of c_{rel} at the cost of taking into account more corner cases.

A schematic overview of the proposed QTM is given in Figure 2.4.

2.3. QUANTUM REGISTER SET-UP

In order to implement our QTM approach we first need to define an encoding to give a physically interpretable meaning to quantum basis states. In our implementation we define a mapping, where each parameter is assigned to its own set of qubits, which combined form the total qubit register. We use the following encoding

$$|\underbrace{a_{n_a}\dots a_1}_{\text{ancillae}}\overbrace{g_{n_g}\dots g_1}^{\text{position}}\underbrace{v_{n_v}\dots v_1}_{\text{velocity}}\rangle\,. \tag{2.5}$$

Here, the qubits $a_{n_a},\dots,a_1$ form the ancillae, with $n_a = 4d-2$, where d is the number of spatial dimensions we are modeling. The $g_{n_g}\dots g_1$ qubits form the positional qubits, with $n_g = \sum_{i=1}^{d} n_{g_i}$, where n_{g_i} is the number of qubits required to number all grid points in the i-th spatial dimension. Finally, the qubits $v_{n_v}\dots v_1$ encode the velocity vector, with $n_v = \sum_{i=1}^{d} n_{v_i}$ where n_{v_i} is the number of qubits required number the velocities of the i-

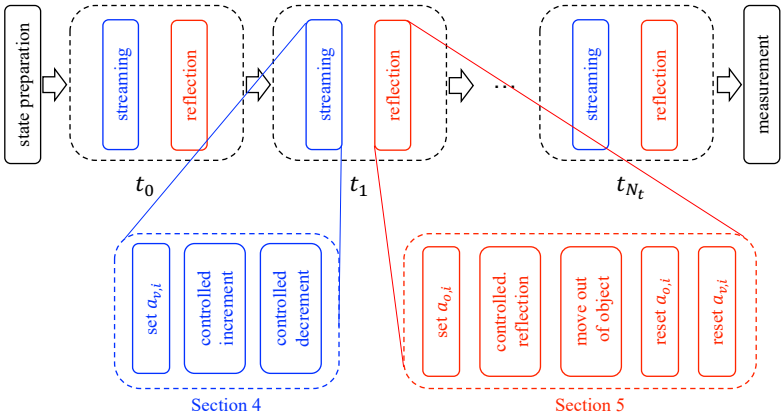


Figure 2.4: Schematic overview of the proposed QTM.

th dimension. In summary, the total number of qubits required to realize our quantum transport method is

$$n_a + n_g + n_v = 4d - 2 + \sum_{i=1}^{d} (n_{g_i} + n_{v_i}), \tag{2.6}$$

which means that a moderate number of a few dozens to hundred fault-tolerant qubits suffice to enable the solution of two and three-dimensional problems.

The details of the mapping of the ancillae, positional and velocity vectors will be explained more in depth in the following sections.

2.3.1. EFFICIENT MAPPING OF VELOCITY VECTOR

One of the speed-ups our algorithm provides compared to the state-of-the-art is due to our mapping of the velocity vector. In each spatial dimension we are working with a predetermined number of discrete velocities N_ξ. For simplicity we will at first only consider the velocity encoding in the one-dimensional case, and call the number of discrete velocities N_ξ. Note that this approach trivially extends to the multi-dimensional case. As N_ξ is the number of discrete velocities required, we need $n_v = \lceil \log_2(N_\xi) \rceil$ qubits to encode all velocities. We subsequently define $N_v = 2^{n_v}$ as the size of the vector representing the encoded velocities. For simplicity we assume N_ξ to be a power of two and so $N_v = N_\xi$ holds.

For our QTM we restrict ourselves to the case that the set of speeds $\mathscr{U}$ consists of positive and negative values, and for each $u \in \mathscr{U}$ both $|u| \in \mathscr{U}$ and $-|u| \in \mathscr{U}$ will hold. We now define $\mathscr{U}_o = [-u_{\max}, \ldots, u_{\max}]$, the ordered list of speeds from largest negative value to largest positive value. Let $u_{\min} = \min_k |u_k|$ as before and let Δu be the distance between the neighboring speeds in $\mathscr{U}_o$, for simplicity we will assume Δu is constant between all indices so $\Delta u = \frac{2u_{\max}}{N_v}$. However, we are not restricted to this simplification.

We propose the mapping

$$|v_{n_v}\dots v_1\rangle = \begin{bmatrix} -u_{\min} \\ -u_{\min}-\Delta u \\ \vdots \\ -u_{\max} \\ u_{\min} \\ u_{\min}+\Delta u \\ \vdots \\ u_{\max} \end{bmatrix}, \tag{2.7}$$

of the discrete velocities to the quantum state. The motivation of this velocity mapping is that we can flip the direction of the velocity by flipping only one qubit. This can be seen by noting that the distance between the index of velocity $-|u_k|$ and $|u_k|$ in $\mathscr{U}_o$ is precisely $\frac{N_v}{2} = 2^{n_v-1}$, and so we can flip between these two velocities by flipping the most significant qubit which encodes the sign of the velocity vector.

This means that we can write the quantum register encoding the velocity in the one dimensional case as

$$|v_{\text{dir}}\, v_{n_v-1}\dots v_1\rangle. \tag{2.8}$$

In the above expression the first qubit encodes the direction of the velocity (positive or negative in the given dimension), and the next n_v-1 qubits encode the magnitude of the velocity. We define $|00\dots0\rangle$ to encode $-u_{\min}$, $|00\dots1\rangle$ to encode $-u_{\min}-\Delta u$ and $|11\dots1\rangle$ to represent $u_{\max}$ etc. It can easily be seen that this leads to a velocity encoding as given in Equation (2.7).

Example *We give an example of our velocity encoding to show that flipping the direction can be established by flipping only the most significant qubit. Let us consider the 1-dimensional case and say we have $N_v = 8$. Now let $u_1 = u_{\min}$, $u_2 = u_{\min}+\Delta u$ and $u_3 = u_{\min}+2\Delta u$ etc., then our velocity encoding gives*

$$|v\rangle = \begin{bmatrix} -u_1 \\ -u_2 \\ -u_3 \\ -u_4 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}.$$

Say our particle is traveling with velocity u_2, then $|v\rangle = |101\rangle$. Now flipping the most significant qubit gives $|v\rangle = |001\rangle$ which maps to $-u_2$ as required.

Extending the above approach to the multidimensional case, the quantum register for the d-dimensional velocity becomes

$$|v_{\text{dir},d}\dots v_{\text{dir},1}\, v^d_{n_{v_d}}\dots v^d_1\, v^{d-1}_{n_{v_{d-1}}}\dots v^{d-1}_1\dots v^1_{n_{v_1}}\dots v^1_1\rangle. \tag{2.9}$$

Former encodings of the velocity vector for quantum transport methods were set-up such that in order to flip the velocity of a particle, all qubits encoding the velocity needed to be flipped [20]. Our new encoding saves $\mathcal{O}(n)$ qubit flips per reflection. Notice that since reflections are usually implemented in multi-control fashion, as described in Section 2.5, this ends up in saving $\mathcal{O}(n)$ computationally costly operations per reflection. In Section 2.7 we give an in depth complexity analysis and comparison of the different methods.

2.3.2. MAPPING OF GRID POINT LOCATIONS ONTO QUBIT STATES

The mapping of the grid point locations onto qubit states is rather straightforward. As before, let $|g_{n_g} \dots g_1\rangle$ represent the state of the qubits encoding the grid point location of the particles. Then, we can write the qubits encoding the location more detailed as

$$|g^d_{n_{g_d}} \dots g^d_1 g^{d-1}_{n_{g_{d-1}}} \dots g^{d-1}_1 \dots g^1_{n_{g_1}} \dots g^1_1\rangle, \tag{2.10}$$

where $g^i_{n_{g_i}} \dots g^i_1$ encodes the i-th dimension of the location of grid points by representing the binary value of the location. The set-up of the grid is rather straightforward with different grid points in each dimension and the particles being able to move one step forward and backward in each time step and in each dimension.

2.3.3. ANCILLAE

The ancillae are the last of the register expressed in Equation (2.7) and are used within the computation only. Each dimension requires one ancilla to keep track of which velocities will be streamed and one ancilla to keep track of the resetting during the reflection step, furthermore we require $2(d-1)$ ancillae to implement a quantum comparator method that will be used in the reflection step, leading to a total of $4d-2$ ancillae required throughout the computation. The exact set-up and utilization of the ancillae will be described in Sections 2.4 and 2.5.

2.4. EFFICIENT QUANTUM STREAMING OPERATION

As described in Section 2.2 the main ingredients of the quantum transport method are the streaming and the reflection operations. In this section we will introduce a novel streaming operation based on the Quantum Draper Adder [6], which we specialize to an efficient quantum incrementation (decrementation) procedure that is cheaper than methods currently in use [5, 8, 7, 20]; see Section 2.7.3.

2.4.1. EFFICIENT QUANTUM INCREMENTATION (DECREMENTATION)

An increment (decrement) operation takes a quantum state $|j\rangle$ to the state $|j+1\rangle$ $\big(|j-1\rangle\big)$. This operation is cyclic, meaning that $|2^n-1\rangle$ ($|0\rangle$) gets incremented (decremented) to $|0\rangle$ ($|2^n-1\rangle$). Let U_{inc} (U_{dec}) express the unitary that increases (decreases) the n qubit state $|j\rangle$, this gives

$$U_{\text{inc}}|j\rangle = |j+1\rangle, \tag{2.11}$$

$$U_{\text{dec}}|j\rangle = |j-1\rangle. \tag{2.12}$$

This quantum primitive is used in many different quantum algorithm and fields such as Quantum Random Walks and Quantum Computational Fluid Dynamics, to name just a few. In the literature this primitive is typically implemented by cascading multi-controlled NOT operations [5, 8, 7, 20, 3]. While this implementation looks elegant on paper, decomposing multi-controlled NOT operations into gates that are native to quantum computers, i.e., single-controlled NOT gates, will lead to a significant increase of the circuit depth.

In this chapter we provide an alternative method leading to a quantum streaming operation which can be implemented more efficiently on real-world quantum computers. The method is inspired by the Quantum Draper Adder (QDA) [6] and uses the same principle, but in contrast to the regular Drapper adder that computes $|a+b\rangle$, where a and b are natural numbers encoded in a quantum register, in our implementation the phase shift operations are not controlled as the addition by $b=1$ is always known beforehand. As we moreover want our operation to be cyclic, no qubit for holding a potential carry over value is required. We are, to the best of our knowledge, the first to use such a QDA inspired approach for the quantum incrementer (decrementer) and in Section 2.7.4 we show that it leads to a significant reduction in the amount of CNOT gates required to run the algorithm.

Our method can be expressed by the circuit given in Figure 2.5. Let $|j\rangle$ be the basis state that we wish to increment (decrement). We claim that

$$|j+1\rangle = (QFT)^{\dagger} U_{P,+} (QFT)\,|j\rangle\,, \tag{2.13}$$

and define

$$U_{\text{inc}} = (QFT)^{\dagger} U_{P,+} (QFT), \tag{2.14}$$

where QFT stands for Quantum Fourier Transform and $U_{P,+}$ will be defined below. We will now show that (2.13) indeed holds. First we remark that

$$QFT\,|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{kj}\,|k\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i k j}{N}}\,|k\rangle\,, \tag{2.15}$$

holds by definition. Then the circuit $U_{P,+}$ consists of one phase shift gate[1] per qubit with an angle determined by the qubit number of the qubit to which it is applied. We apply the phase shift gate $P(\theta)$ to the j-th qubit[2] with angle

$$\theta_j = \frac{\pi}{2^{n-1-j}} = \frac{\pi 2^{j+1}}{N}. \tag{2.16}$$

So $P(\theta_j)$ gets applied to qubit j and adds amplitude $e^{\frac{2\pi i 2^j}{N}}$ to each basis state for which qubit j is in the state 1. This means that if we apply the operation $P(\theta_j)$ to each qubit for the basis state $|k\rangle$ it gets multiplied by a factor $e^{\frac{2\pi i k}{N}}$.

[1] The single qubit phase shift gate can be expressed in matrix form as $P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$.

[2] In this section, for simplicity, we index the qubits ranging from 0 to $n-1$.

And so we get

$$U_{P,+}(QFT)\,|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{2\pi i k j}{N}} e^{\frac{2\pi i k}{N}}\,|k\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{2\pi i k(j+1)}{N}}\,|k\rangle\,. \tag{2.17}$$

It then directly follows that

$$(QFT)^\dagger U_{P,+}(QFT)\,|j\rangle = QFT^\dagger \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{2\pi i k(j+1)}{N}}\,|k\rangle = |j+1\rangle\,. \tag{2.18}$$

In other words our proposed algorithm increases each basis state by 1, whereby the periodic property of $e^{i\theta}$ ensures that the so-defined increase operation is cyclic. Naturally the decrease operation becomes

$$|j\rangle = \Big((QFT)^\dagger U_{P,+}(QFT)\Big)^\dagger |j+1\rangle = (QFT)^\dagger U_{P,+}^\dagger (QFT)\,|j+1\rangle\,. \tag{2.19}$$

Since $U_{P,+}$ consists of a layer of phase shift gates taking the conjugate transpose is equal to shifting the angles to their negative counterparts. So for $U_{P,+}^\dagger$ we apply the phase shift gate $P(-\theta_j)$ to the j-th qubit with angle $-\theta_j$, where θ_j is as given in (2.16). For simplicity we define $U_{P,-} = U_{P,+}^\dagger$. It then follows that

$$U_{\text{dec}} = (QFT)^\dagger U_{P,-}(QFT). \tag{2.20}$$

2.4.2. STREAMING STEP

In the streaming step particles migrate from one discrete point in space to another. In each time step the particles that travel at a discrete speed such that they reach the next grid point in the current time step get incremented (or decremented) one position in space. The list of speeds for which particles traveling at that specific speed need to be incremented (decremented) at a particular time step is pregenerated by the CFL counter as described in Section 2.2.2. Then, controlled on their speeds, the particles are incremented (decremented) one position at a time. This means that we implement the quantum incrementation (decrementation) method as described in the last section in a controlled fashion as will be detailed below.

First of all we notice that we do not have to control the entire incrementation (decrementation) primitive from Section 2.4.1. Since the incrementation (decrementation) step consists of the QFT followed by a layer of phase shift gates followed by $\text{QFT}^\dagger$, it suffices to only control the layer of phase shift gates on the speed of the particles, as the QFT and $\text{QFT}^\dagger$ operations naturally cancel out each other. Furthermore, since the incrementation and decrementation steps are performed directly after each other on the same qubits, we do not have to perform a $\text{QFT}^\dagger$ operation after the phase shift gates of the incrementation step, as this would cancel out with the QFT operation at the start of the decrementation step.

Second, we notice that if we make use of an ancilla we can perform the incrementation and decrementation operations for all speeds that take a step in the current time step using a single operation in a given dimension i.

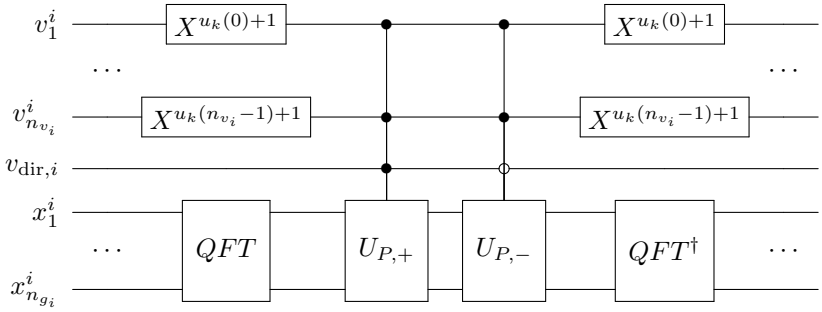


Figure 2.5: Circuit representation of the proposed streaming step in dimension i for one speed $|u_k|$ without extra ancilla qubits.

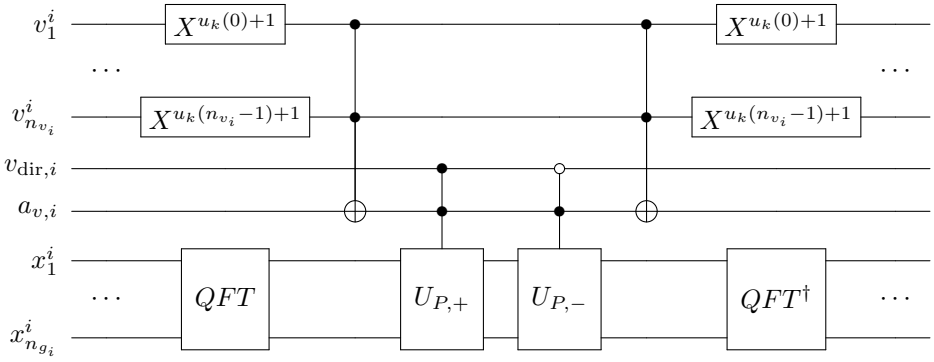


Figure 2.6: Circuit representation of the proposed streaming step in dimension i for one speed $|u_k|$ with ancilla qubit. Note that here we reset the $a_{v,i}$ ancilla after the streaming step. In the proposed algorithm this ancilla will only be reset after the reflection operation has been performed.

To this end, let $|u_k|$ be (one of) the absolute values of the speeds for which the streaming step is to be performed at a certain point in the algorithm. Then in each dimension i we need to increment the particles traveling at speed $u^{(i)} = \pm|u_k|$. Because of the way the binary encodings of u_k and $-u_k$ are related in each dimension, we first perform a controlled-NOT operation between the $n_{v_i} - 1$ qubits determining the absolute value of the velocity and an ancilla qubit $a_{v,i}$. If there are multiple values k such that particles traveling at speed $\pm u_k$ need to be incremented in the given time step, this process is applied to all such k. This means that we simply flip the value of the ancilla qubit $a_{v,i}$ to 1 for all the registers of the particles traveling at a speed such that they should reach the next grid point in dimension i in the current time step. We then use the ancilla $a_{v,i}$ in combination with the directional velocity qubit $v_{\mathrm{dir},i}$ to increase or decrease the position of the correct particles in space in each dimension i. In practice, the streaming step using the ancilla qubits $a_{v,i}$ should be implemented as it leads to a more efficient algorithm, due to the fact that we now perform the incrementation (decrementation) primitive for all the particles taking a step in the given time step using a single operation.

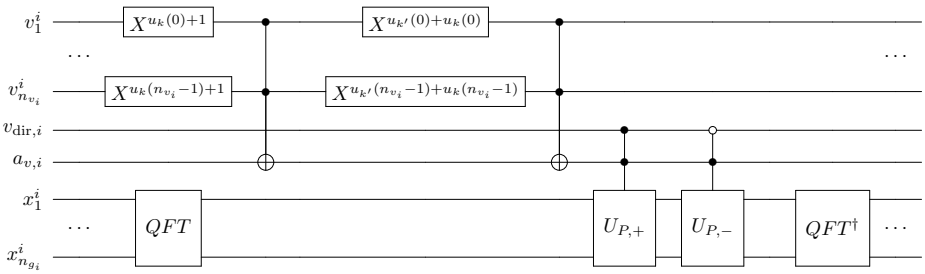


Figure 2.7: Circuit representation of the proposed streaming step in dimension i for two speeds $|u_k|$ and $|u_{k'}|$ with ancilla qubit. Note how the proposed ancilla qubit allows to perform the streaming $U_{P,+}$ for both speeds in one operation. Here we do not reset the ancilla qubit $a_{v,i}$ after the streaming step, as in the proposed algorithm this ancilla will only be reset after the reflection operation has been performed.

Let $u_k(n)$ represent the value of the n-th bit when representing the index of the speed u_k in the velocity vector[3]. Then, we perform the operation $\mathrm{X}^{u_k(l-1)+1}$[4] on each qubit $v_l^i \in \{v_1^i, \dots, v_{n_{v_i}}^i\}$ followed by the multi-controlled NOT operation between the $v_1^i \dots v_{n_{v_i}}^i$ qubits and the $a_{v,i}$ qubit for all speeds $|u_k|$ that take a step in the given time step, before performing the controlled streaming operations. Notice that we need to make sure to apply the $\mathrm{X}^{u_k(l-1)+1}$ operations again after performing the multi-controlled NOT, to reset the speeds to their original states before we apply the $\mathrm{X}^{u_{k'}(l-1)+1}$ associated with the next speed $u_{k'}$ for which the particles need to be streamed in this timestep. Furthermore, notice that we can simply combine these subsequent operations and get $\mathrm{X}^{u_k(l-1)+1+u_{k'}(l-1)+1} = \mathrm{X}^{u_k(l-1)+u_{k'}(l-1)}$ to be applied before the multi-controlled NOT operation to set the ancilla $a_{v,i}$ for the speed $u_{k'}$. Figure 2.5 shows the circuit of the streaming step without the ancilla $a_{v,i}$. Figure 2.6 shows what the circuit of the streaming step with the ancilla $a_{v,i}$ looks like when streaming the particles with speed u_k in dimension i. Figure 2.7 shows how using the ancilla qubit $a_{v,i}$ enables streaming the particles with several (in this example two) speeds u_k and $u_{k'}$ in dimension i using a single $U_{p,+}$ and $U_{p,-}$ operation. In Figure 2.7 we do not reset the ancilla qubit $a_{v,i}$ after performing the streaming step, as in practice these will only get reset after the specular reflection step of the algorithm has been performed.

In the above we have presented the streaming algorithm for a single dimension i. When working with multiple dimensions we perform the exact same operations in each dimension. Note how the streaming of the particle in the different dimensions can be applied to the qubits simultaneously, as different qubits are involved in the streaming step for each dimension.

[3] For example, assume we want to control on the speed being equal to u_2, let $|u_2\rangle = |v_{\text{dir}} v_m \dots v_0\rangle = |10010\rangle$. Then $u_2(0) = 0$ and $u_2(1) = 1$ etc.

[4] Since $u_k(l-1)$ will be zero if the $l-1$-th qubit is in the state $|0\rangle$ when representing the velocity vector u_k, in this case $\mathrm{X}^{u_k(l-1)+1} = \mathrm{X}^1 = \mathrm{X}$ so the qubit gets flipped to the $|1\rangle$ state. If the qubit was already in the state $|1\rangle$, however, we get $\mathrm{X}^{u_k(l-1)+1} = \mathrm{X}^2 = \mathrm{I}$ and so the state of the qubit remains the same. Therefore this operation can be used to prime the states

2.5. QUANTUM SPECULAR REFLECTION STEP

After the completion of the streaming operation, particles that come into contact with an obstacle and have virtually moved into it have to change their travel path; See Section 2.2.3.

Here, we propose a novel and fail-safe approach to the quantum reflection operation. First, the particles that virtually traveled into the obstacle have their velocity direction reversed in the direction normal to the wall encountered. Afterwards, these particles are placed back into the flow domain. As a result of both operations, the particle is located in the correct grid point and travels in the right physical direction in the next time step.

To achieve this there are some corner cases that need to be taken into account explicitly to avoid incorrect reflections, furthermore we make sure that there are no particles residing inside the object at the end of a time step.

2.5.1. SPECULAR REFLECTION STEPS - REQUIREMENTS AND POSSIBLE BREAKDOWN CASES

In this section, we translate the general procedure of the reflection step into a concrete quantum algorithm. Strategies to mitigate the erroneous behavior that might occur for the different corner cases will be discussed on Section 2.5.2.

First we need to identify particles that have reached a grid point inside the obstacle, to ensure that they are placed back into the fluid domain before the start of the next streaming step. This means that controlled on the current location of the particles (namely inside the obstacle), we set them back onto a grid point outside of the obstacle. Performing such an operation on a quantum computer is, however, far from being trivial since we cannot alter the state of certain qubits controlled on their own states, as this would constitute a non-unitary operation. In our case this means that we cannot simply alter the position of the particles in space based on their current position, as that would be precisely attempting an operation on some qubits controlled by themselves.

A way to overcome this is to use a system of ancillae to facilitate this back placement. This, however, introduces a new complexity since the ancillae would also have to be reset after the positional move has been performed in order to be useable in the next time step. The resetting of the ancillae to their original position is nontrivial because we clearly cannot use the original requirements to simply flip the ancillae back, as the original control states were positional and we used the ancillae to control a positional change. Instead, we will have to reset the ancillae based on the new location of the particles in combination with their direction and velocities, i.e., we reverse engineer the particles' previous position.

Another nontrivial aspect of the specular reflection step is to implement it such that all of the reflections are physically correct and we do not encounter unphysical behavior around the corner points. If one were to simply reflect the velocity in the y-direction upon hitting a y-wall and the velocity in the x-direction upon hitting an x-wall[5], incorrect behavior around the corner points will emerge. This is because corner points

[5] Here we define x-wall (y-wall) as a wall spanning points accross the y (x) axis and being at only one point in the x (y) dimension. Therefore particles that encounter such a wall get reflected in the x (y) direction when considering specular reflection boundary conditions.

are points inside the object that are part of walls in multiple directions, i.e. in the two-dimensional case they are part of both an x-wall and a y-wall. What will happen in a non-fail-safe implementation of the specular reflection step is that particles hitting such a grid point coming through just the x- or y-wall have their velocities reflected in both directions, instead of only the one orthogonal to the wall they came through.

Figure 2.8 gives an example of what can go wrong in a non-fail-safe implementation of the specular reflection step. We see that in the cases of the green arrows the behavior is correct, as the velocity is reflected in both the x- and y-direction since the particle hits both an x- and y-wall. For the case of the blue arrows we see that the behavior is also correct, since the velocity parallel to the wall is zero in this case, therefore reflecting the velocity in this direction has no effect. However, for any particle approaching the corner point from another direction, such as the red arrow, the reflection based on a non-fail-safe implementation will certainly be wrong. This is due to the fact that such a particle hits a point associated to both an x-wall *and* a y-wall, even though physically it can easily be seen that the particle only hits either an x- *or* a y-wall. In a non-fail-safe reflection operation this means that the velocity of the particle is erroneously reversed in both the x- *and* the y-direction.

2.5.2. FAIL-SAFE SPECULAR REFLECTION - 2D CASE

In what follows we propose a novel fail-safe specular reflection approach. Figure 2.9 shows an obstacle placed on a grid, with both the grid points inside the object and the grid points outside the object drawn. Furthermore, Figure 2.9 shows the possible reflections that can occur around the obstacle. Using this figure we describe how we can implement a unitary operation that treats all possible reflection cases correctly.

First of all we require d extra ancillae $a_{o,i}$, where i represents the spatial dimension, that facilitate moving the particles out of the obstacle for the next time step and keeping track of which velocity components should be reflected when a particle hits a corner point. Lastly, we make use of the earlier defined ancillae $a_{v,i}$ to correctly set and reset the $a_{o,i}$ ancillae at the beginning and end of the specular reflection step, respectively. Note that using the $a_{v,i}$ ancillae does not add to the complexity of the circuit since we can reuse their state from the streaming step, and we reset them after the specular reflection step instead of directly performing the reset after the streaming operation.

Upon reaching a blue (green) encircled grid point inside the object, the particle has hit an x-wall (y-wall). When hitting a grid point that is part of the x-wall (y-wall) in the object, we flip the ancilla qubit $a_{o,x}$ ($a_{o,y}$) controlled on the $v_{\text{dir},x}$ and $a_{v,x}$ ($v_{\text{dir},y}$ and $a_{v,y}$) qubits.

Specifically, we flip the extra-defined ancillae only when we just took a step in the direction that the wall reflects, and when we travel in a direction that the wall would reflect based on the position of the object (meaning that if we travel in the negative y-direction and we hit a grid point in a y-wall in the bottom of the object we do not flip the $a_{o,y}$ ancilla). This is followed by flipping $v_{\text{dir},x}$ ($v_{\text{dir},y}$) controlled by the $a_{o,x}$ ($a_{o,y}$) ancillae and incrementing the position by one index in the x (y) direction. Here, the term incrementing amounts to streaming in the direction $v_{\text{dir},x}$ ($v_{\text{dir},y}$).

The final step consists of resetting the ancillae $a_{o,i}$. The grid points directly outside the object that are not 'in the vicinity of a corner point' of the object constitute the trivial

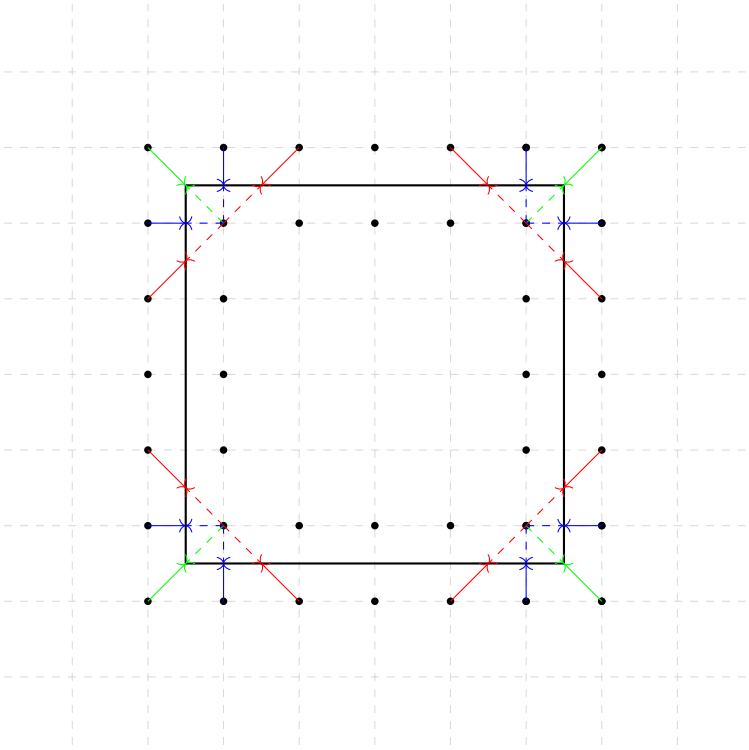


Figure 2.8: Illustration of the different cases of specular reflection at the corner points of an obstacle. While particles traveling into the obstacle along the blue and green velocity trajectories are reflected correctly even by non-fail-safe reflection algorithms, particles approaching the corner points along the red-arrow trajectories require special treatment as is done in our fail-safe specular reflection algorithm.

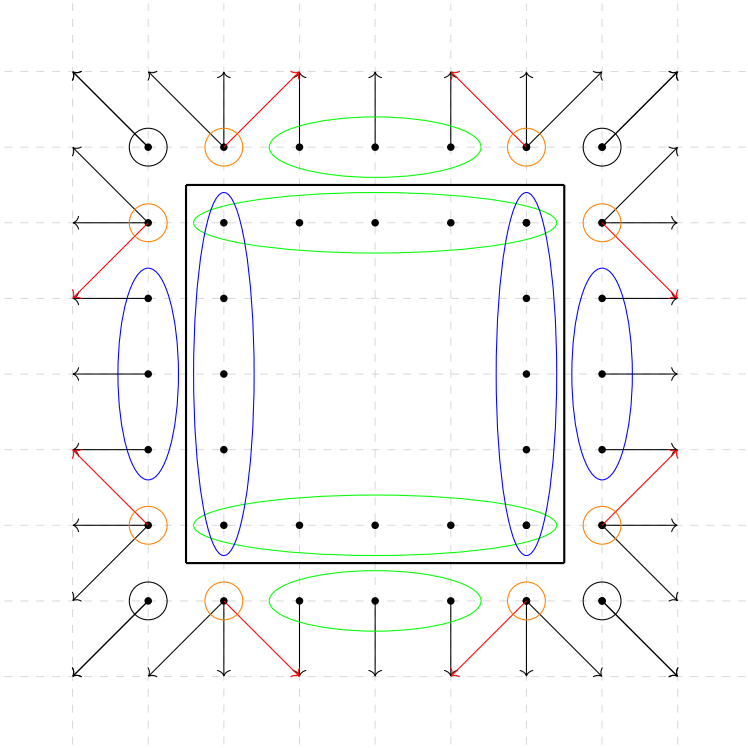


Figure 2.9: Illustration of all possible corner cases to be taken into account when particles collide with an obstacle (black box) and the physically correct reflection behavior as enforced by our fail-safe specular reflection algorithm.

case. By 'in the vicinity of a corner point' we mean that the current grid point the particle is in, has as adjacent grid point inside the object a grid point that is part of both an x-wall and a y-wall. In Figure 2.9 the grid points that are not considered to be 'in the vicinity of a corner point' are identified as the blue (green) encircled grid points outside of the object. All that needs to be done for them is to reset the a_{o_x} ($a_{o,y}$) ancilla controlled on the position, the direction of the x (y) velocity and $a_{v,x}$ ($a_{v,y}$). More specifically, we reset the ancilla for a particle in a grid point that is adjacent to an x-wall (y-wall) if and only if $a_{v,x} = 1$ ($a_{v,y} = 1$) and $v_{\text{dir},x}$ ($v_{\text{dir},y}$) points away from the wall.

Around the corner points of the object we need to be more careful, however. This is due to the fact that the particles could have gotten there via multiple directions, in which case different ancillae would need to be reset.

For the black encircled grid points we have that both the ancillae $a_{o,x}$ and $a_{o,y}$ need to be reset controlled on $v_{\text{dir},x}$, $v_{\text{dir},y}$, $a_{v,x}$ and $a_{v,y}$. Only when $a_{v,x} = a_{v,y} = 1$ holds and $v_{\text{dir},x}$ and $v_{\text{dir},y}$ are such that they both point away from the object, do $a_{o,x}$ and $a_{o,y}$ need to be reset.

For the orange encirled points we do the following. We first reset the ancillae based on the same criteria as the blue (green) encircled qubits. Subsequently, we note that the respective ancilla should not have been reset only in the case of the red arrow. Therefore, we simply re-reset the ancillae based on the criteria that reflect the case of the red arrows, consisting of position in combination with direction in both the x and y direction ($v_{\text{dir},x}$, $v_{\text{dir},y}$) and whether a step was taken in both directions in the former time step ($a_{v,x}$ and $a_{v,y}$). Specifically, for the arrow located on the highest row on the left, this would mean resetting the ancilla controlled on $v_{\text{dir},x}$ being positive, $v_{\text{dir},y}$ being positive, the position being the dot on the highest row second from the left and $a_{v,x}$ and $a_{v,y}$ both being activated. This strategy for defining hand-crafted resetting patterns can easily be adapted to the remaining three corners.

2.5.3. FAIL-SAFE SPECULAR REFLECTION - 3D CASE

We will now provide a generalization of our two-dimensional method to three dimensions. The idea of the set-up is the same, we define grid points just inside the object and grid points outside and directly adjacent to the object. Then, each grid point inside the object is associated with an x-wall or an y-wall or a z-wall. Now, if a particle reaches a grid point associated to an x-wall (y-wall, z-wall), was traveling in a direction that this wall would reflect (meaning $v_{\text{dir},x}$ ($v_{\text{dir},y}$, $v_{\text{dir},z}$) was in the right state) and $a_{v,x} = 1$ ($a_{v,y} = 1$, $a_{v,z} = 1$), the ancilla $a_{o,x}$ ($a_{o,y}$, $a_{o,z}$) will be flipped. Notice that here we are using the exact same logic as in the two-dimensional case.

Subsequently the particles are placed back to the adjacent points inside the fluid domain based on the states of $a_{o,x}$, $a_{o,y}$ and $a_{o,z}$. This step is realized in the same way as in the two-dimensional case.

Finally, we need to reset the ancillae $a_{o,x}$, $a_{o,y}$ and $a_{o,z}$. This is again performed by considering the current position of the particle, in combination with their direction and whether or not they were streamed in the last time step. Compared to the two-dimensional case, in three dimensions there are more distinct cases to be taken into account. Instead of distinguishing only between corner points and non-corner points, we need to distinguish between corner points (where three walls come together), points

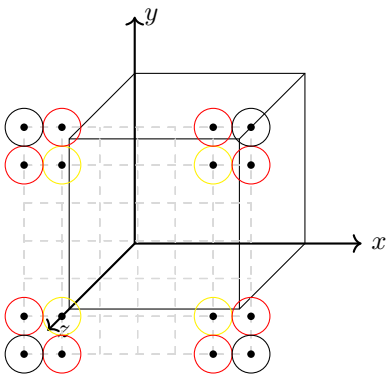


Figure 2.10: Illustration of the corner cases specific to a three-dimensional obstacle. The action applied to the different color-coded grid points is described in Section 2.5.3.

along the edges of the obstacle (where two walls come together) and points on the side of the objects.

For the points on the side of the object and the points along the edges of the obstacle, the rules for reflection will be the same as in the two-dimensional case. Here the points on the edges of the obstacle behave the same as the corner points in the two-dimensional case, and the points on the side of the objects behave the same as the non-corner points in the two-dimensional case.

The corner points where three walls come together require a more in depth consideration. Figures 2.10 and 2.11 show an object in our three-dimensional grid and uses colors to indicate which grid points need to be taken into account as special cases around corner points. In the black encircled points the $a_{o,x}$, $a_{o,y}$ and $a_{o,z}$ qubits are reset if the velocity in all three dimensions travels away from the object and the ancillae $a_{v,x}$, $a_{v,y}$, $a_{v,z}$ are all equal to 1. In the red encircled points the ancillae of the two dimensions which get reflected by the two walls that intersect are reset, if the particle is moving away from the object in the two respective dimensions and the particle took a step in both dimensions and just before did not travel a step in the third dimension towards the object. In the yellow encircled point we consider the following behaviors. Here we only reset the ancilla associated with the dimension the wall reflects in if we just took a step in that dimension and the directional qubit points away from the object. Furthermore, we need to check that in none of the other two dimensions we just took a step towards the object.

2.5.4. EFFICIENT OBJECT ENCODING

In this section we propose a novel approach based on quantum comparison operations for encoding objects in such a way that we can efficiently identify whether or not a grid point is part of a certain object wall. Our approach requires $2(d-1)$ extra ancillae and will be explained at the hand of a two-dimensional example.

Assume that we want to encode an x-wall (note that everything works the same to encode an y-wall), for example the left-most x-wall of Figure 2.9. Let the grid points just

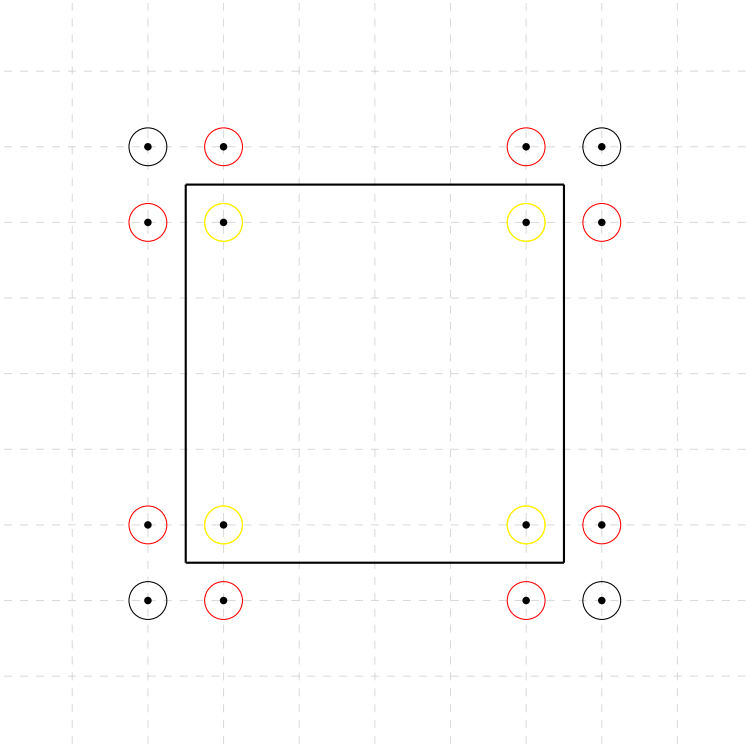


Figure 2.11: View from the positive z-axis onto the z-wall of the obstacle. Note that the z-wall lies half a grid point in the z-direction below the grid with the encircled points.

inside the object next to the x-wall, i.e., the left-most blue encircled grid points inside the object of Figure 2.9, range from $[l,u]$ in the y-axis. We first need to determine whether or not we are in one of the left-most blue encircled grid points inside the object with the aid of two quantum comparison operations. The first quantum comparison operation checks whether $y \geq l$ holds. If so we flip the first extra ancilla from its start state $a_{l,1} = 0$ to $a_{l,1} = 1$. Then we check if $y \leq u$ holds, and if so we flip the second extra ancilla to $a_{u,1} = 1$. Now, controlled on these two ancillae and the state of the qubits encoding the position in the x dimension of these left-most blue encircled qubits, we flip the ancilla $a_{o,x}$ indicating that we are in a wall which reflects the velocity in the x-direction. Now we simply reset the $a_{l,1}$ and $a_{u,1}$ ancillae so that they can be reused for other walls, by performing the same quantum comparison operations as described before[6]. In the two-dimensional case we perform this operation for all walls in order to set the $a_{o,x}$ and $a_{o,y}$ qubits as required. After completion of this step we continue in the same fashion as described in Section 2.5.2.

What remains is to reset the $a_{o,x}$ and $a_{o,y}$ qubits after the reflection and streaming steps have been performed which we accomplish, again, with the aid of the quantum comparison operation. To continue with our example of the left-most x-wall of Figure 2.9, we again perform two comparison operations to check whether we are in the left-most blue encircled grid points right outside the wall, which amounts to checking $y \in [l+1, u-1]$. This can be achieved with the same logic as before namely by checking $y \geq l+1$ and $y \leq u-1$. All the other steps and the resetting are performed in the same manner as explained before. The extension to the three-dimensional case is straightforward.

2.5.5. QUANTUM COMPARISON OPERATION

In our implementation we use the quantum comparison operation from [9], which compares the integer value i of the n-qubit quantum state $|i\rangle$ with a pre-determined constant k saving the result of the comparison in a separate qubit.

The quantum comparison algorithm works as follows. Say we wish to determine whether the integer value i encoded in basis state $|i\rangle$ is smaller than k[7]. Then, we first subtract the integer k from the value encoded by the $n+1$ qubits encoding the state $|0\rangle \otimes |i\rangle$, where the prepended $|0\rangle$ qubit will hold the result of the computation. Now there are two cases, $i \geq k$ and $i < k$.

Assume that we are in the case $i \geq k$. Then subtracting k from i encoded in $|0\rangle \otimes |i\rangle$ gives $|0\rangle \otimes |i-k\rangle$, which leaves the prepended $|0\rangle$ qubit unchanged. Now we simply perform an addition of the value k to the n-qubits that were used to encode i and now encode $|i-k\rangle$. This gives $|0\rangle \otimes |i-k+k\rangle = |0\rangle \otimes |i\rangle$. And so in total we end up with a quantum register in the state $|0\rangle \otimes |i\rangle$. Where the $|0\rangle$ state is the result of the computation which means that $i \geq k$ in fact holds, and the last n qubits are again in the original state $|i\rangle$.

[6] Note that in the example of Figure 2.9 it would be most efficient to first flip the ancilla $a_{o,x}$ as well, controlled on the $a_{l,1}$ and $a_{u,1}$ qubits in combination with the qubits encoding the position in the x dimension being in the position of the right-most blue encircled grid points inside the object. But for now we were only considering the left-most x-wall.

[7] Note that $k \leq 2^n - 1$ always holds, since otherwise we trivially know that k is larger than any value that n qubits can encode.

Now let us assume that we are in the second case $i < k$. Again, we start by periodically subtracting the integer k from i encoded in the state $|0\rangle \otimes |i\rangle$, only now since $k > i$ we end up flipping the state of the prepended $|0\rangle$ qubit. This is because periodically subtracting k from i in a state encoded by $n+1$ qubits results in the state $|2^{n+1} - k + i\rangle$. Since $k \leq 2^n - 1$ and $i \geq 0$ we must have that $2^{n+1} - k + i \geq 2^n$ and so the state of the most significant qubit must be equal to $|1\rangle$. This means that the total qubit register is now in the state $|1\rangle \otimes |2^{n+1} - k + i - 2^n\rangle = |1\rangle \otimes |2^n - k + i\rangle$.

Then, as before, we simply add the integer k again to the last n qubits of the register leaving us with $|1\rangle \otimes |2^n - 1 + i\rangle = |1\rangle \otimes |i\rangle$ due to periodicity. Notice how the most significant qubit is left in the state $|1\rangle$ indicating that in fact $k > i$, whilst again the qubits encoding $|i\rangle$ have not changed.

Multiple realizations of quantum addition and subtraction operations are described in the literature [12, 19, 4, 6] and can be used for the comparison operation, all having their own complexities and required ancillae associated to them.

A quantum primitive for checking whether $i \geq k$ holds can be easily designed from the above ($i < k$) by simply negating the qubit that holds the result after performing $i < k$.

Implementing a quantum primitive that determines $i \leq k$ is a bit more involved as we need to consider two different cases. First, if $k = 2^n - 1$ simply flip the ancilla holding the result, as $i \leq k$ trivially holds, otherwise we can implement $i < k+1$ to get the desired result. Notice that here we can easily implement this two case method for $i \leq k$ since k is an integer determined before the running of the algorithm.

2.6. RESULTS

To demonstrate the correct functioning of our QTM we implemented the proposed algorithm in Qiskit [1] and performed some preliminary simulation runs for a moderately small grid in two spatial dimensions. All runs were performed with Qiskit's local quantum simulator Aer on an 8-core Intel i7-10610U CPU running at 1.8GHz. For simplicity, we assumed perfect qubits, i.e. no noise model and all-to-all qubit connections, i.e., a generic QPU.

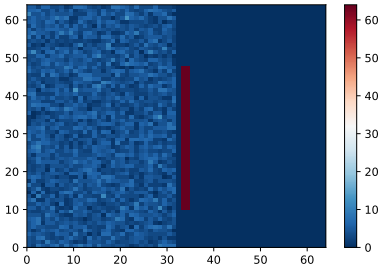
Our overall algorithm consumes just 22 qubits in total, 6+6=12 qubits for the grid locations, 2+2=4 qubits to encode the velocity vectors ($N_v = 2$ in both directions), and 6 ancillae. Based on the amount of qubits, today's quantum computers should be able to execute our QTM solver, however, the circuit depth exceeds capacities of todays devices by orders of magnitude so that we are only able to show results produced on a quantum computer simulator.

Figure 2.12 shows a sequence of numerical results computed on a 64×64 grid with an internal object of size 3×39 located with its lower left corner at position $(34, 11)$. The initial state was prepared by applying Hadamard gates to all the qubits encoding the grid in the y-dimension and all but the most significant qubit encoding the grid in the x-dimension, that is, all particles are equidistributed in the left half of the fluid domain, whereas the right half is in vacuum state. Perfectly reflecting boundary conditions are prescribed at the internal obstacle while periodic boundary conditions are prescribed at all four domain boundaries.

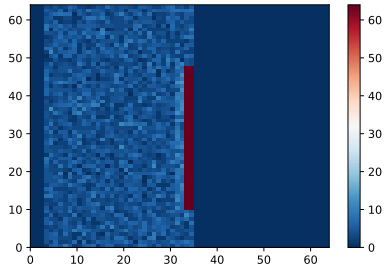
In the preparation step of the quantum state we also applied a Hadamard gate the the $v_{\text{dir},y}$ ancilla, spreading out the direction of the velocities of the particles in the y-dimension. Finally, we apply a NOT gate to the $v_{\text{dir},x}$ ancilla, leading to the particles traveling in the positive x-direction. Figures 2.12b–2.12f illustrate how particles move in the positive x-direction and in both the positive and negative y-directions filling the vacuum behind the obstacle as expected.

In order to stay as close as possible to the capabilities of a physical quantum computer we performed 8.192 shots. Knowing the exact size and position of the obstacle we excluded 117 from the 4.096 possible states from the measurement so that, on average, each grid point gets measures twice.

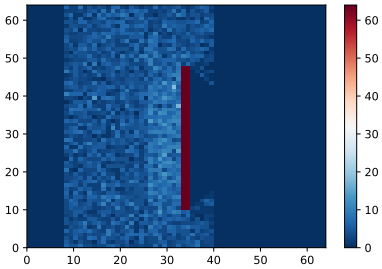
Figure 2.13 depicts the same sequence of results with 524.288 measurements showing a much better enunciation of the flow pattern. All plots show the density of particles at the respective grid points. Future research will focus on problem-specific measurements of application specific integral quantities of interest that might require much less measurements.



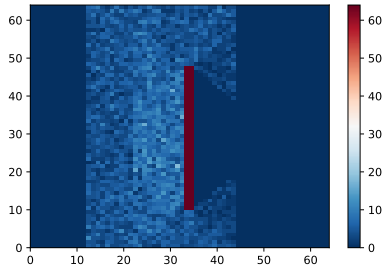
(a) Output after 0 timesteps.



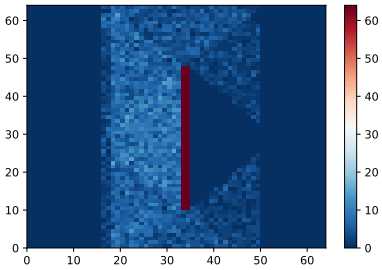
(b) Output after 3 timesteps.



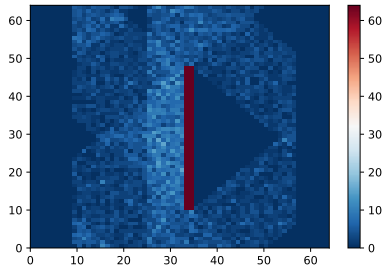
(c) Output after 8 timesteps.



(d) Output after 12 timesteps.

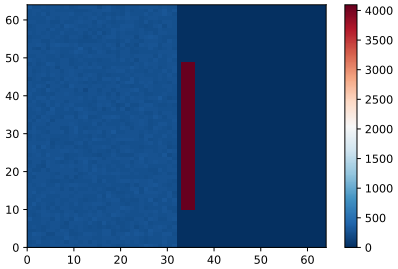


(e) Output after 18 timesteps.

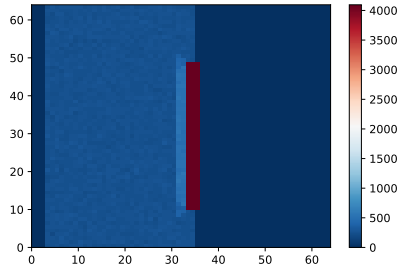


(f) Output after 25 timesteps.

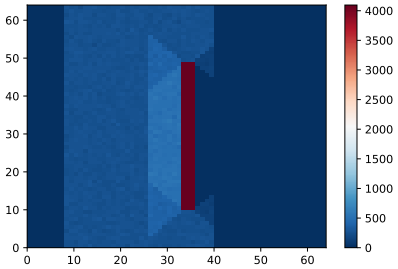
Figure 2.12: Sequence of solutions computed on a 64×64 grid by the proposed QTM solver on 22 simulated qubits adopting 8.192 measurement.



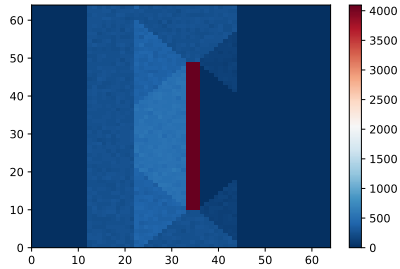
(a) Output after 0 timesteps.



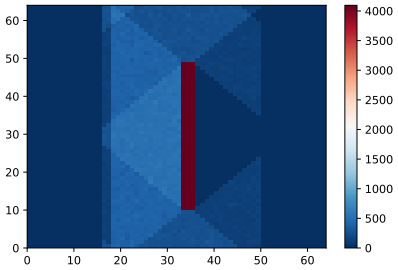
(b) Output after 3 timesteps.



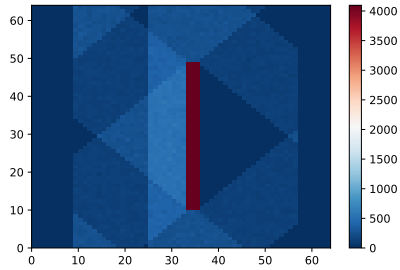
(c) Output after 8 timesteps.



(d) Output after 12 timesteps.



(e) Output after 18 timesteps.



(f) Output after 25 timesteps.

Figure 2.13: Sequence of solutions computed on a 64×64 grid by the proposed QTM solver on 22 simulated qubits adopting 524.288 measurements.

2.7. COMPLEXITY ANALYSIS

In this section we give a detailed complexity analysis of our approach. Since most quantum hardware can only implement single- and two-qubit gates natively [17, 13], we decompose the multi-qubit operations into two-qubit gates to find a realistic quantum complexity for the proposed method. Specifically we choose to give the complexity in terms of CNOT gates, while CNOT is one of the native gates for IBM QPUs [13] it needs to be emulated by other two-qubit gates on other quantum hardware such as the QPUs

from Rigetti [17]. However, in these cases the CNOT gate can be decomposed using a fixed number of the natively implemented two-qubit gates, so this does not effect the overall complexity.

We subsequently compare our found complexity to the complexity of the current best known quantum algorithm for the transport equation [20]. In doing so we show that our method not only outperforms the current state-of-the-art by implementing the quantum reflection step in a fail-safe manner, but we also reach a better complexity.

2.7.1. COMPLEXITY OF MULTI-CONTROLLED NOT OPERATIONS

In order to provide a cost analysis for our algorithm, we first determine the cost associated to implementing a multi-controlled NOT gate, denoted as C^{p}NOT in the following, in terms of CNOT gates. As the decomposition of C^{p}NOT gates into CNOT gates is a field of active research, we selected a few representative approaches, namely, one without ancilla qubits, one with a linear amount of ancilla qubits and multiple with a constant amount of ancilla qubits.

A decomposition without any ancilla qubits was given by Barenco et al. in [2]. The reported decomposition requires $48(p+1)^2 + \Theta(p)$ 'basic operations'[8], from which we deduce that approximately cp^2 with $22 < c < 48$ CNOT gates will be required leading to a complexity of $\mathcal{O}\left(p^2\right)$ CNOT gates.

Nielsen and Chuang present an alternative decomposition with a linear amount of ancilla qubit [15]. The decomposition of a single C^{p}NOT gate requires $p-1$ ancillae and $2(p-1)$ Toffoli gates[9]. Each Toffoli gate in turn is decomposed using 6 CNOT gates (and some other operations) [15]. Therefore, a total of $12(p-1)$ CNOT gates is required to decompose a C^{p}NOT operation into native one- and two-qubit gates.

As a third alternative the paper by Barenco et al. [2] provides a decomposition of C^{p}NOT operations for $p \geq 5$ using a single ancilla, $8(p-3)$ Toffoli gates and $48(p+2)$ 'basic operations' leading to cp CNOT gates with $70 \leq c \leq 96$.

In addition to the three approaches mentioned above, we can also use the quantum comparison operation discussed in Section 2.5.5 in combination with only one ancilla qubit. In fact, one can find whether or not the p control qubits are in the state $|q_1 \dots q_p\rangle = |1 \dots 1\rangle$ by using the comparison operation to check whether $q \geq 2^p - 1$ holds, where q is the integer value encoded by the qubits $q_1 \dots q_p$. The cost of such a comparison operation depends on the costs of the constant addition operation. Using the Draper adder as described in Section 2.4 this method requires $4p^2 + 6p + 3$ CNOT operations[10]. Therefore this method requires $\mathcal{O}\left(p^2\right)$ CNOT gates and one additional ancilla.

[8] Here the term basic operations implies either a CNOT or single qubit gate.

[9] A Toffoli gate is a CCNOT gate, i.e. a controlled NOT gate with two control qubits.

[10] Since a QFT operation applied to k qubits consists of the following two-qubit gates, first $\frac{k(k-1)}{2}$ controlled phase shift gates followed by $\lfloor k/2 \rfloor$ swap gates. A swap gate can be decomposed using 3 CNOT operations and a controlled phase shift operation requires two CNOT operations. Therefore the total cost of a QFT operation applied to k qubits is $k(k-1) + \lfloor \frac{3k}{2} \rfloor$ CNOT operations. Then the total cost given in the text follows from the fact that the Draper Adder implementation in combination with the comparison operation consists of first a QFT and QFT† applied to $p+1$ qubits followed by a QFT and QFT† applied to p qubits.

This implementation appears to match the complexity of the 'recursion method', provided by Qiskit [1], which also requires 1 ancilla qubit. IBM does not, however, give a complexity analysis in terms of the number of CNOT gates required or a reference detailing the implementation. We numerically analyzed the complexity of this implementation by running it for several values of p (up to $p = 40$) and found experimentally that it requires $\approx 2p^2$ CNOT gates.

Having, in our opinion, the best trade-off between the amount of CNOT gates and ancilla qubits required for realistic values of p, we choose to use the 'comparison' or 'recursion' multi-controlled NOT decomposition in our further analysis.

We would like to stress here that there might be cheaper methods for implementing a multi-controlled NOT operation possible as this is a topic of active research. This will only decrease the total cost of our method and it will not change the conclusion presented in Section 2.7.3 of our method having the lowest complexity of any known QTM.

2.7.2. COMPLEXITY OF OUR QTM SOLVER

In this section we give a detailed analysis of the costs of our QTM solver by first analyzing the different steps separately and, subsequently, combining these results to find the total complexity of the algorithm.

COMPLEXITY OF QUANTUM STATE PREPARATION

In order to run the QTM solver, we first need to prepare the input state to represent the desired particle distribution and velocities, at the initial time $t = 0$. As it is known that the preparation of arbitrary quantum states can become exponentially expensive [16, 15], we restrict ourselves for the numerical results presented in this paper to equidistributed particles and mention a few publications that describe efficient preparation procedures for specific quantum state distributions.

Preparing general sparse quantum states is an active area of research with promising results [22, 21, 10], since our input quantum state is typically highly sparse, this is potentially useful for the efficient state preparation of our quantum states. Another interesting result for the input state preparation is given by Grover and Rudolph [11], who present an efficient process for preparing quantum states that form a discrete approximation of an efficiently integrable probability density function, such as log-concave distribution functions.

COMPLEXITY OF THE STREAMING STEP

The quantum streaming step consists of a streaming operation applied to the particles traveling at a pre-determined velocity. In order to implement this we first apply a C^pNOT gate between the velocity qubits in each dimension i and the $a_{v,i}$ ancilla qubits. Each C^pNOT gate has a total of $p = n_{v_i} - 1$ control qubits. We implement this operation for the particles whose velocity magnitudes are marked for advancement in that particular timestep. The number of velocity magnitudes being advanced in each timestep n_t^v can trivially be capped by N_v, but in practice will be equal to 1 in 89.6% of the cases. We found this percentage by performing numerical analysis for N_v up to 1024. Subsequently, the increase operation is applied controlled on the $a_{v,i}$ qubits.

Our increase operator consists of a QFT operation applied to the n_{g_i} qubits in each dimension i, followed by a phase shift on the n_{g_i} qubits and a final QFT† operation. In each dimension i, a QFT operation requires $\mathcal{O}(n_{g_i}^2)$ two-qubit operations [14], the phase shifts applied to the n_{g_i} qubits is controlled on the $a_{v,i}$ ancilla qubits and so this operation costs $\mathcal{O}(n_{g_i}^2)$ two-qubit operations. Therefore, our increase operator only requires $\mathcal{O}(n_{g_i}^2)$ CNOT operations, controlled on the $a_{v,i}$ qubits in each dimension i, which allows us to cap its complexity by $\mathcal{O}(dn_{g_{\max}}^2)$ CNOT operations. In total that leaves us with a complexity of the streaming step of $\mathcal{O}\left(dn_t^v\right)$ in the number of C^{p}NOT operations with $p = n_{v_{\max}} - 1 + \mathcal{O}\left(dn_{g_{\max}}^2\right)$ CNOT operations.

Using the complexity analysis of the multi-controlled NOT operations provided in Subsection 2.7.1, we get an overall complexity of

$$\mathcal{O}\left(dn_t^v n_{v_{\max}}^2 + dn_{g_{\max}}^2\right)$$

in the amount of CNOT gates for the streaming step.

COMPLEXITY OF THE SPECULAR REFLECTION STEP

The specular reflection step for each wall starts with two quantum comparison operations, applied to the qubits encoding the location in a single dimension with the a_l and a_u qubits storing the result. This leads to a complexity of $\mathcal{O}\left(n_{g_{\max}}^2\right)$ CNOT gates when using the Draper adder [6] to implement the comparison operations.

Subsequently a multi-controlled NOT gate applied to the ancilla qubits $a_{o,i}$, $i \in \{1, \ldots, d\}$, controlled on the position in the dimension the considered wall reflects, the $2(d-1)$ a_u, a_l ancillae and one $a_{v,i}$ qubits and one directional qubit $v_{\text{dir},i}$ is applied. Therefore the second step of the specular reflection for each wall consists of a C^{p}NOT operations with $p \leq n_{g_{\max}} + 2(d-1) + 2 = n_{g_{\max}} + 2d$.

Since the two steps above are applied to each of the n_w walls we get a total complexity of $\mathcal{O}\left(n_w n_{g_{\max}}^2\right)$ CNOT gates and $\mathcal{O}(n_w)$ C^{p}NOT operations with $p \leq n_{g_{\max}} + 2d$.

Then, controlled on the $a_{o,i}$ qubits we perform a NOT operation on the $v_{\text{dir},i}$ qubits. This operation consists of d CNOT gates, and so the complexity is $\mathcal{O}(d)$ CNOT gates. This is followed by a single incrementation operation applied to the positional qubits controlled on $a_{o,i}$ in each dimension. The incrementation operation again has complexity $\mathcal{O}\left(dn_{g_{\max}}^2\right)$ CNOT operations.

Subsequently, we reset the $a_{o,i}$ ancillae. Which is established for each wall by first performing two quantum comparison operations, followed by a multi-controlled NOT gate targeting the $a_{o,i}$ ancillae controlled on the qubits encoding the position in a single dimension in combination with the $2(d-1)$ a_u, a_l ancillae, one $a_{v,i}$ qubit and one directional qubit $v_{\text{dir},i}$. This leaves us with a total complexity of again $\mathcal{O}\left(n_w n_{g_{\max}}^2\right)$ CNOT gates and $\mathcal{O}(n_w)$ C^{p}NOT operations with $p \leq n_{g_{\max}} + 2d$.

Only for the resetting of the $a_{o,i}$ qubits we need to take into account the special rules for resetting the corner cases by applying a multi-controlled NOT gate targeting an ancilla $a_{o,i}$, controlled on the position as well as the $a_{v,i}$ qubits and the direction qubits $v_{\text{dir},i}$. Therefore the complexity of resetting an ancilla $a_{o,i}$ for a corner case becomes

$\mathcal{O}(1)$ C^{p}NOT operations with $p \leq n_g + 2d$. Since the number of corner cases is linear in the amount of walls the total complexity becomes $\mathcal{O}(n_w)$ C^{p}NOT operations with $p \leq n_g + 2d$.

At the end of the specular reflection step, we reset the $a_{v,i}$ ancilla qubits in each dimension i, using a C^{p}NOT gate with $p = n_{v_i} - 1$. As explained in Section 2.7.2 the total costs of this operation amount to $\mathcal{O}\left(d n_t^v\right)$ C^{p}NOT operations with $p = n_{v_{\max}} - 1$.

In total, the complexity of the reflection step is equal to $\mathcal{O}\left(n_w n_{g_{\max}}^2\right)$ CNOT gates plus $\mathcal{O}(n_w)$ C^{p}NOT operations with $p \leq n_{g_{\max}} + 2d$ plus $\mathcal{O}(n_w)$ C^{p}NOT operations with $p \leq n_g + 2d$ plus $\mathcal{O}(d)$ CNOT gates plus $\mathcal{O}\left(d n_{g_{\max}}^2\right)$ CNOT gates.

When combining these we get a total complexity of $\mathcal{O}\left((n_w + d) n_{g_{\max}}^2\right)$ CNOT gates plus $\mathcal{O}(n_w)$ C^{p}NOT operations with $p \leq n_g + 2d$, for the complexity of the reflection step.

Translating these results in terms of multi-controlled NOT complexities into single-controlled NOT complexities using the results from Subsection 2.7.1, we get the following CNOT gate complexity

$$\mathcal{O}\left(n_w \left(n_g + 2d\right)^2 + (n_w + d) n_{g_{\max}}^2\right) = \mathcal{O}\left(n_w n_g^2 + (n_w + d) n_{g_{\max}}^2\right) = \mathcal{O}\left(n_w n_g^2\right).$$

TOTAL QTM COMPLEXITY

Combining the complexities of the convection and specular reflection step as detailed above, we obtain that the total complexity of the algorithm per timestep amounts to $\mathcal{O}\left((n_w + d) n_{g_{\max}}^2\right)$ CNOT gates and $\mathcal{O}(n_w)$ C^{p}NOT gates with $p \leq n_g + 2d$ and $\mathcal{O}\left(d n_t^v\right)$ in the number of C^{p}NOT operations with $p = n_{v_{\max}} - 1$.

Using the multi-controlled NOT decomposition as before, this leads to

$$\mathcal{O}\left(n_w n_g^2 + d n_t^v n_{v_{\max}}^2 + d n_{g_{\max}}^2\right) = \mathcal{O}\left(n_w n_g^2 + d n_t^v n_{v_{\max}}^2\right)$$

CNOT gates per time step.

2.7.3. COMPLEXITY OF ALTERNATIVE QTM IMPLEMENTATIONS

An alternative quantum algorithm for the transport equation presented in the literature is the one by Todorova and Steijl [20]. Though not being fully fail-safe in the specular reflection step, the authors present some complexity analysis which led us to perform a rigorous comparison of the complexities of both approaches.

As reported in their paper, Todorova and Steijl conclude a complexity in terms of C^{p}NOT gates of $\mathcal{O}\left(d N_v \log_2(D/h)\right)$[11] for the streaming step per timestep with $p = n_{g_{\max}} + n_{v_{\max}}$. The complexity of their specular-reflection boundary condition, again, quantified in terms of C^{p}NOT gates, is $\mathcal{O}\left(d N_v \log_2(D/h)\right)$. Here, the authors do not provide an explicit estimation for p, but since they control on positions in all d spatial dimensions and streaming speeds combined, we derive a complexity of $p = n_g + n_{v_{\max}}$.

[11] In their work D refers to the size of the domain and h to the space between grid points so D/h expresses the numbers of grid points per dimension.

Since assumptions and parameters in the paper by Todorova and Steijl differ from ours, it is not immediately clear how to compare their complexity analysis with ours on a fair basis. Rewriting their complexity result in terms of the parameters adopted in our analysis yields $\mathcal{O}\left(dN_v \log_2(D/h)\right) = \mathcal{O}\left(dN_v n_{g_{\max}}\right)$ C^{p}NOT operations with $p \leq n_g + n_{v_{\max}}$. Finally, rewriting their complexity result in terms of single-controlled CNOT rather than C^{p}NOT gates, yields a total complexity of $\mathcal{O}\left(dN_v n_{g_{\max}}(n_g + n_{v_{\max}})^2\right)$.

Another difference that makes our complexity hard to compare with the complexity of the algorithm by Todorova and Steijl is that the aforementioned authors do not adopt the variable n_v^t, and instead use the maximum value N_v. For ease of comparison we will replace n_v^t with N_v in our estimates which is possible as $n_v^t \leq N_v$ holds trivially. Finally, Todorova and Steijl also do not take the amount of walls into account explicitly, but instead seem to consider it a constant, we need to relax this parameter in our complexity estimation as well. With all the aforementioned changes in place we can rewrite our leading complexity terms as $\mathcal{O}\left(n_g^2 + dN_v n_{v_{\max}}^2\right)$ CNOT operations, which is still significantly cheaper than the QTM approach by Todorova and Steijl having a total CNOT complexity of $\mathcal{O}\left(dN_v n_{g_{\max}}(n_g + n_{v_{\max}})^2\right)$.

2.7.4. COMPLEXITY COMPARISON OF INCREMENTATION OPERATIONS

One of the improvements we propose in this chapter with respect to other known method [20] is the use of the Quantum Draper Adder (QDA) [6] inspired approach to implement the quantum incrementation operation, leading to a cheaper quantum primitive for the streaming operation. In this section we provide a comparison of the costs in terms of CNOT gates of the incrementation method with our QDA inspired approach compared to the approache implemented in [20].

Consider the QDA inspired incrementation operation applied to the qubits g_i encoding the position in dimension i. It consists of a QFT followed by a phase shift gate and finally a QFT† operation. The cost of implementing the QFT (QFT†) operation is $n_{g_i}^2 + \frac{1}{2}n_{g_i}$ CNOT operations as shown in Section 2.7.1. Therefore our incrementation operation can be applied at a total cost of $2n_{g_i}^2 + n_{g_i}$ CNOT operations.

The cost of the quantum incrementation operation proposed in [20] consists of $\sum_{p=0}^{n_{g_i}-1}$ C^{p}NOT gates. Only when assuming that a C^{p}NOT can be decomposed by $\mathcal{O}(p)$ CNOT gates, we find that the total costs of the incrementation operation implemented in these papers have the same order of magnitude with respect to the amount of CNOT gates as our QDA inspired incrementation method. To the best of our knowledge, however, a decomposition of a C^{p}NOT gate that requires $\mathcal{O}(p)$ CNOT gates, either requires an extra $p-1$ ancilla qubits or requires a single ancilla qubit and c CNOT-operations with a large constant c, making our method more efficient; See 2.7.1.

2.7.5. TABULAR OVERVIEW OF COMPLEXITIES

For the ease of the reader we have provided an overview of the complexities of the different steps of our QTM as well as that of the the other quantum transport method by Todorova and Steijl (T & S) [20]. In order to provide a fair comparison of the different algorithms we have rewritten the complexities given in each paper to use the same variables.

Method	Complexity per timestep	Streaming	Reflection
S & M	$\mathcal{O}\left(n_g^2 + dN_v n_{v_{\max}}^2\right)$	$\mathcal{O}\left(dn_t^v n_{v_{\max}}^2 + dn_{g_{\max}}^2\right)$	$\mathcal{O}\left(n_w n_g^2\right)$
T & S	$\mathcal{O}\left(dN_v n_{g_{\max}}(n_g + n_{v_{\max}})^2\right)$	$\mathcal{O}\left(dN_v n_{g_{\max}}(n_g + n_{v_{\max}})^2\right)$	$\mathcal{O}\left(dN_v n_{g_{\max}}(n_g + n_{v_{\max}})^2\right)$

2.8. CONCLUSION AND OUTLOOK

Our detailed complexity analyses show that already a moderate number of a few dozens to a hundred fault-tolerant qubits suffice to solve the transport equation in 2 and 3 dimensions on a quantum computer. As our primary focus lies on algorithms that are implementable on upcoming fault-tolerant quantum computers, all algorithmic steps are optimized towards directly implementable single- and two-qubits gates, instead of theoretically more elegant but hard-to-implement multi-controlled operations that call for decomposition into native gates. Next to that we optimize the encoding of the discrete velocity to allow for a more efficient implementation of the reflection operation, the implementation of which we have made fail-safe to allow for physically correct behavior upon collision with a wall. Furthermore we have shown that our approach is significantly more efficient than state of the art quantum methods for the transport equation.

Since we are only at the start of exploring the potential of quantum computers for simulating flow problems, there are still quite some restrictions to the current method such as being limited to particles having relative velocities in each dimension smaller than 1, having a total simulated time dependent on the velocities present and using Cartesian grids where object walls are aligned with the grid. In order for the field to develop and grow future work should tackle these issues and develop algorithms without those restrictions. Another cornerstone on the way to quantum algorithms for realistic CFD applications is the treatment of nonlinear behavior, as it is typically encountered in real world flows.

Further development of quantum SDK's that allow for a straightforward implementation of the described algorithm will also be required to bring QCFD to its stage of practical implementability and allow for large scale simulations and comparisons to be performed. The final key element that needs to evolve before QCFD can reach its full potential is the fidelity of quantum hardware, as fault-tolerant computers are required for such methods but not yet available. This, of course, lies beyond the control of the computational scientists and is for the experimentalists to realise.

BIBLIOGRAPHY

[1] A-tA-v et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[2] Adriano Barenco et al. "Elementary gates for quantum computing". In: *Physical Review A.* (1995). URL: https://journals.aps.org/pra/pdf/10.1103/PhysRevA.52.3457.

[3] Ljubomir Budinski. "Quantum algorithm for the Navier-Stokes equations by using the streamfunction-vorticity formulation and the lattice Boltzmann method". In: *International Journal of Quantum information* (2021). URL: https://arxiv.org/abs/2103.03804.

[4] Steven A. Cuccaro, Thomas G. Draper, and Samuel A. Kutin. "A new quantum ripple-carry addition circuit". In: (1998). URL: https://arxiv.org/pdf/quant-ph/0008033.pdf.

[5] B. L. Douglas and J.B. Wang. "Efficient quantum circuit implementation of quantum walks". In: *Physical Review A 79, 052335* (2009). URL: https://journals.aps.org/pra/pdf/10.1103/PhysRevA.79.052335.

[6] Thomas G. Draper. "Addition on a Quantum Computer". In: (1998). URL: https://arxiv.org/pdf/quant-ph/0008033.pdf.

[7] F. Fillion-Gourdea and E. Lorin. "Simple digital quantum algorithm for symmetric first-order linear hyperbolic systems". In: *Numerical Algorithms* (2018). DOI: doi:10.1007/s11075-018-0639-3.

[8] Fraṇois Fillion-Gourdeau, Steve MacLean, and Raymond Laflamme. "Algorithm for the solution of the Dirac equation on digital quantum computers". In: *Quantum Physics* (2017). URL: https://journals.aps.org/pra/pdf/10.1103/PhysRevA.79.052335.

[9] Craig Gidney. "Factoring with $n+2$ clean qubits and $n-1$ dirty qubits". In: (2017). URL: https://arxiv.org/abs/1706.07884.

[10] Niels Gleinig and Torsten Hoefler. "An Efficient Algorithm for Sparse Quantum State Preparation". In: *2021 58th ACM/IEEE Design Automation Conference (DAC).* 2021, pp. 433–438. DOI: 10.1109/DAC18074.2021.9586240.

[11] Lov Grover and Terry Rudolph. "Creating superpositions that correspond to efficiently integrable probability distributions". In: (2002). URL: https://arxiv.org/pdf/quant-ph/0208112.pdf.

[12] Thomas Häner, Martin Roetteler, and Krysta M. Svore. "Factoring using 2n+2 qubits with Toffoli based modular multiplication". In: *Quantum Information & Computation* (2017). URL: https://arxiv.org/pdf/1611.07995.pdf.

[13] IBM. In: *https://quantum-computing.ibm.com/* (2021).

[14] Damian Musk. "A Comparison of Quantum and Traditional Fourier Transform Computations". In: *IEEE Computing in Science and Engineering, 22, 6* (2020). DOI: https://doi.org/10.1109/MCSE.2020.3023979.

[15] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information. 10th Anniversary Edition.* 4th ed. Cambridge University Press, 2016. ISBN: 9781107002173.

[16] Martin Plesch and Caslav Brukner. "Quantum State Preparation with Universal Gate Decompositions". In: *Physical Review A* (2011). URL: https://journals.aps.org/pra/abstract/10.1103/PhysRevA.83.032302.

[17] Rigetti. In: *https://www.rigetti.com/* (2022).

[18] Merel A. Schalkers and Matthias Möller. "Efficient and fail-safe quantum algorithm for the transport equation". In: *Journal of Computational Physics* 502 (2024), p. 112816. DOI: https://doi.org/10.1016/j.jcp.2024.112816.

[19] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. "Quantum Addition Circuits and Unbounded Fan-Out". In: *Quantum Information & Computation* (2009). URL: https://arxiv.org/pdf/0910.2530.pdf.

[20] B. N. Todorova and R. Steijl. "Quantum algorithm for the collisionless Boltzmann equation". In: *Journal of Computational Physics, 409, 109347* (2020). DOI: http://dx.doi.org/10.1016/j.jcp.2020.109347.

[21] Tiago M. L. de Veras, Leon D. da Silva, and Adenilton J. da Silva. "Double sparse quantum state preparation". In: *Quantum Information Processing* 21.6 (June 2022). DOI: 10.1007/s11128-022-03549-y. URL: https://doi.org/10.1007%2Fs11128-022-03549-y.

[22] Xiao Yuan Xiao-Ming Zhang Thongyang Li. "Quantum State Preparation with Optimal Circuit Depth: Implementations and Applications". In: (2022). URL: https://arxiv.org/pdf/2201.11495.pdf.

3

QUANTUM MOMENTUM EXCHANGE METHOD

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. In this chapter we present 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.

3.1. 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). An overview of the

This chapter is based on the publication Momentum exchange method for quantum Boltzmann methods by Schalkers and Möller [6].

research done on quantum computational fluid dynamcis can be found in the introduction 1.

What all the methods presented in the earlier literature 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 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. The quantum observable presented in this chapter marks the first 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 3.2. In Section 3.3 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 3.4 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 3.5 and ultimately in Section 3.6 we introduce the Quantum Momentum Exchange Method. Finally Section 3.7 is dedicated to explaining how the QMEM can be efficiently implemented in practice and Section 3.6.1 gives insight into the computational costs.

3.2. THE LATTICE BOLTZMANN METHOD

We have given an in-depth overview of the lattice Boltzmann method in the introduction.

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 Figure 3.1 we give four examples of different combinations of DdQq possible. In this chapter we are only considering the D1Q3, D2Q9 and D3Q27 cases.

We furthermore write $\boldsymbol{e}_i$ 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

$$\boldsymbol{e}_i = \begin{cases} (0,0) & \text{for } i = 0 \\ (1,0),(0,1),(-1,0),(0,-1) & \text{for } i = 1,2,3,4 \\ (1,1),(-1,1),(-1,-1),(1,-1) & \text{for } i = 5,6,7,8. \end{cases} \tag{3.1}$$

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.

3.3. MOMENTUM EXCHANGE METHOD

The momentum exchange method was proposed by Ladd [4] to determine the force acting on an object in order to calculate the drag and lift coefficients of an obstacle

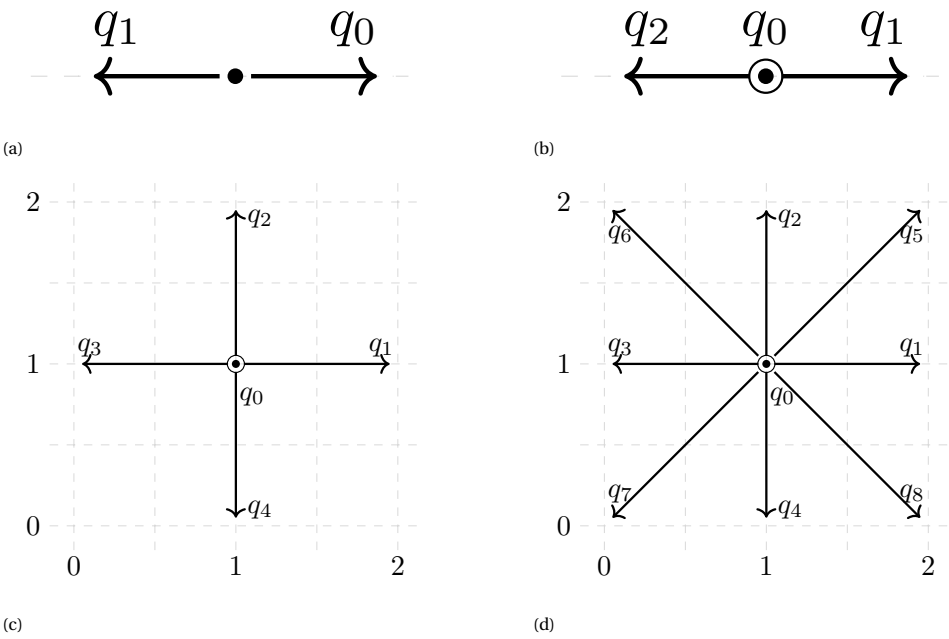


Figure 3.1: Four examples of different types of DdQq possible. Figure 3.1a portrays the D1Q2 setting and Figure 3.1b portrays the D1Q3 setting (where a stationary particle can be included). Figure 3.1c portrays the D2Q5 setting and Figure 3.1d shows the D2Q9 setting.

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 Figure 3.2. Bounce back boundary conditions are often used in combination with the Lattice Boltzmann method [3]. In Section 3.5 we give an in-depth explanation of bounce back boundary conditions as well as how to implement them in our QLBM scheme. We adopt the momentum exchange method as described in [3]. Then the force exerted on the object by the particles can be expressed as

$$\mathbf{F} = \sum_{i \in Q} \left(\boldsymbol{e}_i f_i(\mathbf{x}_f, t) - \boldsymbol{e}_{\bar{i}} f_{\bar{i}}(\boldsymbol{x}_f, t) \right). \tag{3.2}$$

In the above expression $\boldsymbol{x}_f$ refers to a point in the fluid space adjacent to the obstacle and $\boldsymbol{e}_{\bar{i}}$ represents the velocity of the particles after particles with velocity $\boldsymbol{e}_i$ 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 $\boldsymbol{e}_{\bar{i}} := -\boldsymbol{e}_i$ and $f_i(\boldsymbol{x}_f, t) = f_{\bar{i}}(\boldsymbol{x}_f, t)$ by definition, therefore we can rewrite Equation (3.2) to

$$\boldsymbol{F} = \sum_{i \in Q} 2\boldsymbol{e}_i f_i(\boldsymbol{x}_f, t). \tag{3.3}$$

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

$$F_j = \left(\sum_{i \in Q} 2\boldsymbol{e}_i f_i(\boldsymbol{x}_f, t) \right)_j. \tag{3.4}$$

3.4. 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 chapter 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 [5, 8, 1, 2], as such the measurement procedure presented here can be used with those papers.

Flow field encoding Building on our previous work [5], the quantum encoding of the discretized density function reads

$$|\underbrace{a_{n_a}\dots a_1}_{\text{ancillae}}\overbrace{g_{n_g}\dots g_1}^{\text{position}}\underbrace{v_{n_v}\dots v_1}_{\text{velocity}}\rangle\,, \tag{3.5}$$

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

$$|g_{n_g}\dots g_1\rangle = |g^d_{n_{g_d}}\dots g^d_1\, g^{d-1}_{n_{g_{d-1}}}\dots g^{d-1}_1\dots g^1_{n_{g_1}}\dots g^1_1\rangle\,, \tag{3.6}$$

where $g^j_{n_{g_j}}\dots g^j_1$ encodes the j-th 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

$$|v_{n_v}\dots v_1\rangle = |v^d\, v^d_{\text{dir}}\dots\dots v^1\, v^1_{\text{dir}}\rangle\,, \tag{3.7}$$

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 Chapter 2 where the v_{dir} qubits are grouped together, this is done simply to make the observable in Section 3.6.1 easier to visualize as a matrix. Another difference from the setup presented in Chapter 2 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 chapter 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 3.5 and 3.6, 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.

3.5. 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 obstacles and resetting them to their original position inside the flow domain [7, 3]. This is different from the specular reflection boundary conditions that we presented in Chapter 2, which reverses only the normal component of the velocity vector. Figure 3.2 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

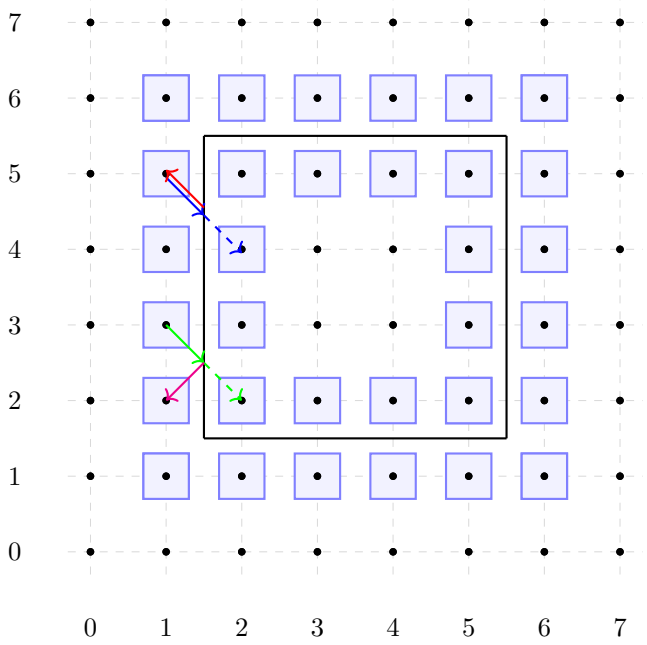


Figure 3.2: Illustration of bounce back boundary conditions (top, in red and blue arrows) versus specular reflection boundary conditions (bottom, in green and magenta).

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 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 Chapter 2. In Figure 3.4 we show how this efficient object encoding method can be implemented for the example given in Figure 3.3. In Figure 3.4 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 QFT†. The mathematics behind this procedure is explained in Section 5.4 of [5].

As a next step we flip the state of the v^j_{dir} 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

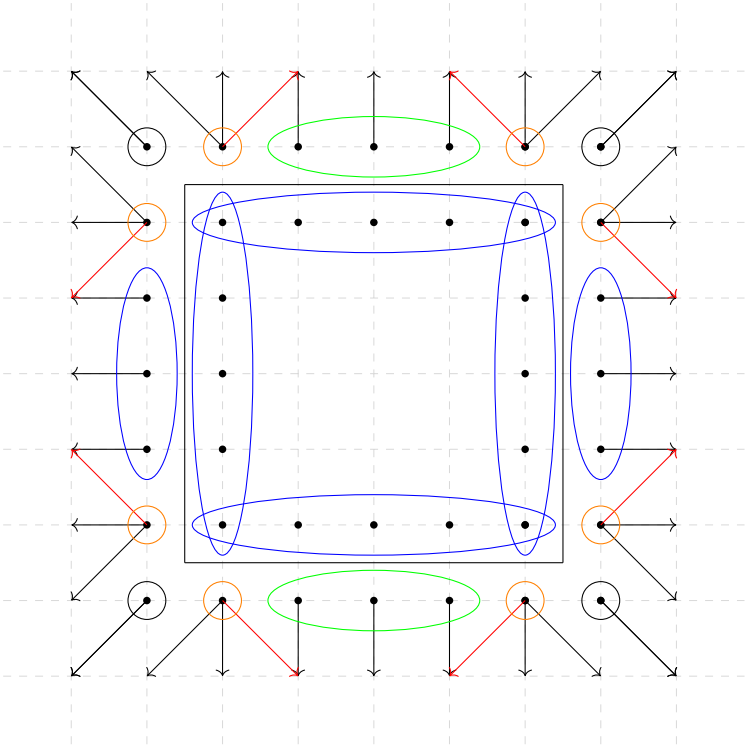


Figure 3.3: Illustration of all possible corner cases to be taken into account when particles collide with an obstacle (black box).

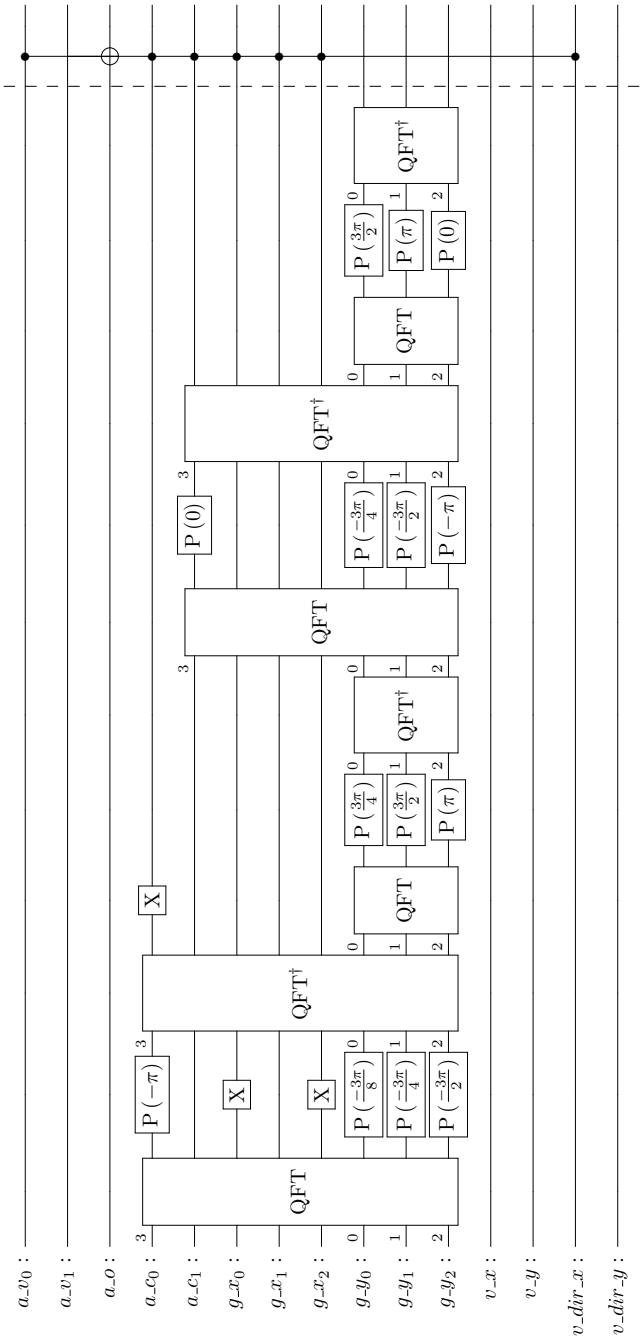


Figure 3.4: The first part of the bounce back boundary conditions, applied to the example of Figure 3.3 to properly reset the particles moving to the right in the x-direction hitting the particles on the left wall. This part of the algorithm sets the ancilla qubit, indicating that particles have virtually travelled into the object and need to have their velocity reversed and be moved out.

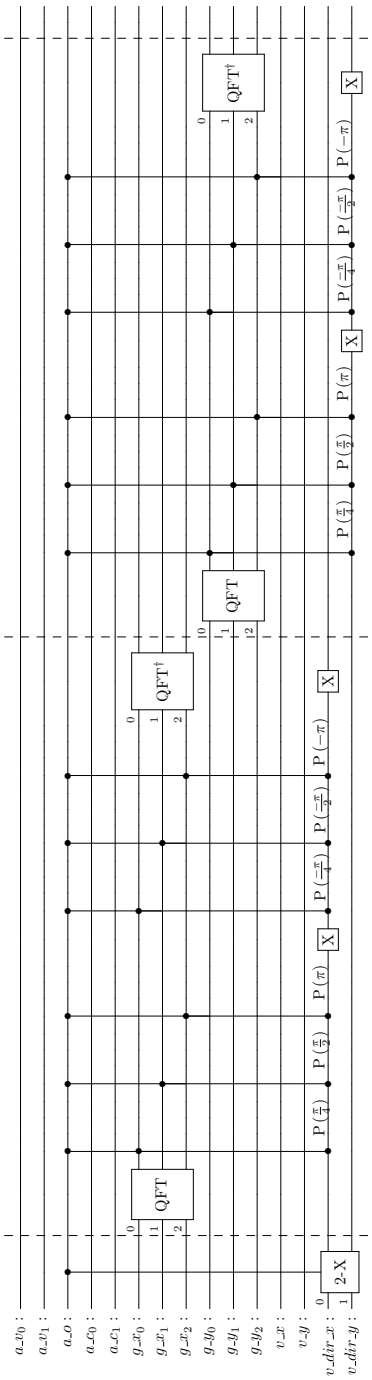


Figure 3.5: This figure shows part of the bounceback boundary conditions quantum algorithm where the velocity qubits of the particles hitting the object get reversed and subsequently moved back out of the object.

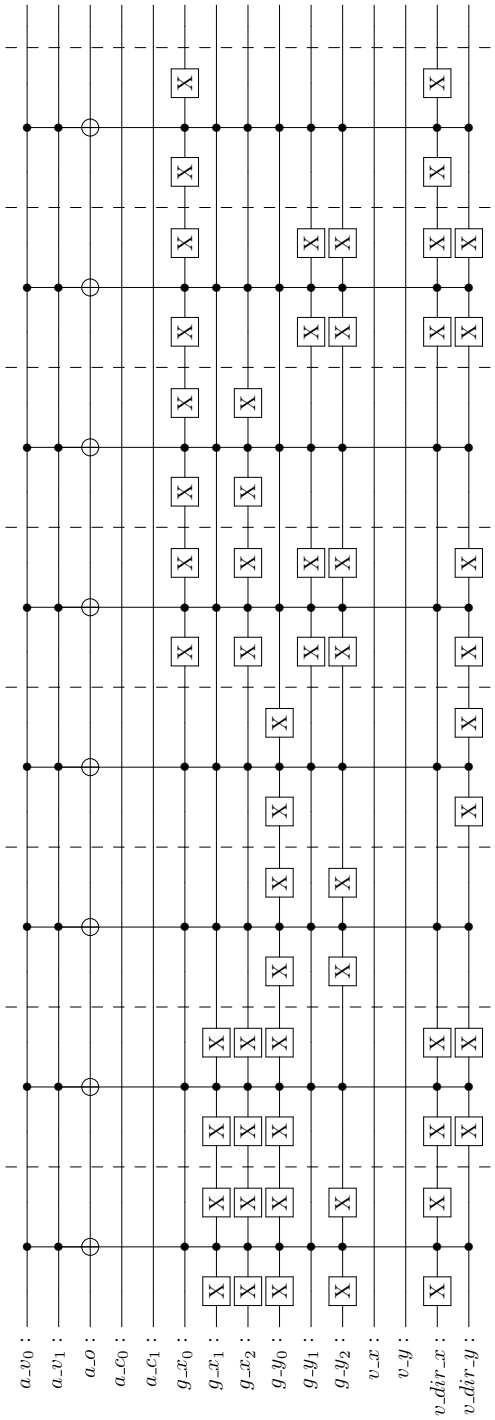


Figure 3.6: This figure shows part of the bounceback boundary conditions quantum algorithm where the ancilla qubit a_o indicating the particles that hit the object and got reversed and subsequently moved back out of the object gets reset.

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 controlled double NOT operation shown in the beginning of Figure 3.5. 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 [5] the blue and green encircled points outside of the object constitute the trivial case in Figure 3.3. 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 Figure 3.3.

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 Figure 3.3 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 Figure 3.6 we show how the ancilla qubits are reset. In this picture it can be seen that controlled on the conditions described above, an X 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 Figure 3.6.

3.6. 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 3.4. 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 Equation (3.3) and finally we describe how this method can be imple-

mented 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 [5, 8]. 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.

3.6.1. 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, equation (3.3) 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 matrix indices corresponding to 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

$$O_{\text{OME}} = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & B_{\text{OME}} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}, \tag{3.8}$$

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

$$B_{\text{OME}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.9}$$

The example given in Equation (3.8) 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 Figure 3.7. Since $B_{\text{OME}} = B_{\text{OME}}^{\dagger}$, 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.

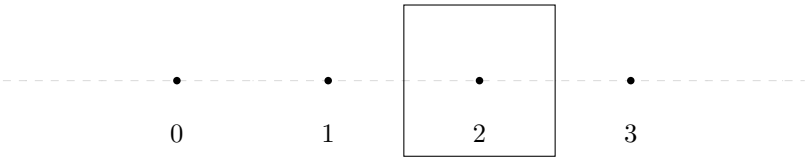


Figure 3.7: Example of the D1Q3 case with four grid points and one obstacle located on the third grid point.

For the D1Q3 case with four grid points described above we have the following quantum state encoding[1]

$$|\psi\rangle = \frac{1}{\sqrt{\sum_{x,i} f_i(\mathbf{x},t)}} \sum_{x,i} \sqrt{f_i(\mathbf{x},t)}\, |g_2 g_1 v v_{\text{dir}}\rangle\,, \tag{3.10}$$

where $g_2 g_1$ represent the binary value of the location of the x-axis, $|v v_{\text{dir}}\rangle = |10\rangle$ indicates streaming in the positive x-direction, $|v v_{\text{dir}}\rangle = |11\rangle$ indicates streaming in the negative x-direction and $|v v_{\text{dir}}\rangle = |00\rangle$ as well as $|v v_{\text{dir}}\rangle = |01\rangle$ 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 (3.10) can be written as the coefficient vector relative to the computational basis as follows:

$$|\psi\rangle = \sum_{x,v} \alpha_{x,v} |g_2 g_1 v v_{\text{dir}}\rangle = \frac{1}{\sqrt{\sum_{x,i} f_i(\mathbf{x},t)}} \begin{bmatrix} 0 \\ \sqrt{f_0(0,t)} \\ \sqrt{f_1(0,t)} \\ \sqrt{f_2(0,t)} \\ 0 \\ \sqrt{f_0(1,t)} \\ \sqrt{f_1(1,t)} \\ \sqrt{f_2(1,t)} \\ 0 \\ \sqrt{f_0(2,t)} \\ \sqrt{f_1(2,t)} \\ \sqrt{f_2(2,t)} \\ 0 \\ \sqrt{f_0(3,t)} \\ \sqrt{f_1(3,t)} \\ \sqrt{f_2(3,t)} \end{bmatrix}. \tag{3.11}$$

Using expression (3.11) and some basic linear algebra it follows that

$$\langle\psi|O_{\text{OME}}|\psi\rangle = \frac{2 f_1(1,t)}{\sum_{x,i} f_i(\mathbf{x},t)}. \tag{3.12}$$

[1] For the remainder of this chapter we leave out the superscripts indicating the dimension for notational simplicity.

Since the value of $\sum_{x,i} f_i(\mathbf{x}, t)$ is known when starting the algorithm we can simply multiply Equation (3.12) by $\sum_{x,i} f_i(\mathbf{x}, t)$ to find the value of the force we wish to calculate as described in Equation (3.3), which can subsequently be used to calculate the drag and lift coefficient following the procedure described in [4].

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 $|v^i\rangle$.

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.

3.7. 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 Subsection 3.6.1 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.

3.7.1. 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 force in the positive / negative direction. Figure 3.8 shows the quantum circuit that implements this quantum momentum exchange method for the example of Figure 3.3.

3.7.2. 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 Equation (3.3).

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

$$\begin{aligned}
&\left(\sqrt{f_{v_0}(x_0,t)}\,|\left(g_{n_g}\dots g_1\right)_0\left(v_{n_v}\dots v_1\right)_0\rangle + \dots + \right.\\
&\left.\sqrt{f_{v_1}(x_1,t)}\,|\left(g_{n_g}\dots g_1\right)_1\left(v_{n_v}\dots v_1\right)_1\rangle\right)|0\rangle_{a_{o,+}} + \\
&\left(\sqrt{f_{v_2}(x_2,t)}\,|\left(g_{n_g}\dots g_1\right)_2\left(v_{n_v}\dots v_1\right)_2\rangle + \dots + \right.\\
&\left.\sqrt{f_{v_3}(x_3,t)}\,|\left(g_{n_g}\dots g_1\right)_3\left(v_{n_v}\dots v_1\right)_3\rangle\right)|1\rangle_{a_{o,+}},
\end{aligned} \tag{3.13}$$

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 $|\left(g_{n_g}\dots g_1\right)_0\left(v_{n_v}\dots v_1\right)_0\rangle + \dots + |\left(g_{n_g}\dots g_1\right)_1\left(v_{n_v}\dots v_1\right)_1\rangle$ 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 $|\left(g_{n_g}\dots g_1\right)_2\left(v_{n_v}\dots v_1\right)_2\rangle + \dots + |\left(g_{n_g}\dots g_1\right)_3\left(v_{n_v}\dots v_1\right)_3\rangle$ 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

$$f_{v_2}(x_2,t) + \dots + f_{v_3}(x_3,t), \tag{3.14}$$

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.

3.8. CONCLUSION

In this chapter we have presented a quantum approach to determine the force of the flow field acting on an object immersed in the fluid. The method is easily implementable and shows that a measurement can be done efficiently when only considering the force acting on an object as the quantity of interest. The number of measurements needed

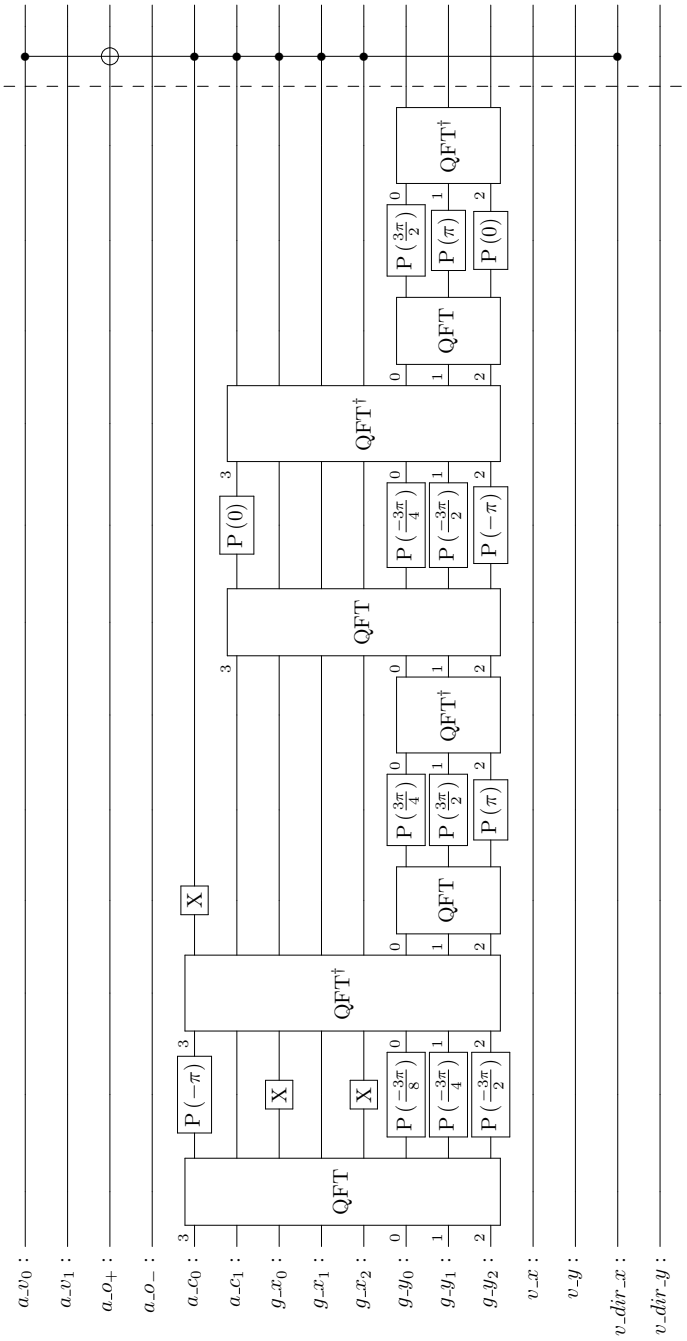


Figure 3.8: The quantum algorithm for the QMEM, applied to the example of Figure 3.3 to determine the force of the particles moving to the right in the x-direction hitting the particles on the left wall. This part of the algorithm sets the ancilla qubit, indicating that particles have virtually travelled into the object. Subsequently the ancilla qubit $a_{o,+}$ is measured to determine the force.

depends on the relative size of the object, making it only non-exponential and therefore efficient in cases where the object is not exponentially small relative to the global grid. 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.

BIBLIOGRAPHY

[1] Ljubomir Budinski. "Quantum algorithm for the advection-diffusion equation simulated with the lattice Boltzmann method". In: *Quantum Information Processing 2021* (2020). URL: https://link.springer.com/article/10.1007/s11128-021-02996-3.

[2] Ljubomir Budinski. "Quantum algorithm for the Navier-Stokes equations by using the streamfunction-vorticity formulation and the lattice Boltzmann method". In: *International Journal of Quantum information* (2021). URL: https://arxiv.org/abs/2103.03804.

[3] Timm Krüger et al. *The lattice Boltzmann method*. Springer, 2017. URL: https://link.springer.com/book/10.1007/978-3-319-44649-3.

[4] Anthony J. Ladd. "Numerical Simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation". In: *Journal of Fluid Mechanics* 271 (1993), pp. 285–309.

[5] Merel A. Schalkers and Matthias Möller. "Efficient and fail-safe quantum algorithm for the transport equation". In: *Journal of Computational Physics* 502 (2024), p. 112816. DOI: https://doi.org/10.1016/j.jcp.2024.112816.

[6] Merel A. Schalkers and Matthias Möller. "Momentum exchange method for quantum Boltzmann methods". In: *Computers & Fluids* 285 (2024), p. 106453. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2024.106453.

[7] Ulf D. Schiller. "Thermal fluctuations and boundary conditions in the lattice Boltzmann method". PhD thesis. Johannes Gutenberg-Universität, Jan. 2008.

[8] B. N. Todorova and R. Steijl. "Quantum algorithm for the collisionless Boltzmann equation". In: *Journal of Computational Physics, 409, 109347* (2020). DOI: http://dx.doi.org/10.1016/j.jcp.2020.109347.

4

On the importance of data encoding in quantum Boltzmann methods

In recent years, quantum Boltzmann methods have gained more and more interest as they might provide a viable path towards solving fluid dynamics problems on quantum computers once fault-tolerant many-qubit systems become available. The major challenge in developing a start-to-end quantum algorithm for the Boltzmann equation consists in encoding relevant data efficiently in quantum bits (qubits) and formulating the streaming, collision and boundary conditions as one comprehensive unitary operation. The current literature on quantum Boltzmann methods mostly proposes data encodings and quantum primitives for individual phases of the pipeline assuming that they can be combined to a full algorithm.

In this chapter we disprove this assumption by showing that for encodings commonly discussed in literature either the collision or the streaming step cannot be unitary. We furthermore describe the intuition behind this impossibility and identify possible methods to work around it.

4.1. Introduction

Quantum Boltzmann methods (QBM) have been widely researched and developed over the past years. What remained an open problem is the development of a full-fledged QBM that implements both the streaming and the collision step as unitary operations. In this chapter we prove rigorously that for the encoding schemes considered for universal quantum computers in all previous publications it is impossible to implement both streaming and collision as a unitary, downgrading them as candidates for any practical

This chapter is based on the publication On the importance of data encoding in quantum Boltzmann methods by Schalkers and Möller [7].

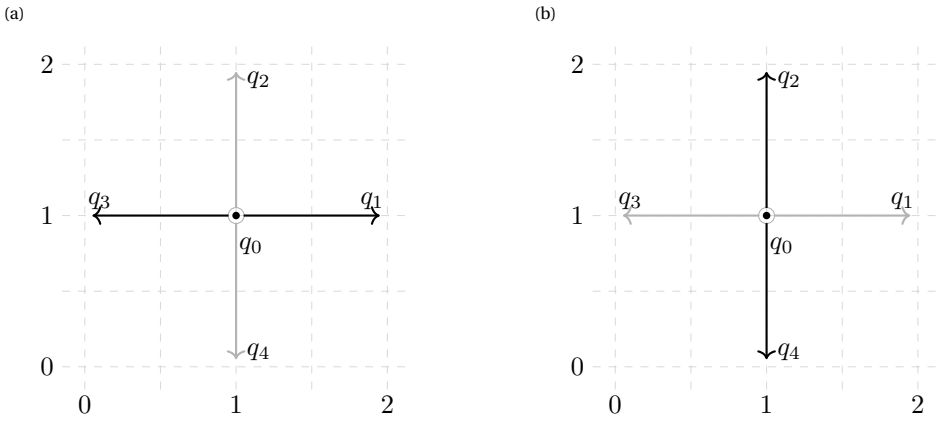


Figure 4.1: Illustration of two velocity combinations of the D2Q5 (and D2Q4) velocity spectrum that belong to the same equivalence class with total momentum 0 and mass 2: 4.1a particles streaming in the q_1 and q_3 direction, and 4.1b particles streaming in the q_2 and q_4 direction.

QBM. The proves we provide are general and applicable to both lattice Boltzmann and lattice gas models. This chapter aims to help researchers gain insight in why certain encoding methods cannot be used to solve the full Boltzmann equation and stimulate a paradigm shift in QBM research from focusing on encodings and algorithms for individual steps of the pipeline to developing full-fledged QBM algorithms.

4.2. DATA ENCODING

As in any computational field, data encoding is pivotal for reaching a good result. More than five decades of classical CFD research and application have established 'good practices' for storing field data such as densities and velocities at, e.g., the grid points or cell centers as floating-point numbers following the IEEE-754 standard. Every now and then new hardware developments stimulate research into non-standard formats, like reduced or mixed-precision [3], but, in general, data encoding is not considered to be an open problem.

Not so in QCFD and, in particular, quantum Boltzmann methods where different encoding methods are used in different papers. The two mainstream encodings of the velocity vector are the amplitude based encoding [9, 1, 2, 6] and the computational basis state encoding [10, 11, 13, 12, 5, 4, 8]. In this section we will review the main data encodings currently used for QBM and show that in all of them either the streaming step or the collision step cannot be unitary. This result, though discouraging at first sight, should be interpreted as wake-up call that novel quantum encodings for CFD states are imperative for devising full-fledged QCFD applications in the future. We propose one such novel encoding in Chapter 5 and discuss its potential and limitations.

4.2.1. AMPLITUDE BASED ENCODING

The first type of encoding we consider is the so-called amplitude based encoding, used for several quantum Boltzmann methods [9, 1, 2, 6]. The amplitude based encoding of the velocity vector is such that at each location $|x\rangle$ there can be multiple particles with different velocities, for instance $|v_0\rangle$, $|v_1\rangle$, $|v_2\rangle$ and $|v_3\rangle$ for D2Q4. Here and below, $|i\rangle$ denotes the representation of i as bit string. The state of the system at this point x can then be encoded as[1]

$$|x\rangle\left(\alpha_0|v_0\rangle+\alpha_1|v_1\rangle+\alpha_2|v_2\rangle+\alpha_3|v_3\rangle\right), \tag{4.1}$$

where α_0, α_1, α_2 and α_3 are complex numbers that simply represent the relative weight or number of particles traveling at the given velocity at grid point x. For simplicity we will assume that $|\alpha_0|^2+|\alpha_1|^2+|\alpha_2|^2+|\alpha_3|^2=1$, and so in this example there are only particles at grid point x but the proof extends trivially to the general case with particles spread around the grid.

In order to show that this encoding of the velocity vector inevitably leads to non-unitary collision operators, let us first take a close look at what is required from a collision operator. A collision operator U_{col} needs to map the velocities of the incoming particles to a so-called equivalent outgoing state. Two states are equivalent if the total mass and momentum of all particles combined are the same. For example having one particle of mass 1 traveling in the positive x direction and one particle of mass 1 in the negative x direction is equivalent to one particle of mass 1 traveling in the positive y direction and one particle of mass 1 in the negative y direction, as for both states the total momentum is 0 and the mass is the same. This means that if there is only one particle with one direction at a specific point in space, the collision operator cannot change this as there is no other direction this one particle could be traveling with the same momentum. Using these requirements we can set up a generic collision operation U_{col} that solely meets the basic requirements of behavior it needs to portray. The first requirement is that there should be at least a combination of incoming velocity states that leads to a new combination containing at least one velocity state that was previously not present. Let the state $|\psi_1\rangle$ be an example of an incoming state for which a state in its equivalence class includes at least some velocity which is not included in the original state. Without loss of generality assume that $|\psi_1\rangle$ consists of two different velocity states $|v_0\rangle$ and $|v_1\rangle$, meaning that $|\alpha_0|,|\alpha_1|>0$ and $\alpha_2=\alpha_3=0$. Then we can write the state of the system as

$$|\psi_1\rangle=|x\rangle\left(\alpha_0|v_0\rangle+\alpha_1|v_1\rangle\right). \tag{4.2}$$

Now assume that an equivalent velocity combination exists consisting of particles traveling with velocities $\beta_2|v_2\rangle+\beta_3|\psi_3\rangle$, where we have $|\beta_2|,|\beta_3|>0$ and we let $|\psi_3\rangle$ be any combination of all basis states except $|v_2\rangle$. To realize this potential outcome of a collision as a quantum algorithm, we need to implement the transformation between both equivalent states as a unitary operation U_{col} which changes the states of the velocity en-

[1] Note that we distinguish in our notation between the grid point x and its representation $|x\rangle$ as part of the quantum register.

codings as follows

$$\begin{aligned} |\psi_1'\rangle &= I \otimes U_{\text{col}} |\psi_1\rangle \\ &= |x\rangle \otimes U_{\text{col}} (\alpha_0 |v_0\rangle + \alpha_1 |v_1\rangle) \\ &= |x\rangle \left(\gamma_0(\alpha_0 |v_0\rangle + \alpha_1 |v_1\rangle) + \gamma_1(\beta_2 |v_2\rangle + \beta_3 |\psi_3\rangle)\right). \end{aligned} \tag{4.3}$$

Here, if $\gamma_0 = 1$ and $\gamma_1 = 0$ no collision is taking place (and we simply implement an identity operation) and if $\gamma_1 = 1$ we fully change from the original velocities to its alternative representative from the same equivalence class.[2] Note that to preserve unitarity $|\gamma_0|^2 + |\gamma_1|^2 = 1$ must hold.

Let us now consider another system in state $|\psi_2\rangle = |x\rangle |v_2\rangle$. Applying the unitary operation U_{col} should not effect the state at all as a single speed is only in an equivalence class with itself, and so the required behavior for U_{col} is

$$\begin{aligned} |\psi_2'\rangle &= I \otimes U_{\text{col}} |\psi_2\rangle \\ &= |x\rangle U_{\text{col}} |v_2\rangle \\ &= e^{i\theta} |x\rangle |v_2\rangle, \end{aligned} \tag{4.4}$$

with $\theta \in (0, 2\pi]$. That is, the collision operator must preserve the single-velocity state except for changes in the phase factor $e^{i\theta}$ that can be neglected.

Now that we have identified the required behavior for U_{col} to implement a collision operation, we can prove that any U_{col} that meets both requirements simultaneously cannot be unitary. Here, we resort to the characterization $U_{\text{col}}^\dagger U_{\text{col}} = I$ of unitary operators, with superscript † denoting the adjoint operator.

Proof. To reach a contradiction, assume that U_{col} is a unitary operator. Then it must preserve the inner product for all possible states $|\phi_1\rangle$ and $|\phi_2\rangle$

$$\langle \phi_1 | \phi_2 \rangle = \langle \phi_1 | U_{\text{col}}^\dagger U_{\text{col}} | \phi_2 \rangle. \tag{4.5}$$

However, for a collision operation U_{col} that behaves as expected on the system states described in Equations (4.2) to (4.4), it follows that

$$\begin{aligned} 0 &= \langle \psi_1 | \psi_2 \rangle \\ &= \langle \psi_1 | (I \otimes U_{\text{col}})^\dagger (I \otimes U_{\text{col}}) | \psi_2 \rangle \\ &= e^{i\theta} \left(\gamma_0(\alpha_0 \langle v_0| + \alpha_1 \langle v_1|) + \gamma_1(\beta_2 \langle v_2| + \beta_3 \langle \psi_3|)\right) \langle x | x \rangle |v_2\rangle \\ &= e^{i\theta} \gamma_1 \beta_2. \end{aligned} \tag{4.6}$$

The first equality follows from the fact that $|\psi_1\rangle$ and $|\psi_2\rangle$ are orthogonal by construction. The second one holds under the assumption of U_{col} being unitary, which is disproved by the fact that the entire equality chain only holds for the trivial case $\gamma_1 = 0$ (as $|\beta_2| > 0$ by definition of the state $|\psi_1\rangle$), that is, when U_{col} does not implement the collision operation. From this we can conclude that an amplitude based encoding of the velocity does not allow for a unitary implementation of the collision operation. □

[2] Here γ_0, γ_1 are chosen to reflect the fact that a collision operation should switch weight of a combination of velocities in an equivalent class to another combination of velocities in the same equivalence class. This equation could be written in a less restrictive way by splitting γ_0 and γ_1 up into separate amplitudes γ_i for all the basis states $|v_i\rangle$, the same contradiction of unitarity as presented below however could be reached.

Notice that this proof works for any amplitude based encoding of v where the different possible velocities at a position are all represented by their own basis state as there will always be a case with only a single incoming velocity, for which an identity operation up to a phase shift should take place, while at the same time there will be combinations of velocities for which we want some weight of the system to change from one combination of velocities to another combination of velocities in the same equivalence class. These two antagonizing requirements will always lead to the same contradiction of unitarity proven above and we further expand on this intuition in Section 4.2.3.

4.2.2. COMPUTATIONAL BASIS STATE ENCODING

The second type of encoding of a quantum state considered is the computational basis encoding, used in several quantum lattice Boltzmann papers such as [10, 11, 13, 12, 5, 4, 8]. Using this encoding the contradiction of unitarity in the collision operation can be avoided by encoding the velocity of the qubits at a position $|x\rangle$ in space by identifying each direction particles could be streamed from with its own qubit, which will be set to one if and only if there is a particle streaming from that direction.

As an example consider the D2Q4 lattice depicted in Figure 4.2. In this case the velocity can be encoded using four qubits q_0, q_1, q_2 and q_3 where the state

$$|x\rangle|v\rangle = |x\rangle|q_0 q_1 q_2 q_3\rangle = |x\rangle|0110\rangle \tag{4.7}$$

is such that from the center point $(1,1)$, there is a particle streaming to $(1,2)$ and a particle streaming to $(0,1)$ but not to $(2,1)$ or $(1,0)$.

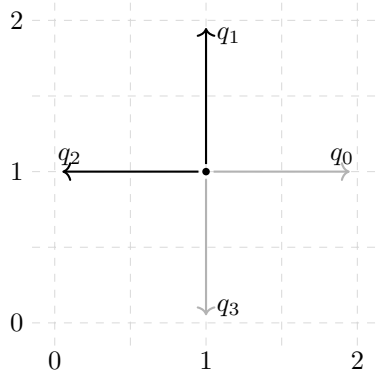


Figure 4.2: Illustration of the computational basis state encoding for the D2Q4 lattice. For each grid point x we set the respective qubit q_j to one if and only if there is a particle streaming in that direction, i.e. $|v\rangle = |q_0 q_1 q_2 q_3\rangle = |0110\rangle$.

Using this encoding the collision step can be defined quite naturally as unitary operation. However, we run into trouble when attempting to define a unitary streaming step U_{str} as we demonstrate in what follows.

To simplify notation let us restrict ourselves to the D1Q2 lattice and consider the two

settings at time t from Figures 4.3 and 4.4, which can be encoded as

$$|\psi_1\rangle = \sum_{x=0}^{3} |x\rangle\,|v\rangle \tag{4.8}$$

$$= \frac{1}{2}\left(|00\rangle\,|00\rangle + |01\rangle\,|11\rangle + |10\rangle\,|10\rangle + |11\rangle\,|10\rangle\right), \tag{4.9}$$

and

$$|\psi_2\rangle = \frac{1}{2}\left(|00\rangle\,|01\rangle + |01\rangle\,|01\rangle + |10\rangle\,|00\rangle + |11\rangle\,|11\rangle\right), \tag{4.10}$$

respectively. It then follows directly that

$$\langle\psi_1|\psi_2\rangle = 0. \tag{4.11}$$

Upon streaming, the systems from Figures 4.3 and 4.4 change from their state at time t (top lattice) to that at time $t+1$ (bottom lattice), i.e.

$$|\psi_1'\rangle = \frac{1}{2}\left(|00\rangle\,|11\rangle + |01\rangle\,|00\rangle + |10\rangle\,|10\rangle + |11\rangle\,|10\rangle\right), \tag{4.12}$$

and

$$|\psi_2'\rangle = \frac{1}{2}\left(|00\rangle\,|11\rangle + |01\rangle\,|00\rangle + |10\rangle\,|01\rangle + |11\rangle\,|01\rangle\right), \tag{4.13}$$

respectively. As in the previous section, we will show by contradiction that any operation U_{str} for which $U_{\text{str}}\,|\psi_1\rangle = |\psi_1'\rangle$ and $U_{\text{str}}\,|\psi_2\rangle = |\psi_2'\rangle$ cannot be unitary.

Proof. Let us assume that U_{str} is unitary, i.e. it preserves the inner product

$$\langle\phi_1|\phi_2\rangle = \langle\phi_1|\,U_{\text{str}}^{\dagger}U_{\text{str}}\,|\phi_2\rangle\,, \tag{4.14}$$

for all states $|\phi_1\rangle\,, |\phi_2\rangle$. Substituting the states (4.9) and (4.10) on the left side, and (4.12) and (4.13) into the right inner product we arrive at the contradiction

$$0 = \langle\psi_1|\psi_2\rangle = \langle\psi_1|\,U_{\text{str}}^{\dagger}U_{\text{str}}\,|\psi_2\rangle = \langle\psi_1'|\psi_2'\rangle = \frac{1}{2}. \tag{4.15}$$

The first equality follows from the orthogonality property (4.11), and the second one from the assumption that U_{str} is a unitary operator, which we just disproved. □

As in Section 4.2.1 this proof extends to any computational basis encoding where each possible combination of velocities at a specific lattice point is encoded using its own basis state, as one can always construct two situations with no overlap at time t that will have non-zero overlap after streaming at time $t+1$. This proof also extends trivially to any other DnQm setting as the streaming possibilities of D1Q2 are essentially a subset of any other system and thus the same example can be used by setting the other streaming directions to 0.

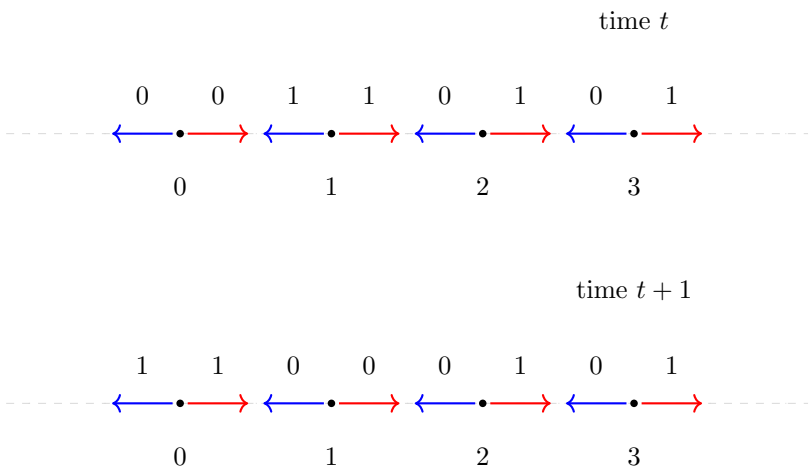


Figure 4.3: D1Q2 example setting 1. The binary encoding above the arrows indicate whether or not particles are flowing there in that time step. 1 indicates that there are particles there and 0 indicates that there are no particles present. In the example setting we consider periodic boundary conditions. The top figure shows the state of the system at time t. The figure below shows the state of the system at time $t+1$.

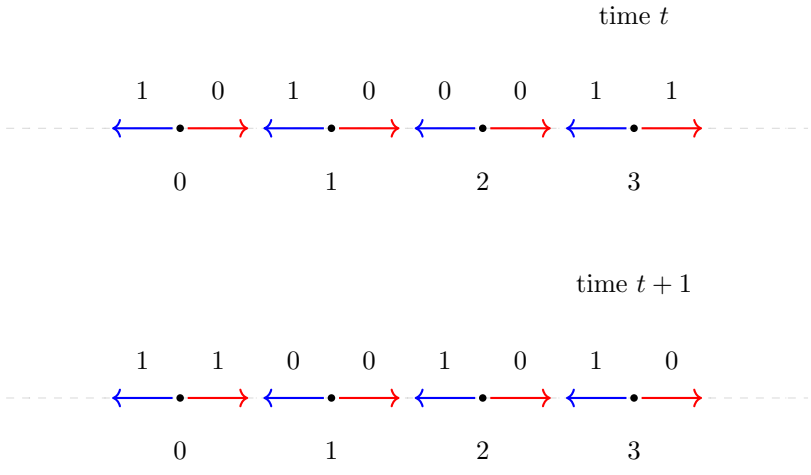


Figure 4.4: D1Q2 example setting 2, the binary encoding above the arrows indicate whether or not particles are flowing there in that time step. 1 indicates that particles are streaming there and 0 indicates that are no particles present. In the example setting we consider periodic boundary conditions. The top figure shows the state of the system at time t. The figure below shows the state of the system at time $t+1$.

4.2.3. INTUITION AND EXTENSION OF NON-UNITARITY PROOFS

In this section we expand on our non-unitarity proofs by providing physical intuition behind the proofs presented above. It is intended to give insight into what types of encodings our non-unitarity proof extends to, and what physical features of the system necessarily lead to the non-unitarity for these encodings.

Consider the proof from Section 4.2.1 that shows that the amplitude based encoding, where each velocity direction is identified through its own basis state leaving the total velocity at a position x to be a superposition of such basis states, prevents the collision operator U_{col} from being unitary. Since it encodes each streaming direction as a different basis state, the quantum encodings of the velocity directions are all orthogonal to one another. This is also necessary, since if the basis states of the possible streaming directions are not orthogonal, we cannot fully distinguish between them. However, this orthogonality of the different velocity directions leads directly to the non-unitarity of U_{col}. Since a collision operator that will rotate a given linear combination of basis states into a linear combination of other basis states in such a way that the represented streaming patterns belong to the same equivalence class, it will also rotate 'pure' velocities represented by a single basis state into another basis state leading to a nonphysical and undesired change of velocities.

Following this line of argumentation it can be seen that the non-unitarity of U_{col} is not so much a result of a specific choice of encoding but an inherent non-unitarity of the collision step itself that directly leads to the idea of computational basis state encoding, where each velocity pattern (i.e. the combination of velocities) at a grid point is encoded as its own basis state, and not as a unitary combination of all the basis states representing a non-zero contribution.

When encoding the velocity pattern at each grid point as a basis state, naturally, the non-unitarity of collision falls away and we can find a straightforward unitary operator to implement the collision step. However, such an encoding will always lead to non-unitarity of streaming due to the non-local nature of a streaming operation. Consider an arbitrary point in space x and imagine two different scenarios with two different combinations of speeds $|v_1\rangle$ and $|v_2\rangle$ at this point. Then the inner product between $|x\rangle|v_1\rangle$ and $|x\rangle|v_2\rangle$ must be 0, as these are different basis states. However, the velocity states of the systems at position x in the next time step do not depend on the current velocity states in the lattice point. In fact, they only depend on the velocity states of the neighboring lattice points. Since the inner product of the states at the point x at the next time step does not depend on the current states at the point x, in the next time step the velocity at the point x of the two systems could be identical, and hence, the inner product could be one. There is no way of ensuring that this can only happen when the inner product at some other point x' of the systems was non-zero before as each grid point has velocity vectors in multiple directions determining its associated velocity basis state.

This shows that any quantum encoding that successfully implements both streaming and collision as a unitary operation must belong to one of the following three types. The first type is an amplitude based type encoding, where the different velocities are not orthogonal and thus not entirely distinguishable. The second type is a computational basis state encoding where the non-locality of streaming is somehow avoided. The last type is a completely novel encoding method that avoids both non-unitarity problems

entirely. In Chapter 5 we will present precisely one such idea.

4.3. CONCLUSION

In this chapter we have shown that data encoding methods considered previously for quantum Boltzmann methods do not allow for treating both streaming and collision as unitary quantum operations. We have provided both a mathematical proof of its impossibility, and insight into the physical properties of the system and encodings that lead to this behavior.

BIBLIOGRAPHY

[1] Ljubomir Budinski. "Quantum algorithm for the advection-diffusion equation simulated with the lattice Boltzmann method". In: *Quantum Information Processing 2021* (2020). URL: https://link.springer.com/article/10.1007/s11128-021-02996-3.

[2] Ljubomir Budinski. "Quantum algorithm for the Navier-Stokes equations by using the streamfunction-vorticity formulation and the lattice Boltzmann method". In: *International Journal of Quantum information* (2021). URL: https://arxiv.org/abs/2103.03804.

[3] Gabriel Freytag et al. "Impact of Reduced and Mixed-Precision on the Efficiency of a Multi-GPU Platform on CFD Applications". In: *Lecture Notes in Computer Science* (2022). URL: https://link.springer.com/chapter/10.1007/978-3-031-10542-5_39.

[4] Y. Moawad, W. Vanderbauwhede, and R. Steijl. "Investigating hardware acceleration for simulation of CFD quantum circuits". In: *Frontiers in Mechanical Engineering* (2022). DOI: 10.3389/fmech.2022.925637. URL: https://www.frontiersin.org/articles/10.3389/fmech.2022.925637/full.

[5] Marco A. Pravia et al. "Experimental demonstration of Quantum Lattice Gas Computation". In: *Quantum Information Processing* (2003). URL: https://link.springer.com/article/10.1023/A:1025835216975.

[6] Merel A. Schalkers and Matthias Möller. "Efficient and fail-safe quantum algorithm for the transport equation". In: *Journal of Computational Physics* 502 (2024), p. 112816. DOI: https://doi.org/10.1016/j.jcp.2024.112816.

[7] Merel A. Schalkers and Matthias Möller. "On the importance of data encoding for quantum Boltzmann methods". In: *Quantum Information Processing* (2024). DOI: https://doi.org/10.1007/s11128-023-04216-6.

[8] René Steijl. "Quantum Circuit Implementation of Multi-Dimensional Non-Linear Lattice Models". In: *MDPI: Applied Sciences* (2023). DOI: https://doi.org/10.3390/app13010529. URL: https://www.mdpi.com/2076-3417/13/1/529.

[9] B. N. Todorova and R. Steijl. "Quantum algorithm for the collisionless Boltzmann equation". In: *Journal of Computational Physics, 409, 109347* (2020). DOI: http://dx.doi.org/10.1016/j.jcp.2020.109347.

[10] Jeffrey Yepez. "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.

[11] Jeffrey Yepez. “Quantum Lattice-Gas Model for computational fluid dynamics”. In: *Physical Review E* (2001). URL: https://journals.aps.org/pre/abstract/10.1103/PhysRevE.63.046702.

[12] Jeffrey Yepez. “Quantum Lattice-Gas Model for the Burgers Equation”. In: *Journal of statistical physics* (2002). URL: https://link.springer.com/article/10.1023/A:1014514805610.

[13] Jeffrey Yepez and Bruce Boghosian. “An efficient and accurate quantum lattice-gas model for the many-body Schrödinger wave equation”. In: *Computer Physics Communications* (2001). URL: https://www.phys.hawaii.edu/~yepez/static/papers/pdf/2002CompPhysCommVol146No3%2C15Pg280.pdf.

5

THE SPACE-TIME METHOD

In the previous chapter we provided mathematical proofs that show that a unitary treatment of both streaming and *collision is impossible with the encodings adopted in current quantum Boltzmann literature, thereby underpinning the need for a new encoding. In this chapter we describe an encoding method, the so-called space-time encoding that can be used to solve the equations of fluid dynamics on a quantum computer by using a lattice gas model approach. In this encoding the number of qubits used to encode the velocity depends on the number of time steps one wishes to simulate, with the upper bound depending on the total number of grid points.*

In light of the non-unitarity result established for existing encodings, our encoding method is to the best of our knowledge the only one currently known that can be used for a start-to-end quantum solver using a lattice based discrete velocity method where both the collision and the streaming step are implemented as a unitary operation.

5.1. INTRODUCTION

As described in the previous chapter, the previously known quantum Boltzmann methods all used encoding methods that did not allow for the full Boltzmann equation to be implemented and solved directly on a quantum computer.

There are existing methods with a work-around for this issue, which usually comes in the form of a stop-and-go algorithm. Here it is assumed that at the end of each timestep the system is measured and re-initialized for the next timestep. The drawback of this method is two-fold, first of all upon measurement the wrong states might be found which means the circuit has to run again until the right values are found. The second issue is that even if the right states are found upon measurement, the system collapses and has to be re-initalized before the next time-step can be run. In practice this is extremely expensive since re-initialization and measurement of the quantum circuits are very costly as described in Section 1.3.

This chapter is based on the publication On the importance of data encoding in quantum Boltzmann methods by Schalkers and Möller [2].

All aforementioned strategies require a stop-and-go approach, what remained an open problem is the development of a full-fledged quantum discrete velocity lattice based method that implements the streaming and the collision steps as unitary operations. In this chapter we present the first-of-its-kind full-fledged discrete velocity lattice based quantum method, our approach builds on a novel encoding scheme, called space-time encoding which we introduce in this chapter.

5.2. LATTICE GAS VS LATTICE BOLTZMANN

In Chapter 1 we have described both lattice gas and lattice Boltzmann methods. The main differences are in the interpretation between single particles that either are or are not present in lattice gas models and relative particle densities in lattice Boltzmann methods. This difference in interpretation has its main effect in the collision operation. Where lattice gas models have rather simple rule based collision operations, lattice Boltzmann models can allow for a more complex collision where the particles densities are spread out over multiple possible basis states. These collision operators are determined using the equilibrium function (1.26), which requires the macroscopic quantities ρ and $\boldsymbol{u}$. Classically this can be calculated and used in each time step. Since we are working on a quantum computer we cannot determine these without measuring and subsequently collapsing the system. Due to the high computational costs of measurement and reinitialization, we focus on methods that do not require reinitilization. As such we do not have a way to calculate the necessary values for determining the equilibrium function. We will therefore use a lattice gas inspired method for our space-time encoding, as the collision operation can be pre-determined for all possible input configurations; see Section 5.4.

5.3. DATA ENCODING

In what follows, we adopt an extended computational basis state encoding, where at each location x we take into account the velocities at all grid points in the vicinity of x. Here, 'in the vicinity of x' means that a particle can theoretically reach the grid point x within the number of time steps still to be performed before measurement. Mathematically speaking being 'in the vicinity of x' means being, respectively, in the so-called extended von Neumann, Moore or hexagonal neighborhood of the point x, depending on the lattice structure.

This leads to a trade-off between the number of time steps that can be performed between measurements and the number of qubits required to encode the velocity at each grid point x. The more time steps one wishes to take between measurement-and-reinitialization cycles, the more qubits are required for our space-time encoding. Obviously the maximum number of qubits required to implement the velocity without any in-between measurements must be such that the entire grid is spanned. For a DnQm lattice this will be mN_g, where N_g is the total number of grid points. When encoding the proposed method on a classical computer mN_g bits would also be required, so when encoding the full domain there is no quantum benefit in terms of (qu)bit numbers. The quantum improvement comes from exploiting quantum parallelism, which is done as long as we do not encode the whole space.

In what follows, let N_t denote the number of streaming steps to be performed between (re-)initialization and measurement. We extend the computational basis state encoding of velocity directions from Section 4.2.2 to take into account all the speed states from grid points in the neighborhood of x that can (at least theoretically) reach x within N_t streaming steps. This takes away the non-locality of the streaming operator, which led to the non-unitarity of U_{str} for the 'regular' computational basis state encoding at the cost of increasing the number of qubits required to encode all required velocity data.

We will give a detailed description of this encoding for the D2Q4 lattice, but want to note that it can be extended naturally to any other choice of DnQm. Consider the D2Q4 lattice given in Figure 5.1 with qubit q_j set to one if and only if there is a particle traveling with velocity direction j from grid point x into a neighboring grid point in the current time step. We now extend this encoding to include *all* possible velocities at positions 'in the vicinity of x' for the total of N_t time steps in order to obtain a unitarily streamable encoding. This is illustrated in Figure 5.2 for a single time step, i.e. $N_t = 1$ yielding the encoding

$$|x\rangle\,|q_{19}\,q_{18}\ldots q_0\rangle\,. \tag{5.1}$$

For D2Q4, the number of qubits encoding the possible velocity states per grid location x grows with the number of time steps (still) to be taken as

$$n_v = 4 + \sum_{i=1}^{N_t} 16i = 8N_t^2 + 8N_t + 4, \tag{5.2}$$

where the maximum number of qubits required to encode all velocity directions over the entire grid equals $4N_g$ as stated before.[1] Similarly it can be shown that for d dimensions the growth rate is of the order $\mathcal{O}\left(N_t^d\right)$.

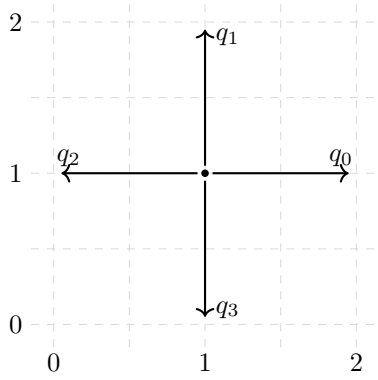


Figure 5.1: Illustration of the computational basis state encoding for D2Q4.

[1] Note that the growth rate of qubit numbers per time step depends on the choice of DnQm. The number of qubits required is equal to the number of points in the extended Von Neumann, Moore or hexagonal neighborhood, depending on which choice of n and m considered.

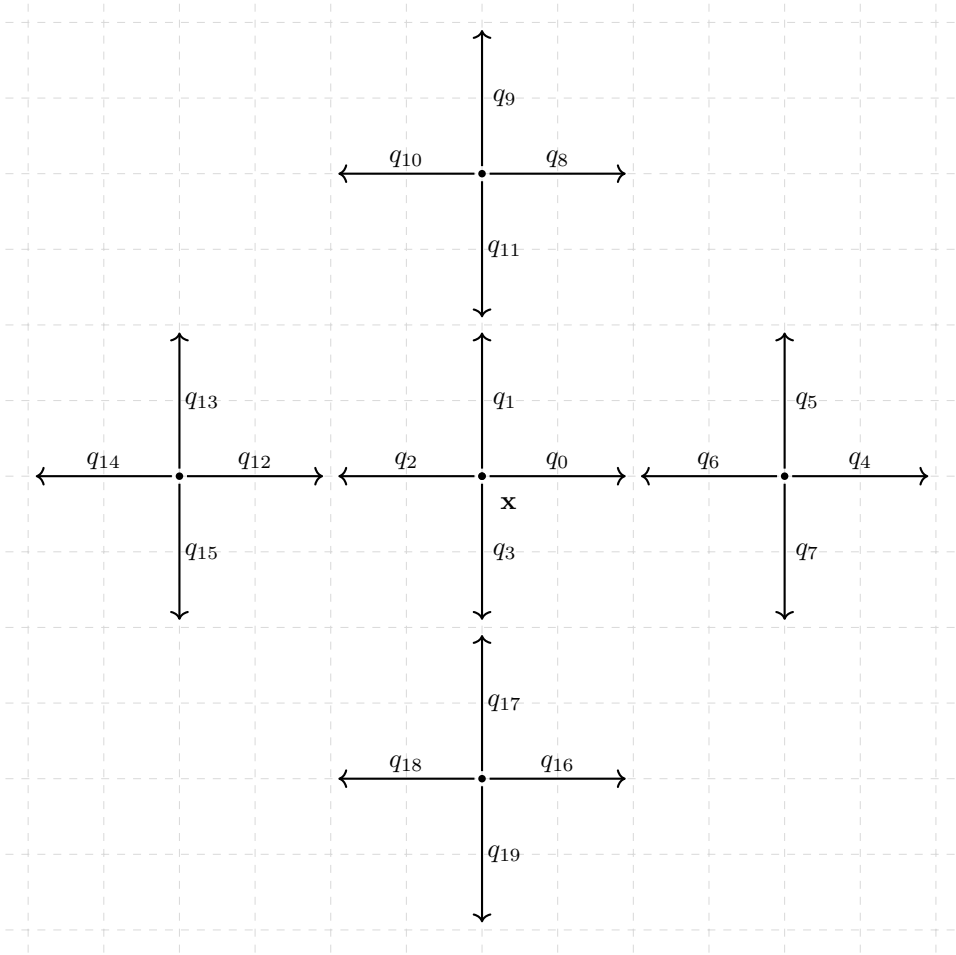


Figure 5.2: Illustration of the space-time encoding for D2Q4 for a single time step.

5.4. COLLISION FOR SPACE-TIME ENCODING

To encode the lattice gas collision step we first identify the equivalence class for the D2Q4 lattice. We note that at each grid point x as represented in Figure 5.1 the states $|q_0 q_1 q_2 q_3\rangle = |1010\rangle$ and $|q_0 q_1 q_2 q_3\rangle = |0101\rangle$ belong to the same equivalence class (cf. Figure 4.1), as they have the same total mass and momentum.[2] We implement the colli-

[2] The other equivalence classes are $|q_0 q_1 q_2 q_3\rangle = |1000\rangle$ and $|q_0 q_1 q_2 q_3\rangle = |1100\rangle$ and all cyclic shifts of these patterns, and $|q_0 q_1 q_2 q_3\rangle = |1111\rangle$. However, they all have just a single representative so that we define the collision operator based on the ambiguous case.

sion step by defining a unitary operator U_{col} which performs the following mappings

$$U_{\text{col}}|1010\rangle = |0101\rangle, \tag{5.3}$$
$$U_{\text{col}}|0101\rangle = |1010\rangle, \tag{5.4}$$

while acting as the identity operation on any other basis state.[3] With the so-defined $U_{\text{col}} \in \mathbb{C}^{2^4 \otimes 2^4}$, we can write the total collision operation for an encoding of the velocities states v consisting of $n_v = 4k$ qubits as k-fold Kronecker products of U_{col} operations, i.e. $U_{\text{col}}^{\text{tot}} = U_{\text{col}} \otimes \cdots \otimes U_{\text{col}}$. Since each U_{col} requires a few CNOT and a single triple controlled NOT gate, see Figure 5.3, the total collision operator can be efficiently implemented even on near-term devices.[4]

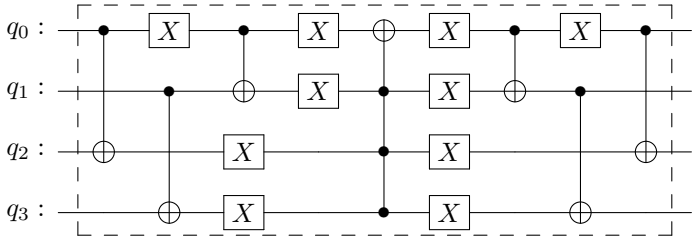


Figure 5.3: An example of an implementation of the collision operation U_{col} for the D2Q4 example.

In practice the total collision operator $U_{\text{col}}^{\text{tot}}$ differs per time step, since its local counterpart U_{col} only needs to be applied to velocity states 'in the vicinity of x'. In the first out of the N_t time steps it is important for all qubits representing velocity states 'in the vicinity of x' to be updated correctly. In the very last time step, however, it is only important for the qubits q_0, q_1, q_2 and q_3 to end up in the correct state. The more time steps t have been taken, the less time steps $N_t - t$ are still to be taken and so the 4-qubit local collision operator U_{col} only needs to be applied to the remaining qubits relevant for encoding the 'directly connected' velocity states as given in Equation (5.2).

[3] This operation is unitary, as can be verified by writing it as

$$U_{\text{col}} = \Pi_{0000,1010}\Pi_{0000,0101}M\Pi_{0000,0101}\Pi_{0000,1010}. \tag{5.5}$$

Here the matrices $\Pi_{i,j}$ represent permutation matrices between the basis states i and j which are trivially unitary and the matrix M is given by

$$M = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.6}$$

Since the permutation matrices are unitary, we only need to show the unitarity of M to show the unitarity of U_{col}. The unitarity of M follows directly from writing out $M^\dagger M$ and finding I.

[4] We can implement the described collision operator by first applying three CNOT operations to the system turning the states into $|1010\rangle \mapsto |1110\rangle$ and $|0101\rangle \mapsto |1111\rangle$. Subsequently a triple controlled NOT is applied to the right-most qubit (controlled on the three left-most qubits). Finally the initial three CNOT operations are applied in reverse order to reset all velocity states correctly.

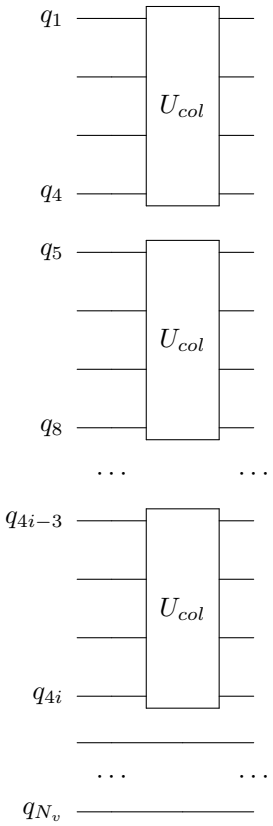


Figure 5.4: The collision operation applied in the i-th time step for the D2Q4 example.

With this logic we can define a collision operator per time step t as

$$U^{\text{tot}}_{\text{col},t} = \underbrace{U_{\text{col}} \otimes \cdots \otimes U_{\text{col}}}_{c \text{ collision operations}} \otimes \underbrace{I \otimes \cdots \otimes I}_{\text{identity operations}} , \tag{5.7}$$

where $c = 2(N_t - t)^2 + 2(N_t - t) + 1$ and the identity operations are added to avoid dimensionality issues. In practice no operation will be applied on the qubits encoding velocity states not 'in the vicinity of x' within $N_t - t$ time steps, Figure 5.4 shows what this looks like as a quantum circuit.

5.4.1. PARTICLE DENSITY INSPIRED COLLISION OPERATION

As stated above, we can choose to implement a different gate to represent collision, which brings the method presented closer to a relative particle density interpretation. We can do this by choosing a collision operator U'_{col} where

$$U'_{\text{col}}|1010\rangle = \alpha|1010\rangle + \beta|0101\rangle , \tag{5.8}$$
$$U'_{\text{col}}|0101\rangle = -\beta|1010\rangle + \alpha|0101\rangle , \tag{5.9}$$

with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$, while again acting as the identity operation on any other basis state. This operator can trivially seen to be unitary with the same arguments as the operator U_{col} with the added argument that

$$\begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix}, \tag{5.10}$$

is unitary.

As can be seen from Equations (5.8) and (5.9) using U'_{col} with $0 < |\alpha|^2, |\beta|^2 < 1$ allows to spread out the distribution of what we originally interpreted as one particle over space in two different possible positions. This allows for a new interpretation of the method where instead of interpreting each position as either occupied or not by a single particle, we can interpret the amplitudes of the occupied position as the relative particle density. This brings our proposed method closer to a lattice Boltzmann method, with a different collision operator. In the method presented in [1] they also present a collision operation that is a essentially a smoothed out version of the original LGA collision operations and they also interpret is as a method closer to lattice Boltzmann. More research needs to be done to determine its exact effects over multiple timesteps and how it compares to BGK and other standard collision operators.

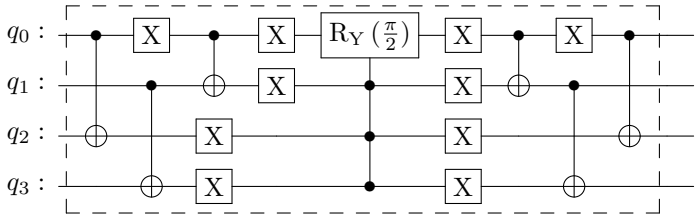


Figure 5.5: An example of an implementation of the collision operation U'_{col} for the D2Q4 example with $|\alpha| = |\beta| = \frac{1}{\sqrt{2}}$.

5.5. STREAMING FOR SPACE-TIME ENCODING

Our space-time encoding enables different manners of implementing the streaming step. It can easily be seen that the way the streaming method should be implemented differs per time step t depending on which positions will be 'in the vicinity of x' in the next time step as well. At the first time step it is important for (almost) all qubits to be streamed to a very specific position, whereas in the last time step it is only important for the qubits q_0, q_1, q_2 and q_3 to end up in the correct state. For the example shown below we are only considering a total of one step to be taken (i.e. $N_t = 1$) and so we only need to consider the speeds that will stream to location x in one time step. In this case that means that streaming consists of performing a swap operation between the following qubit pairs q_0 and q_{12}, q_1 and q_{17}, q_2 and q_6 as well as q_3 and q_{11}, see Figure 5.6.

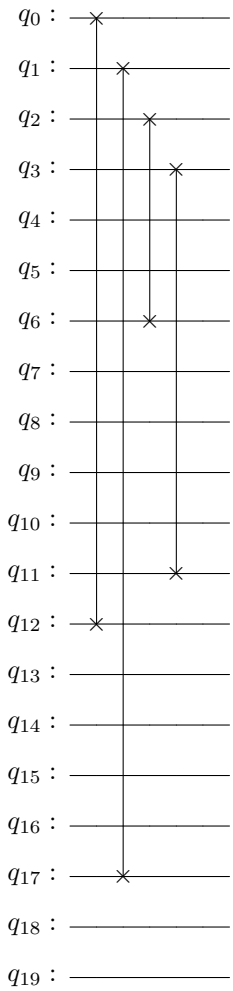


Figure 5.6: An example of an implementation of streaming in the D2Q4 case with $t = 1$.

Also in general (i.e. $N_t > 1$), the streaming step can be implemented by a combination of swap gates. Following the same in-the-vicinity-of-x argument as was used for the collision step, a total of

$$n_{\text{swap}}(t) = 4 + \sum_{i=1}^{N_t - t} 16i = 8(N_t - t)^2 + 8(N_t - t) + 4 \tag{5.11}$$

swap gates are required to update as many velocity-encoding qubits in time step t, whereby these swap operations can be performed largely in parallel.[5] The depth of the streaming

[5] In each time step the swap operations in the 4 (or generally speaking m) different directions can be performed

circuit at time t will amount to

$$d_{\text{str}}(t) = \log_2(N_t - t) \tag{5.12}$$

swap operations at time t. When combining the state preparation with the streaming and collision operations as described the total algorithm can be expressed as in Figure 5.7.

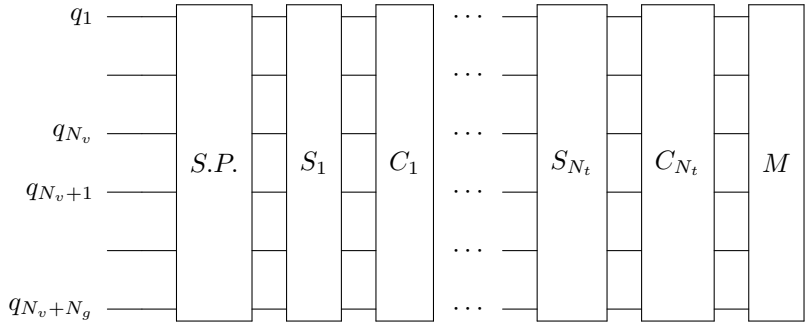


Figure 5.7: The full space-time data encoding quantum Boltzmann algorithm where S.P. stands for state preparation, S_i and C_i are the i-th streaming and collision operations respectively and M stands for measurement.

5.6. COMPLEXITY ANALYSIS

In this section we give an overview of the total number of qubits required and the circuit depth for this method.

5.6.1. NUMBER OF QUBITS REQUIRED

As described above, space-time encoding requires at worst $4N_g$ qubits. When considering a generalization of the space-time method for a DdQq setting, space time encoding requires at worst qN_g qubits. When the number of time steps to be taken, however, is smaller than the number of grid points fewer qubits are required to encode the velocity. In this case the number of qubits required to encode the velocity is of the order $\mathcal{O}\left(N_t^d\right)$.

5.6.2. CIRCUIT DEPTH

The circuit depth per time step for this approach is relatively small. The total depth of the collision operator for the D2Q4 example as is worked out is 20 CNOT gates[6]. Note that a different instantiation of DdQq would have a very different collision circuit with

in parallel. Furthermore the swap operations for the velocities in the same direction but not in the same 'line of streaming' can all be performed in parallel. Therefore we only need to take into account the velocities in the same line of streaming and the depth of the circuit is determined by the longest 'line of streaming', which is equal to $N_t - t$. In each layer of the swap operations at least half of the $N_t - t$ velocities can be swapped to the correct position. Therefore a total of $\log_2(N_t - t)$ swap operations needs to be performed in the t-th time step.

[6] Here we used the qiskit 1.2.0 built-in decomposition, which decomposes a triple controlled NOT gates into 14 CNOT gates.

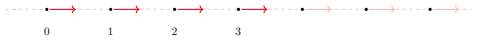
potentially a very different circuit depth.

The depth of the streaming circuit at time t can be derived using the following approach. We first note that we can do all the swaps in parallel that are not in the same 'line of streaming' in the stencil, the longest such line at time t will be $N_t - t$. In order to determine the depth of the circuit we consider the longest such line. We subsequently swap adjacent pairs in the longest such line as shown in Figure 5.8a, which puts $\lfloor \frac{N_t - t}{2} \rfloor$ particles virtually in the correct position. Figure 5.8b shows that after the first swap operations the grid points indexed by odd numbers have the correct value, since they were swapped with their counterpart on the left and we are considering a rightwards stream. The value at a grid point being correct is shown by the arrows getting faded out, as these grid points will no longer participate in the streaming process. For visual simplicity we then renumber the remaining grid points in Figure 5.8c and reorder them accordingly in Figure 5.8d. We subsequently repeat this process as shown in Figures 5.8e and 5.8f. Finally the remaining two grid points which do not yet encode the correct value are swapped as shown in figure 5.8g. In each step of the streaming circuit, as shown in Figure 5.8, approximately half of the grid points that do not yet have the right value, will get the right value assigned to them. Therefore the depth of the streaming circuit at time t consists of $\lceil \log_2 (N_t - t) \rceil + 1$ swap gates. As one swap gate can be decomposed into three CNOT gates the depth of this quantum primitive is $d_t = 3\lceil \log_2 (N_t - t) \rceil + 3$ CNOT gates.

(a) Starting position for the efficient streaming implementation using layers of swap gates for $N_t - t = 7$

(b) Position after the first layer of swap operations for the efficient streaming implementation using layers of swap gates for $N_t - t = 7$

(c) Position after the first layer of swap operations for the efficient streaming implementation after renumbering the remaining grid points.



(d) Position after the first layer of swap operations for the efficient streaming implementation after reordering the remaining grid points.

(e) Position after the second layer of swap operations for the efficient streaming implementation using layers of swap gates for $N_t - t = 7$.

(f) Position after the second layer of swap operations for the efficient streaming implementation after renumbering and reordering the remaining grid points.

(g) Position after the third layer of swap operations for the efficient streaming implementation using layers of swap gates for $N_t - t = 7$.

Figure 5.8: Example of efficient streaming implementation using layers of swap gates for $N_t - t = 7$.

5.7. CONCLUSION AND OUTLOOK

In this chapter we presented the space-time encoding. The space-time encoding is specifically designed to circumvent the problems of non-unitarity described in Chapter 4. Note that for this method we have not presented explicit measurement methods or methods to deal with specular reflection or bounce back boundary conditions. This work will

be presented in future publications.

BIBLIOGRAPHY

[1] Sriharsha Kocherla et al. “Fully quantum algorithm for mesoscale fluid simulations with application to partial differential equations”. In: *AVS Quantum Sci* (2024). DOI: https://doi.org/10.1116/5.0217675.

[2] Merel A. Schalkers and Matthias Möller. “On the importance of data encoding for quantum Boltzmann methods”. In: *Quantum Information Processing* (2024). DOI: https://doi.org/10.1007/s11128-023-04216-6.

6

CONCLUSION AND DISCUSSION

We conclude this thesis by summarizing the main results and identifying the three open questions in this field that need to be answered to determine the future of quantum fluid dynamics.

6.1. OVERVIEW OF RESULTS

In this section we give an overview of the results presented in this thesis. We do this by first revisiting our results for amplitude based encoding techniques. We then summarize the proofs of impossibility where we discuss that for different the standard amplitude based encodings and computational basis state encodings it is not possible to extend the technique to include a collision or streaming step respectively. We finish the summary by zooming into our space-time encoding, which is designed to circumvent the problems of computational basis state encodings and does allow for a streaming step to be implemented.

6.1.1. AMPLITUDE BASED ENCODING

In this thesis we have presented an amplitude based encoding scheme that can solve the transport equation or collisionless Boltzmann equation with a lower complexity in terms of gates used than the former state of the art; see Chapter 2. Furthermore we have presented methods that can correctly implement specular reflection and bounce back boundaries conditions; see Chapter 2 and Chapter 3. Our boundary condition implementation makes use of ancilla qubits to ensure the behavior is correct for all DdQq instances. Our method can be extended for more complex flow fields by extending the area of the object that is treated as an edge case as described in Chapter 2. Furthermore we presented the quantum momentum exchange method, which provides a quantum implementation of the classical momentum exchange method 3.

6.1.2. PROOFS OF IMPOSSIBILITY FOR AMPLITUDE BASED AND COMPUTATIONAL BASIS STATE ENCODING

As shown in Chapter 4, the commonly used amplitude based encoding does not allow for a unitary implementation of the collision operation, due to the orthogonality of the encoding of the different velocities. Furthermore we have shown that for the commonly used computational basis state encoding, streaming cannot be implemented due to its non-locality.

These proofs show the general impossibility of adding a collision or streaming block to these types of existing encoding methods due to their structure. As such the proof shows that adding a collision operation to any amplitude based encoding for which the different velocities are encoded in orthogonal basis states is impossible. Similarly we have proven that it is impossible to add a streaming operation to any computational basis state encoding for which streaming would have to cross the boundaries of locality. We have designed and presented one such example of an encoding in which the boundaries of locality are extended in such a way that multiple time steps using this algorithm is possible.

6.1.3. SPACE-TIME ENCODING

As presented in Chapter 5, the space-time encoding is designed with the proofs of impossibility for commonly known encoding methods in mind. Specifically we extended the computational basis state encoding in such a way that as long as we stay within the pre-determined number of time steps, we stay within the locally defined domain. In

doing this the total number of required qubits grows polynomially with the number of time steps taken, where the degree of the polynomial depends on the spatial dimension, where the number of qubits required can never grow beyond qN_g.

6.2. THREE OPEN QUESTIONS

Finally we want to emphasize three open questions in this field that will determine to what extent QCFD can be of use in the future.

Space-time encoding is possible because it extends computational basis state encoding in such a way that any place the particles can stream to during the total number of time steps taken, become part of the local encoding. Therefore it circumvents the issue of non-locality. The natural question that arises is then whether or not an altered version of amplitude based encoding, where the velocities are encoded in a non-orthogonal way, can lead to an interesting quantum algorithm.

Question 1: Is it possible to find an altered version of the amplitude encoding method allowing for unitary collision?

Since space-time encoding is currently the best known encoding method for implementing both streaming and collision unitarily without the need for a stop and go method, the natural question is whether or not a more efficient encoding can be designed.

Question 2: Is it possible to find a full quantum algorithm without reinitialization that uses fewer qubits than space-time encoding and remains polynomial in circuit depth?

The current reason stop-and-go methods are not very appealing for implementation lies in the costs of measurement and reinitialization. This raises the question of whether new QCFD algorithms can be designed which make use of measurement and reinitialization without exponential cost.

Question 3: Is it possible to design a stop-and-go method with efficient measurement and reinitialization?

Answering these questions will give more insight into the different ways quantum methods for the Boltzmann equation can be useful in society. Specifically the answers to the three questions above will determine if quantum computing can actually promise a speed-up with respect to classical computers for lattice Boltzmann methods.

ACKNOWLEDGEMENTS

No achievement is attained by one person alone. We are all dependent on our surroundings and the cards we have been dealt. I like to think I have had the luck in life to be born in great circumstances which, combined with hard work and dedication, have allowed me to be where I am today.

I am incredibly grateful for my friends and family and the love and joy they bring in my life. Furthermore I want to express my deep gratitude to my supervisors, collaborators and colleagues for the support and fun moments during my academic journey.

The past 28 years have been good, let's hope the next can be even better. Whoever you are reading this, I wish you all the best.

CURRICULUM VITÆ

Merel Annelise SCHALKERS

27-12-1996 Born in Voorhout, The Netherlands.

EDUCATION

2015–2018 BSc Liberal Arts and Sciences
Amsterdam University College

2019–2021 MSc Applied Mathematics
Delft University of Technology

2021-2025 PhD. Applied Mathematics
Delft University of Technology
Thesis: The Quantum Lattice Boltzmmann Method
Towards quantum methods for computational fluid dynamics
Promotors: Em. Prof. dr. ir. C. Vuik
Dr. rer. nat. M. Möller
Dr. D. de Laat

AWARDS

2022 Best paper, best presentation and best young scientist award
I4CS

List of Publications

Conference papers

1. M. Möller and **M.A. Schalkers**, |Lib⟩: a cross-platform programming framework for quantum-accelerated scientific computing, in *Computational science—ICCS 2020. Part VI*, 451–464, Lecture Notes in Comput. Sci., 12142, Springer, Cham,.

2. **M.A. Schalkers** and M. Möller, Learning Based Hardware-Centric Quantum Circuit Generation, in *Innovations for Community Services—I4CS 2022*, 308–322, Communications in Computer and Information Science **1585**.

Journal papers

1. **M.A. Schalkers** and M. Möller, Efficient and fail-safe quantum algorithm for the transport equation, J. Comput. Phys. **502** (2024), Paper No. 112816, 25 pp.

2. **M.A. Schalkers** et al., Explaining Grover's algorithm with a colony of ants: a pedagogical model for making quantum technology comprehensible, Phys. Educ. **59** (2024), no. 3, Paper No. 035003, 12 pp.

3. **M.A. Schalkers** and M. Möller, On the importance of data encoding in quantum Boltzmann methods, Quantum Inf. Process. **23** (2024), no. 1, Paper No. 20, 19 pp.

4. W. Nowak et al., Overdispersion in gate tomography: experiments and continuous, two-scale random walk model on the Bloch sphere, ACM Trans. Quantum Comput. **5** (2024), no. 4, Art. 24, 17 pp.

5. **M.A. Schalkers** and M. Möller, Momentum exchange method for quantum Boltzmann methods, Comput. & Fluids **285** (2024), Paper No. 106453, 8 pp.

6. C.A. Georgescu, **M.A. Schalkers** and M. Möller, QLBM - A quantum lattice Boltzmann software framework, Comput. Phys. Commun. **315** (2025), Paper No. 109699 , 22 pp.

Preprints

1. C.A. Georgescu, **M.A. Schalkers** and M. Möller, Fully Quantum Lattice Gas Automata Building Blocks for Computational Basis State Encodings, Arxiv preprint (2025).