Quantum Algorithm for the Collisionless Boltzmann Equation

Blaga N. Todorova and René Steijl

School of Engineering, University of Glasgow, G12 8QQ, Glasgow, United Kingdom

Abstract

A novel quantum algorithm implementing a discrete-velocity method for the collisionless Boltzmann equation is introduced. The algorithm is designed for application on a quantum computer with a number of quantum bits feasible in the near future (e.g. $40 - 50$). Following the quantum-circuit model of quantum computation, the present works shows the quantum-circuit implementations for the convection or transport part of the kinetic model, inspired by work on quantum algorithms for the Dirac equation. The present work represents the advection step as a quantum walk process, implemented as a series of multiple-input controlled-NOT gates. A detailed discussion on the background to this new method is presented, including how a rarefied-flow problem can be encoded as the quantum state of a qubit register in a quantum computer. A complexity analysis is presented showing potential benefits of the proposed algorithm. Based on the concept of quantum parallelism, the extension to multiple species is demonstrated to not increase the number of required gate operations. A key aspect of the developed algorithm is the implementation of boundary conditions. This work describes how the specular-reflection boundary conditions can be effectively imposed with a quantum circuit implementation. The developed method is then applied to the supersonic flow around a blunt body as well as the free-molecular flow escaping from a rectangular container. As validation, the flow along the stagnation streamline of the blunt-body flow is compared with exact solutions for a piston-driven flow, showing excellent agreement. Finally, directions for future work are discussed in this work.

Keywords: Quantum computing, rarefied flow, kinetic modelling

Accepted for publication, 17 February 2020

1. Introduction

In recent years the field of quantum computing (QC)[1] has grown into an active and diverse field of research and significant progress has been made with building quantum computers. For a small number of applications, quantum algorithms have been developed that would lead to a significant speed-up relative to classical methods when executed on a suitable quantum computer. Despite this research effort, progress in defining suitable applications for quantum computers has been relatively limited. Two decades after their invention, Shor's algorithm for factoring composite integers[2] and Grover's algorithm for quantum search[3] are still among the main applications. Applications to computational science and engineering problems beyond quantum chemistry have only recently begun to appear[4, 5, 6, 7]. Further applications have been developed which take advantage of the unique capabilities of quantum computing platforms, e.g. methods for the solution of linear systems of equations[8], numerical gradient estimation[9], the Poisson equation[10] and the wave equation[11].

The present work aims to investigate the potential of quantum computing and suitably designed algorithms for future computational fluid dynamics applications. In the absence of the required quantum hardware, large-scale parallel simulations on parallel classical computers are required in developing such algorithms. For this purpose, a recently developed quantum computer simulation capability introduced in the MΦC framework[12, 13, 7] is used.

The novel quantum algorithm introduced here implements a discrete-velocity method for the collisionless Boltzmann equation. The full Boltzmann equation is of particular interest for the study of rarefied flows. Particle interactions are accounted for in a complex, computationally demanding particle-collision term. In the collisionless Boltzmann equation this term is neglected, limiting the application to (nearly) free-molecular flows. The discrete-velocity method discretizes the state-space of the Boltzmann equation with a three-dimensional mesh for a three-dimensional flow, leading to a six-dimensional solution space (seven dimensions when time is included). This leads to very large computer memory requirements which often make this approach impractical for applications in 3D. The key benefit on the quantum algorithm is in the representation of the high-dimensional solution space as a quantum state, realized using a limited number of quantum bits (qubits) in a coherent state in the quantum computer considered. Further benefit of the proposed algorithm is its extension to application to multiple-species simulations. While in a classical implementation, doubling the number of species means doubling the required memory, the quantum algorithm handles this by the addition of a single quantum bit. Furthermore it will be shown that no additional quantum gates are required when doubling the number of species, in contrast to the extra computational work required in the classical implementation.

In the near future, the most likely scenario for the introduction of quantum computing hardware is through the quantum co-processor model, i.e. where one or more quantum processing units (QPUs) are loosely coupled to a classical computer with one or more CPUs. In current designs, the quantum processor requires storage at low temperatures in a cryostat leading to a distinct physical separation between the classical and quantum hardware. Coupling takes place by exchanging classical information. In case the quantum hardware involves multiple quantum processors, quantum-entanglement based coupling can be used between these QPUs. In the near term, the most likely hardware layout involves only a single QPU coupled to a classical computer with one or more CPUs. In application of this hybrid quantum/classical approach, the quantum processor acts like a co-processor with the quantum processor dealing with selected computationally demanding tasks. The quantum processor receives information from the CPU and this is used to initialize the quantum state in the quantum processor. During the quantum simulation, the quantum state is transformed by application of quantum gates to the quantum system state vector. Then measurement operations are used to extract classical information from this quantum state and this is subsequently passed to the CPU. Since in quantum mechanics a measurement leads to the (partial) collapse of the quantum state, in the hybrid classical/quantum approach typically multiple realizations of the quantum state will be needed to obtain classical information with acceptable levels of noise and uncertainty. Also, since initializing a particular quantum state in a quantum computer can be a significant challenge, this hybrid approach can only be expected to lead to significant computational speed-ups in case the quantum simulation is significantly faster for the selected problem than conventional solution methods.

As an example of this hybrid classical/quantum approach, the authors introduced a quantum computing application in which the vortex-in-cell method was used to solve the incompressible-flow Navier-Stokes equations in a regular domain. In this solution approach, the Poisson solvers dominating CPU time requirements are based on the quantum computing equivalent of the Fast Fourier Transform, i.e. the Quantum Fourier Transform[7]. In the time-integration used, information exchange between the QPUs and classical part of the hardware will need to take place each time-step, leading to a significant overhead in setting up the quantum states in the QPUs. Furthermore, each time information is drawn from the QPUs quantum errors and statistical noise were present. A key aspect of this study was therefore an investigation of the effect of quantum errors and statistical noise present in the classical information passed from the QPU to the CPU executing the rest of the algorithm. It was found that a noise threshold could be defined for which the vortex-in-cell method was still capable of performing worthwhile simulations for a number of vortex-interaction test cases.

1.1. Contributions of present work

The key contribution of the present work is a quantum algorithm for simulating flows at the kinetic level which does not require the repeated quantum state initialization steps and repeated measurements. In fact, the presented algorithm is designed to be executed fully within a quantum processor with measurement steps only applied at the end of the simulation. Depending on the level of detail required from the computed flow field, multiple realizations are still required. To the best of our knowledge, the novel algorithm presented here is among the very first algorithms related to fluid dynamics that can be performed fully on a quantum computer, i.e. with information transfer between hybrid and classical hardware taking place only at start (initialization) and end of simulation. It therefore does not involve a frequent exchange of information as sketched in the hybrid classical/quantum approach in the previous paragraph, which leads to significant computational advantages in terms of speed-up relative to classical approaches as well as reduction of noise introduced by sampling of measurement data.

A key benefit of the introduced algorithm relative to a classical equivalent is the exponential reduction in memory when expressed in terms of qubits, as will be shown in Section 6. A detailed complexity analysis in that section also shows that the introduced method has a time complexity $O(TdN_v log_2(D/h))$ in terms of required number of multi-qubit gate operation, for a single realization of the problem with N_v discrete velocities per direction in a d-dimensional domain with domain size D and cell-spacing h for a simulation time T. For a classical implementation we find a complexity $O(TdN_v^d \times (D/h)^d)$. Therefore, there is a potential exponential speed up for the quantum algorithm. It should be noted that the complexity of the quantum algorithm quoted here represents the complexity of performing the required operations on the state vector for the time evolution phase of the simulation. Clearly, initialization of the quantum state vector and extracting classical output from this state vector are key aspects to be considered as well. In case these operations cannot be performed efficiently, significant challenges exist in achieving a meaningful speed-up. Both aspects will be discussed in this work. However, it is clear that further research work is needed to address the questions related to efficiently initializing the algorithm as well as extracting classical information at the end of the simulation.

1.2. Related work

Early work in quantum computing relevant to the field of computational fluid dynamics mainly involves the work on quantum lattice-gas models, e.g. by Yepez and co-workers[14, 15]. This work typically involved type-II quantum computers, consisting of a large lattice of small quantum computers interconnected in nearest neighbour fashion by classical communication channels. It was shown that at the mesoscopic scale, a lattice Boltzmann equation results with a nonlocal collision term that depends on the entire system wave function. In more recent work[16], quantum lattice gas models of the Navier-Stokes fluid dynamics formulated for measurement-based quantum computers involving six qubits per node were considered. However, the small number of coherent qubits will ultimately limit the achievable speed-up relative to classical methods. In contrast to these quantum lattice-gas based approaches, the present study focusses on a quantum algorithm designed for near-future 'universal' quantum computers, i.e. without using a clustering of small quantum computers or quantum registers connected in a lattice.

The quantum algorithm proposed here employs an approach for the convection step that can be regarded as a quantum walk. Quantum walks were introduced as the quantum mechanical counterpart of classical random walks. Quantum walks have been proven to be a universal model for quantum computation and represent a powerful tool for building quantum algorithms[17, 18]. A recent summary of quantum walks and its applications was presented by Venegas-Andraca[19]. For the present work, the main link with quantum walks is the conditional move on a regular graph or lattice and the possible quantum circuit implementation. Douglas and Wang[20] presented a number of highly symmetric graphs on which efficient quantum circuits implementing quantum walks can be constructed. Based on these techniques, a quantum-walk based algorithm was introduced by Fillion-Gourdeau et al.[21] for the solution of the Dirac equation on a digital quantum computer. In their algorithm, an operator-splitting decomposition technique is employed that allows for a mapping of the Dirac operator to a quantum walk supplemented by unitary rotations steps in spinor space. The convection step in our algorithm was inspired by these works. A further direct influence was the work presented by Fillion-Gourdeau and Lorin[22] where quantum algorithms were derived for the Cauchy problem for symmetric first order linear hyperbolic systems, employing the reservoir technique.

For the key aspect of imposing non-periodic boundary conditions, the quantum-circuit implementation of specular-reflection boundary conditions as used in our quantum algorithm is the main novelty of this work, since to the best of our knowledge no (directly) related previous work exists. As detailed later, specular reflection is an elastic process in which the velocity component of a particle normal to a solid wall is reversed during collision with the wall, while the velocity component parallel to the wall is retained.

In terms of recent developments for improved and faster discrete-velocity methods for kinetic equations, and in particular for the transport or convection part of the equations, computational efficiency is typically based on a characteristics-based time-integration in a semi-Lagrangian approach[23, 24, 25, 26]. Among these recent developments is the Fast Kinetic Scheme for kinetic equations developed by Dimarco and Loubere[25, 26], that is based on a splitting technique between the transport and relaxation operators with the collision part solved on a grid. For the transport part, an exact solution is used that involves following the characteristics backward in time, without a need for reconstruction of the distribution function at each time step, as is typically used in previous semi-Lagrangian methods for Boltzmann/kinetic equations[23, 24]. Although, the reservoir-technique based method developed here resembles the fast semi-Lagrangian method of Dimarco and Loubere in terms of the characteristics-based time-integration and the exact propagation of piecewise constant discretized particle distribution functions, a key difference is the fact that in our method the main focus was on facilitating quantum computer implementations. This is achieved by creating a scheme in which discrete data on a regular mesh moves from one cell center exactly one mesh width to a nearest neighbour. In contrast the method of Dimarco and Loubere[25, 26] involves an integration continuously in space, i.e. no spatial mesh is involved in the transport step, which means that the the approach used here in deriving a quantum-computer implementation cannot be used for their method.

1.3. Organization of present work

The paper is organized as follows. A summary of quantum computing principles and related work is presented in Section 2. Section 3 presents a summary of key aspects of the collisionless Boltzmann equation. The Reservoir technique used for the time-integration of the collisionless Boltzmann equation is presented in Section 4. Section 5 defines the quantum circuits representing the implementation on a quantum computer. Section 6 presents the complexity analysis of the proposed quantum algorithm. Application of the developed method to steady and time-accurate free-molecular flows are presented in Section 7, including an analysis of the required velocity-space discretization and validation of the method using analytical solutions for free-molecular flow. Finally, Section 8 summarizes key findings and future research directions.

2. Quantum Computing Principles

Before describing the new quantum algorithm for the collisionless Boltzmann equation, a number of definitions commonly used in quantum computing literature are briefly reviewed. A detailed account can be found in a number of standard textbooks, e.g. Nielsen and Chuang[1]. The fundamental unit of quantum computation is the quantum bit or qubit. Whereas a classical bit is confined to existing in either the 0 or 1 state, a qubit can be in a state of superposition, i.e. it exists in both states simultaneously. Upon measuring the qubit, the quantum state collapses to either of these two states, and the qubit is no longer in a state of superposition. The state of a qubit is defined through a pair of complex numbers c_0 and c_1 such that the probability of finding the qubit after measuring in state 0 is $|c_0|^2$ and the probability of measuring state 1 is $|c_1|^2$. The amplitudes are bound by the requirement $|c_0|^2 + |c_1|^2 = 1$. A collection of n_q qubits in a coherent state is termed a *quantum register* of size n_q here. Its quantum state is defined by the wavefunction $|\psi\rangle$ created by superposition as $\left|c_{n_q-1}\right\rangle \otimes \left|c_{n_q-2}\right\rangle \ldots |c_1\rangle \otimes |c_0\rangle$, often written as $\left|c_{n_q-2}c_{n_q-1} \ldots c_1c_0\right\rangle$. Here $c_0, \ldots, c_{n_q-1}$ are complex numbers representing quantum wave number amplitudes associated with each of the n_q qubits. In the following, the qubits within a quantum register are ordered such that the left-most qubit represents the most-significant bit, while the least-significant bit is the right-most bit in the register.

2.1. Measurement of a quantum state

For a quantum system with n_q coherent qubits, measurement of the quantum system can lead to 2^{n_q} possible outcomes, with the likelihood of each outcome defined by amplitudes of the n^{n_q} complex numbers created by tensor product of the n_q complex amplitudes corresponding to quantum states of the n_q coherent qubits. Similar to the situation for an isolated qubit, measurement of the quantum state will change the quantum state and reduce or completely remove superposition. In this context a computational basis with 2^{n_q} states is defined, where a full measurement leads to a complete collapse of the quantum state into one of the possible 2^{n_q} with a likelihood corresponding to the 2^{n_q} complex amplitudes defining the quantum state. Therefore, to extract detailed information of the quantum state $|\psi\rangle$, a large number of realizations are needed to be created following by measurement. Statistical analysis will then provide the information on each of the 2^{n_q} possible outcomes. Other measurements with a partial collapse are also possible. For example, information about the state of one of the n_q qubits can also be obtained using a suitable measurement operation. After such a measurement, the quantum state $|\psi\rangle$ will then have a reduced level of superposition, i.e. with 2^{n_q-1} degrees of freedom.

2.2. Quantum circuit model

In the present work, the quantum circuit model of quantum computing is used. In this case, the unitary operation on a quantum state allowed by quantum mechanics are represented by a series of quantum (logic) gates acting on the quantum state. A quantum logic gate is an elementary quantum computing device which performs a fixed unitary operation on selected qubits in a fixed period of time. Written in matrix form, unitary means that the determinant of the transformation is unity. The application of a single-qubit gate on qubit iq in a n_q qubit register, can be written as the following matrix operation (U represents 2×2 unitary matrix) on the wavefunction $|\psi\rangle = \sum_{j=0}^{2^{n_q}-1} c_n |n\rangle$, with the complex amplitudes represented by $\vec{c}$

$$\begin{pmatrix} c_{n_i} \\ c_{n_i+2^{n_q-1-iq}} \end{pmatrix} \leftarrow U \begin{pmatrix} c_{n_i} \\ c_{n_i+2^{n_q-1-iq}} \end{pmatrix} = \begin{pmatrix} U_{11}c_{n_i} + U_{12}c_{n_i+2^{n_q-1-iq}} \\ U_{21}c_{n_i} + U_{22}c_{n_i+2^{n_q-1-iq}} \end{pmatrix} \quad (1)$$

where $n_i = (i/2^{iq})2^{iq+1} + (i \bmod 2^{iq})$ for every integer $i \in [0, 2^{n_q} - 1]$. Here, qubits are numbered $iq = 0$ for the least significant bit, to $iq = n_q - 1$ for most significant bit. Of particular interest for the implementation of the proposed quantum algorithm are controlled-NOT operations, i.e. where the state of a *target qubit* is swapped from $|0\rangle$ to $|1\rangle$ or vice versa depending on the value of one or more *control qubits.* For the one-qubit NOT operation (no control bits) and the two-qubit $CNOT$ (one control bit) the corresponding unitary matrices are shown in Equation (2),

$$U_{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \; ; \; U_{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2}$$

By the conditional negation of the state of the target qubit, the $CNOT$ gate is known to play an important role in changing the level of quantum entanglement between the qubits in the quantum register, as analyzed in detail for small quantum circuits with up to 6 qubits by Karafyllidis[27]. A three-qubit gate with two control qubits is commonly called Toffoli gate and involves a sparse 8×8 matrix. The present quantum algorithm uses a series of multiple-input controlled-NOT gates to change the quantum state vector. In a quantum network consisting of quantum logic gates the computational steps are synchronized in time. The output of some of the gates are connected to the input of others. In the following sections, quantum circuits will be presented. Here, the vertical direction shows the qubit register with the left qubit at the top and the right-most qubit at the bottom. In the horizontal direction, going from left to right, the quantum state is changed by a series of quantum gates. Here, the cross ("x') represents a negation under the conditions that the control qubits have the required status: full circles mean that the control qubit should be in state $|1\rangle$, while open circles represent the control state $|0\rangle$.

2.3. Storing a multi-dimensional solution space on a quantum computer

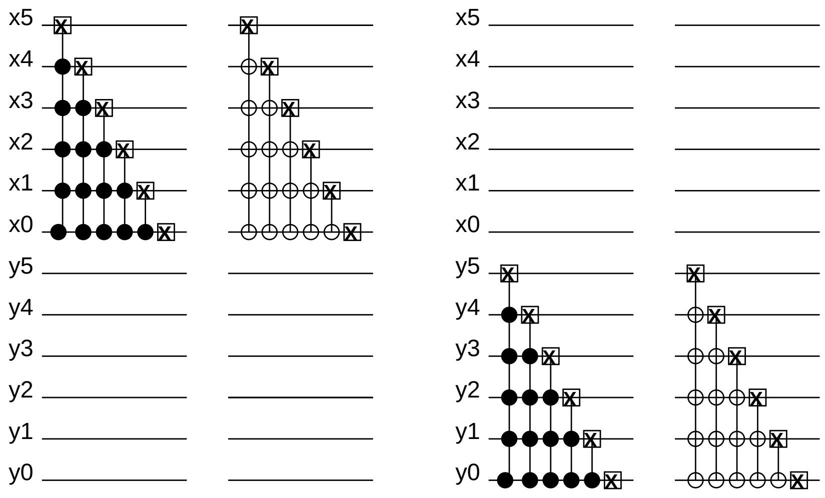
The quantum state vector $|\psi\rangle$ for a register with n_q coherent qubits is represented by a Hilbert space of dimension 2^{n_q}. Specifically, in a quantum computer simulation on classical computer we use 2^{n_q} complex numbers to fully define this state. For a function f discretized on a (regular) mesh with N mesh points, it follows that $log_2(N)$ qubits would suffice to create the required number of degree-of-freedom in the solution space. However, it is important to stress that the quantum state vector only represents the likelihood that upon measurement the quantum state collapses into a particular state[1]. In other words, with $n_q = log_2(N)$ we cannot extract the full information for all N degrees of freedom using a single realization of this quantum state. However, for as long as this classical information is not needed the quantum state has the required number of degree-of-freedom. For example, a function on a one-dimensional mesh with 32 mesh points can be stored in a 5-qubit register $|q_4q_3q_2q_1q_0\rangle$. The extension to two spatial dimensions is straightforward, i.e. the function on a two-dimensional mesh with 32×32 mesh points can be stored in a 10-qubit register $|q_9q_8q_7q_6q_5q_4q_3q_2q_1q_0\rangle$. For convenience, a renaming of the qubits will be introduced to better reflect the role of of the different qubits, i.e. the 10 qubits are indexed as, $|q_{x,4}q_{x,3}q_{x,2}q_{x,1}q_{x,0}|q_{y,4}q_{y,3}q_{y,2}q_{y,1}q_{y,0}\rangle$, distinguishing the qubits associated with the x- and $y-$direction indices, respectively.

Similarly, we can use the concept of adding further qubits to create space for vector data to be stored on the grid. For example, if we need to store discretized functions f and g on a two-dimensional mesh with 32×32 mesh points, we can use a 11-qubit register, with qubits indexed as $|q_{x,4}q_{x,3}q_{x,2}q_{x,1}q_{x,0}|q_{y,4}q_{y,3}q_{y,2}q_{y,1}q_{y,0}|q_g\rangle$, with q_g representing the qubit added to create space for function 'g' along with 'f'. Each additional qubits added after q_g would further double the available space. Clearly, other orderings are possible as well. In fact, for the simulation of the quantum circuits in the C++ simulator developed for the present work, the qubit ordering is especially important in terms of the strides in computer memory created by different qubit operations that are simulated.

2.4. Implementing a 'streaming' operation on a quantum computer

Going back to the one-dimensional function f discretized on N mesh points, we can identify N discrete values f_i, with $i \in [0, N-1]$. For the 32 grid points the indices i can also be represented in binary representation as $00000, 00001, \ldots, 11111$, providing a direct link to the states of the 5 qubits. Specifically, if all 5 qubits are in state $|0\rangle$, index $i = 0$ and all 5 qubits in state $|1\rangle$ corresponds to 31.

For the current quantum algorithm, two key operations are needed: a 'streaming operation' to the left and one to the right. For the streaming operation to the left we require that the value on mesh point i, i.e. f_i, is moved left to mesh point $i-1$. Similarly, a right streaming operation moves the value from point i to the right neighbor, i.e. $i+1$. It turns out that both operations can be effectively implemented in the quantum circuit model of quantum computing using a series of CNOT gates with multiple-control qubits[21, 22]. For an example 64×64 mesh, quantum-circuit implementation



(a) x-direction (b) y-direction

Figure 1: Quantum circuit representation of streaming operation in x-direction (a) and y-direction (b). Circuit for 64×64mesh. Circuits with filled circles represent 'right' streaming (test for $|1\rangle$ state of control qubits). Open circles represent tests for $|0\rangle$ state of control qubits used in 'left' streaming.

of the 'left' and 'right' streaming operations for both x- and $y-$directions are shown in Figure 1. It can be seen that for the 'right' streaming (left-hand circuit in each figure), the implementation involves a series of multiple-control NOT gates with each of the 6 qubits for each direction acting as target qubit during one gate operation. Negation of the target qubit (indicated with 'X' in the circuits) takes place in case all control qubits are in state $|1\rangle$ (indicated as filled circles in the figures). For the 'left' streaming operation, all control qubits in state $|0\rangle$ leads to negation of the target qubit. It is important to stress that this implementation assumed periodic boundaries conditions are imposed in both directions for the considered two-dimensional domain.

As a further illustration, the unitary transformations acting on the quantum state $|\psi\rangle$ are now detailed for a simple example with 3 qubits representing a one-dimensional domain with 8 points indexed $0, \ldots 7$ from left to right. Then, the right streaming operation involves three steps: (1) a Toffoli gate with qubit 2 as target (i.e. the top qubit in a quantum circuit diagram), and qubits 1, 0 and control qubits, (2) a CNOT gate with qubit 1 the target qubit and qubit 0 the control qubit, (iii) a NOT gate (negation operation) on qubit 0. The action taken on the quantum state can be written as three matrix-vector multiplications, with $U^{(k}, k = 1, 2, 3$ representing the unitary transformations as,

$$U^{(1)}\,|\psi\rangle \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} C_{000} \\ C_{001} \\ C_{010} \\ C_{011} \\ C_{100} \\ C_{101} \\ C_{110} \\ C_{111} \end{pmatrix} = \begin{pmatrix} C_{000} \\ C_{001} \\ C_{010} \\ C_{111} \\ C_{100} \\ C_{101} \\ C_{110} \\ C_{011} \end{pmatrix} \tag{3}$$

representing the application of the Toffoli gate. The CNOT gate changes the state as,

$$U^{(2)}\,|\psi\rangle \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} C_{000} \\ C_{001} \\ C_{010} \\ C_{111} \\ C_{100} \\ C_{101} \\ C_{110} \\ C_{011} \end{pmatrix} = \begin{pmatrix} C_{000} \\ C_{111} \\ C_{010} \\ C_{001} \\ C_{100} \\ C_{011} \\ C_{110} \\ C_{101} \end{pmatrix} \tag{4}$$

and for the final application of the NOT gate on the least-significant bit,

$$U^{(3)}\,|\psi\rangle \rightarrow \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} C_{000} \\ C_{111} \\ C_{010} \\ C_{001} \\ C_{100} \\ C_{011} \\ C_{110} \\ C_{101} \end{pmatrix} = \begin{pmatrix} C_{111} \\ C_{000} \\ C_{001} \\ C_{010} \\ C_{011} \\ C_{100} \\ C_{101} \\ C_{110} \end{pmatrix} \tag{5}$$

This example shows that all states have 'streamed' one position towards the right.

In the quantum simulator used in this work, the unitary transformation matrices are not explicitly formed. Since the effect of the unitary transformations can be represented effectively using a series of loops, as sketched in

Equation (1) for a single-qubit gate operation, this storage is not required. The 2^{2n_q} storage requirement for the matrices would limit simulations to very small numbers of qubits.

2.5. Key challenges for current quantum algorithm development

The previous two subsections illustrated two important aspects of the quantum algorithm introduced in this work. The storage of data representing a multi-dimensional solution space, as illustrated in subsection 2.3 will form the foundation for the data structure of the discrete-velocity method developed here. The 'streaming' operations illustrated in section 2.4 in turn will form the basis for the convection step in the discretization of the convection step in the collisionless Boltzmann equation.

For the development of the quantum algorithm implementing the discrete-velocity method for the collisionless Boltzmann equation, a number of key aspects are still to be addressed;

1. In contrast the lattice-gas models and the widely-used Lattice Boltzmann methods, the convection step in a discrete-velocity method for kinetic equations does not involve a streaming step in which during a time step the state in one lattice node moves towards a neighboring node. Instead, depending on the considered discrete velocity as well as the chosen time step, the 'move' involves only a (small) fraction of a lattice spacing. To prevent the need for moves of less than a lattice spacing, the current algorithm will use the *reservoir technique*, as defined in following sections. Using this approach, and an important further simplification, the discretized convection step in the discrete-velocity method will be transformed into a series of 'streaming' operations acting on selected discrete-velocity data at different time steps;
2. The application of non-periodic boundary conditions: the quantum algorithm developed here targets free-molecular flow simulations in and around objects with solid walls, e.g. flow around bluff bodies as well as flow evacuating from reservoirs. The simplest relevant boundary condition we have implemented so far in the quantum algorithm applies specular-reflection boundary conditions (defined in more detail in next section);
3. Initialization of the quantum state vector. The preparation of a completely arbitrary quantum state in an N-dimensional Hilbert space has complexity of order N, i.e., exponential in the number of qubits

($N = 2^{nq}$). This complexity would seriously comprise the proposed quantum algorithm ([28, 29]). Therefore, efficient methods of initialization, i.e. without exponential scaling in number of qubits, are required. For quantum algorithms developed for simulations of quantum systems and quantum chemistry, a substantial amount of research work has resulted in a number of efficient methods for quantum-state preparation with polynomial complexity in number of qubits (e.g. Georgescu et al.[29] and references therein). For other quantum algorithm applications, efficient quantum-state initialization methods have been investigated, e.g. Grover and Rudolph[30], Soklakov and Schack[31], resulting in efficient methods for specific cases (for example log-concave probability distribution functions). For the proposed algorithm, the initial state needs to initialized such that it represents a Maxwellian distribution in velocity-space. This same state is defined for each lattice point in space. Since the Maxwellian distribution is log-concave with respect the molecular velocities, based on the work by Grover and Rudolph[30] it can be expected that initialization methods polynomial in the number of qubits can be devised. However, this aspect needs further investigation in future work.

4. Extraction of the solution data: in the present work, the full solution of a number of example free-molecular flow simulations will be presented. Since the quantum algorithm was simulated in our quantum computer simulator, this information is readily available. For cases in which the quantum algorithm would actually be executed on one or more QPUs, an important limitation occurs. As a result of the quantum measurements required to extract 'classical' information from the quantum state and the resulting (partial) collapse of the coherent state explained previously, only a limited amount of data can be extracted from a single realization. It is therefore inevitable that multiple realizations of simulations using the proposed algorithm are necessary to sample the amplitudes in the quantum state vector. In general, a technique called quantum state tomography (QST) can be applied to learn the full state. However, this leads to a scaling of the required number of samples with the size of the state space. An alternative approach is therefore needed that avoids this exponential scaling in the number of qubits. The intended use of the developed algorithm is therefore different from 'conventional' CFD methods returning information on the full multi-dimensional flow field. More specifically, the new algorithm

would have its main use in efficiently providing information on particle number densities and concentrations in multi-species flows while avoiding the need for extraction of the full flow field. One approach to this, inspired by quantum algorithms for quantum simulation and quantum chemistry, would be the estimation of certain physical quantities such as correlation functions or spectra of operators, as this is more efficient than taking the long route through QST. A detailed discussion is given by Ortiz et al. [32] and Somma et al.[33]. A key challenge for CFD applications of quantum algorithms is to develop a similar approach involving efficient use of correlation functions and spectra of operators. To the best of the authors' knowledge no published work exists related to this important research question. Furthermore, if we just want to obtain a single amplitude, the technique of amplitude estimation can be employed[34]. The technique can produce an estimate $\tilde{p}$ of the probability p of measuring the state to be within a specified subspace, with error bounded by $\delta = |\tilde{p} - p|$. In applying amplitude estimation, each iteration involves running the original algorithm forwards and backwards once. The number of realizations can be shown to scale as $1/\delta$. How amplitude estimation can be employed for the current quantum algorithm and its application to kinetic modelling needs further investigation in future work.

The solutions developed to address the first two aspects in the newly developed quantum algorithm are the key contributions of the present work. The last two aspects are essential in the practical application of quantum algorithms, and will be discussed further in Section 6. However, it is clear that both aspects need further research in future work.

3. Collisionless Boltzmann equation

The Boltzmann equation defines the single-particle distribution in a three-dimensional phase (velocity-space) for each point in three-dimensional space and therefore involves a seven-dimensional solution space (including time) for a gas consisting of a single monatomic species[35, 36]. In the present work, we consider highly rarefied gas flows (large Knudsen numbers) and make the assumption that these flows can be modeled as free-molecular gas flows, i.e. neglecting the collisions between the gas molecules. The collisions between gas molecules and domain boundaries is included, since this represents an essential feature of the considered flows. When neglecting inter-particle collisions and further body forces acting on the gas, the Boltzmann equation reduces to the collisionless Boltzmann equation, written here for a single-species flow as,

$$\frac{\partial F(\vec{x},\vec{c};t)}{\partial t} + \vec{c}\cdot\frac{\partial F(\vec{x},\vec{c};t)}{\partial \vec{x}} = 0 \tag{6}$$

$$F_{initial} = \frac{\rho}{(2\pi RT)^{3/2}}\exp\Big[-\frac{(\vec{c}-\vec{u}_0)^2}{2RT}\Big] \tag{7}$$

where $F(\vec{x},\vec{c};t)$ is the single-particle distribution function, and $\vec{x}=(x,y,z)^T$ and $\vec{c}=(c_x,c_y,c_z)^T$ represent three-dimensional space and three-dimensional phase (velocity) space, respectively. $F_{initial}$ defines the Maxwell-Boltzmann equilibrium distribution, for a local gas mass density ρ, temperature T and mean gas velocity $\vec{u}_0$, used in the present work as initial conditions of the simulations. The particle number density n and gas mass density ρ are related as $\rho = nm$, for molecular mass m. For the Maxwell-Boltzmann equilibrium distribution it follows that the most probable (thermal) speed of a particle depends on temperature $V_{mp} = \sqrt{2RT}$ with R the specific gas constant for the gas considered. For a molecular mass m this gas constant $R = k_b/m$, with k_b the Boltzmann constant. In rarefied gas dynamics, the so-called speed ratio defines the ratio of the molecular speed V and the most probable speed, $s = V/\sqrt{2RT} = \sqrt{\gamma/2}M_\infty$ with $\gamma = 5/3$ for a monatomic gas and M_∞ the free-stream Mach number. For an external flow around a solid object as considered in this work, the flow at free-stream conditions will be at the Maxwell-Boltzmann equilibrium distribution for a given free-stream density, temperature and mean velocity. For the flow considered here, periodic boundary conditions are imposed on the edges of the computational domain. For the solid walls considered, specular-reflection boundary conditions are applied, which can be described as follows. For a molecule traveling with a pre-collision velocity $\vec{c} = (u,v,w)^T$, the reflective collision with the wall creates a post-collision velocity $\vec{c}' = \vec{c} - 2\vec{n}(\vec{n}\cdot\vec{c})$, where $\vec{n}$ represents the unit wall normal vector. For the 2D test cases considered here, stationary solid walls will be used aligned with either the $x-$ or $y-$direction, so that $v-$ or $u-$ velocity component, respectively, will be reversed. In the quantum algorithm presented here, this effect is achieved by reversing this velocity for the first lattice point within the solid object next to the imposed wall, as detailed further in Section 5.3.

3.1. Dimensional reduction of the collisionless Boltzmann equation

For quasi-one dimensional or quasi-two dimensional flows, a dimensional reduction approach is commonly used to reduce significantly the required overhead. Starting from the full three-dimensional collisionless Boltzmann equation with single-particle distribution function $F(\vec{x}, \vec{c}; t)$, the dimensional reduction to two-dimensional problems replaces F with two reduced distribution functions as follows:

$$f(x, y, c_x, c_y; t) = \int_{-\infty}^{\infty} F(\vec{x}, \vec{c}; t) dc_z \ ; \ g(x, y, c_x, c_y; t) = \int_{-\infty}^{\infty} c_z^2 F(\vec{x}, \vec{c}; t) dc_z$$

$$\frac{\partial f}{\partial t} + \vec{c}\frac{df}{d\vec{x}} = 0 \ ; \quad \frac{\partial g}{\partial t} + \vec{c}\frac{dg}{d\vec{x}} = 0 \tag{8}$$

the corresponding gas density, mean velocities and temperature can be obtained from f and g using the following moments,

$$\rho = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f dc_x dc_y \ ; \ \rho \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \begin{pmatrix} c_x \\ c_y \end{pmatrix} f dc_x dc_y$$

$$\frac{3}{2}\rho T + \frac{u_0^2 + v_0^2}{2} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left[\frac{c_x^2 + c_y^2}{2} f + g\right] dc_x dc_y$$

For a quasi-one dimensional flow, a similar dimensional reduction can be introduced. As in the two-dimensional reduction, the original single Boltzmann equation is replaced with two lower-dimensional equations governing the two reduced distribution functions.

3.2. Extension to multiple-species mixtures

For a gas mixture with n_{sp} species, the collisionless Boltzmann model becomes a system of n_{sp} equations, which due to the absence of inter-particle collisions are uncoupled,

$$\frac{\partial F^{is}(\vec{x}, \vec{c}; t)}{\partial t} + \vec{c} \cdot \frac{\partial F^{is}(\vec{x}, \vec{c}; t)}{\partial \vec{x}} = 0 \tag{9}$$

$$F^{is}_{initial} = \frac{n_{is} m_{is}}{(2\pi(k_m/m_{is})T)^{3/2}} \exp\left[-\frac{(\vec{c} - \vec{u}_0)^2}{2(k_m/m_{is})T}\right] \tag{10}$$

for $is = 0, \ldots, n_{sp} - 1$. Here, $F^{is}(\vec{x}, \vec{c}; t)$ is the single-particle distribution function for species is. The number density of each species is is n_{is} and m_{is} is the molecular mass for species is. $F^{is}_{initial}$ defines the Maxwell-Boltzmann equilibrium distribution for species is, for a local number density n_{is}, temperature T and mean gas velocity $\vec{u}_0$. In the equilibrium distribution function, $\vec{u}_0$ is the mixture-mean velocity, i.e. the gas velocity obtained from the mass-averaged mean of each species mean velocity. In the present work, the Maxwell-Boltzmann equilibrium distributions are used as initial conditions in the simulations. As for the single-species Boltzmann equation, a dimensional reduction can be applied to each of the n_{sp} distribution functions, leading to a total of $2n_{sp}$ distribution functions in the governing equations.

3.3. Discrete-velocity method

The present method implements a discrete-velocity method with a Cartesian, uniform mesh in state space. The trapezoidal method is used to evaluate the numerical moments in state space when extracting the corresponding continuum quantities of the flow. Since we are focusing on the collisionless Boltzmann equation, a number of key differences with application of the discrete-velocity methods to model kinetic problems with collisions will occur. Firstly, the removal of collisions between the particle changes the characteristics of the flow significantly relative to more traditional gas dynamics, e.g. because of the initial conditions and boundary conditions it is for example quite likely that discontinuities arise in the particle distribution functions which describe within phase (velocity) space the probability that a particle has a particular velocity[37]. The macroscopic flow quantities that follow from taking moments of the distribution function over the full velocity space are then also likely to have discontinuities in space. Secondly, in the time-integration method, the evaluation of the continuum quantities of the flow is not needed at each time step since there is no need to construct a local equilibrium function typically required in a BGK-type relaxation model. For both reasons, in the present work it was decided to use a simple trapezoidal model on a uniform mesh instead of more advance collocation methods in state space. Also, the following assumptions will be made:

- N_u and N_v define the number of discrete-velocity mesh points in $x-$ and $y-$direction and in this work we assume $N_u = N_v$, although the algorithm is not restricted to this;
- For each direction, the discrete velocities are defined as $c_k \in [c_{min}, \dots, c_{max}]$, for $k = 0, \dots, n_{DV} - 1$ (with $n_{DV} = N_u$ or $n_{DV} = N_v$) and a uniform step size in velocity space $\Delta c = (c_{max} - c_{min})/n_{DV}$. Furthermore,

$c_{min} = -c_{max}$. The implementation of the specular-reflection boundary condition in the current quantum algorithm requires this assumption;

- Discrete velocities are indexed from 0, i.e. the most negative value, to $n_{DV} - 1$ representing the discrete-velocity with the largest positive value

In all cases, the simulations will be initialized with an equilibrium Maxwellian distribution defined for a initial flow state.

4. Reservoir technique for the collisionless Boltzmann equation

The reservoir technique was analyzed and applied to Godunov-type schemes for gas dynamics with the aim of achieving zero or very low numerical diffusion by Alouges et al.[38]. In their work it was applied to the Collela-Glaz solver, showing that for the Sod tube test problem impressive accuracy can be achieved when compared to results from finite-volume methods involving higher-order reconstructions (MUSCL, ENO WENO), despite the first-order accuracy of the stencil used in the advection step.

The motivation behind the developed time-integration method based on the reservoir technique for the collisionless Boltzmann equation is similarly the reduction of numerical dissipation for the first-order method preferred here over more advanced time-integration methods and spatial discretization methods (MUSCL, ENO, WENO, etc.).

4.1. Finite-Volume method for one-dimensional collisionless Boltzmann equation

For illustration purposes, a one-dimensional uniformly spaced finite-volume domain is considered with cell (center) index j. The cell interface between cells $j-1$ and j is denoted with index $j-1/2$ and similarly data related to right cell interface of cell j (connecting cells j and $j+1$) are indexed as $j+1/2$. The uniform cell spacing is Δx. A discrete-velocity method (DVM) with uniformly spaced segments based on the trapezoidal integration rule is employed with n_{DV} discrete velocities to discretize the phase space of the one-dimensional collisionless Boltzmann equation. The discrete velocities are defined as $c_k \in [c_{min}, \ldots, c_{max}]$, for $k = 0, \ldots, n_{DV} - 1$ and a uniform step size in velocity space $\Delta c = (c_{max} - c_{min})/n_{DV}$. Furthermore, $c_{min} = -c_{max}$. It is assumed that all discrete velocities are non-zero, making the system of equations strictly hyperbolic. The reduced particle distribution function

$f(x_j, c_k; t^n)$ in cell j for discrete-velocity k at time level t^n is denoted here as $f^n_{k;j}$. Equivalently $g^n_{k;j}$ for reduced particle distribution function $g(x_j, c_k; t^n)$. Using upwind fluxes in velocity space, the discretized one-dimensional collisionless Boltzmann equation then becomes,

$$\begin{pmatrix} f_{k;j} \\ g_{k;j} \end{pmatrix}^{n+1} = \begin{pmatrix} f_{k;j} \\ g_{k;j} \end{pmatrix}^{n} - c_k \frac{\Delta t_n}{\Delta x} \begin{cases} \begin{pmatrix} f_{k;j} \\ g_{k;j} \end{pmatrix}^{n} - \begin{pmatrix} f_{k;j-1} \\ g_{k;j-1} \end{pmatrix}^{n} & \text{for} \quad c_k > 0 \\ \begin{pmatrix} f_{k;j+1} \\ g_{k;j+1} \end{pmatrix}^{n} - \begin{pmatrix} f_{k;j} \\ g_{k;j} \end{pmatrix}^{n} & \text{for} \quad c_k < 0 \end{cases} \tag{11}$$

The Euler forward-in-time integration as used in this equation clearly limits the admissible time-step according to the CFL criterion to $\Delta t_n \leq \frac{\Delta x}{c_{max}}$ where c_{max} represents the largest discrete velocity in absolute value.

Considering Equation (11) we can observe that if we use a first-order accurate method in space (i.e. distribution functions are assumed constant within each cell), time-integration with CFL=1 leads to an exact propagation for the distribution function(s) $f_{k;j}$ and $g_{k;j}$ corresponding to the largest discrete velocity in absolute value, i.e. k is such that $|c_k| = c_{max}$. For all other indices k, the distribution functions will move less than a cell width Δx during the time step Δt_n, leading to the need to interpolate the discretized solution within each cell and therefore introducing numerical dissipation. The reservoir technique as applied here to collisionless Boltzmann equation is aimed at avoiding this and involves 'exact' propagation of each of the distribution functions from one cell-center to next neighbors during the time-integration process. In the context of the quantum algorithms described in subsequent sections, this property of 'streaming' of distribution functions from one cell-center to another is an essential aspect enabling a relatively simple implementation. A further helpful characteristic of the considered reduced collisionless Boltzmann system is that both reduced distribution functions convect at the same discrete velocities, i.e. the convection operator in both equations have the same eigenvalues.

4.2. Reservoirs, CFL counters and variable time-step

For each cell face, CFL counters are introduced as $c^n_{k;j\pm1/2}$ for $k = 0, \ldots, n_{DV}-1$. At the start of the time-integration these counters are all initialized to zero. These counters will be updated during each time step by $|c_k|\Delta t_n/\Delta x$. For convenience, the following temporary variables are introduced,

$$\mathcal{C}^{n+1}_{k;j\pm1/2} = c^n_{k;j\pm1/2} + |c_k|\frac{\Delta t_n}{\Delta x} \tag{12}$$

Since for the considered system, the eigenvalues (discrete-velocities) in the upwind discretization are identical for each cell face, the CFL counters are identical for each cell face as well. This greatly simplifies the following integration method since only a single set of counters for all discrete velocities have to be considered rather than a set for each cell face.

The time-step Δt_n is limited such that $\mathcal{C}^{n+1}_{k;j\pm1/2} \leq 1$. Here, the time-step Δt_n is selected by finding the minimum among all j and k,

$$\Delta t_n = min_{j,k}\Big([1 - c^n_{k;j\pm1/2}]\frac{\Delta x}{|c_k|}\Big) \tag{13}$$

This choice of time step will result in at least one of the CFL counters to reach 1 in the considered time step, i.e. $\mathcal{C}^{n+1}_{k;j\pm1/2} = 1$ for one or more discrete velocities k (typically only for one eigenvalue during each time step), while never exceeding the value of 1. The underlying idea of the reservoir technique is to introduce reservoirs for each cell face $R^n_{k;j\pm1/2}$ for $k = 0, \ldots, n_{DV} - 1$ which are initially set to zero at the start of the time integration. At each time step we fill up the reservoirs $R_{k;j\pm1/2}$ with the current numerical flux difference upwinding depending on the sign of c_k. As long as the CFL counter for the considered discrete velocity remains less than 1, this process continues, i.e. with the CFL counters gradually updated according to

$$c^{n+1}_{k;j\pm1/2} \quad = \quad c^n_{k;j\pm1/2} + |c_k|\frac{\Delta t_n}{\Delta x} \tag{14}$$

Furthermore, temporary variables $\tilde{f}_{k;j\pm1}$ and $\tilde{g}_{k;j\pm1}$ are introduced to facilitate the update due to the numerical flux difference upwinding for both reduced distribution functions. The idea is that the temporary variables will be updated when the CFL counter hits the value 1, while the update will go into the reservoirs for CFL counters below 1. The temporary variables $\tilde{f}_{k;j\pm1}$ and $\tilde{g}_{k;j\pm1}$ are used to update the solution to the new time level $n+1$, therefore the only non-zero updates will occur for the discrete velocity (or velocities) for which the CFL counter reached 1. Once a CFL counter $c^n_{k;j\pm1/2}$ for a discrete velocity k has reached 1, this counter as well as the reservoir associated with this discrete velocity will be set to 0 before the start of the next time-step.

4.3. Simplification of scheme to facilitate implementation

In the time integration described above we can identify a cycle during which data for each discrete velocity gets updated at least once, i.e. this

Table 1: Reservoir method for velocity-space boundaries ± 8 reference velocity units ($\sqrt{2RT_r}$) and $\Delta x = 1$.

n_{DV}	n_{cycle}	Δc	$c_{k,min}$	$c_{k,max}$	T_{cycle}	ave. Δt
16	49	1.000	0.5000	7.5000	2.0	0.04167
32	213	0.500	0.2500	7.7500	4.0	0.01887
64	825	0.250	0.1250	7.8750	8.0	0.00971
128	3327	0.125	0.0625	7.9375	16.0	0.00481

cycle involves a time $T_{cycle} = 1/c_{k,min}$, with $c_{k,min}$ representing the smallest discrete velocity in absolute value. The number of time-steps depends on the eigenvalue spectrum considered. Table 1 shows the details of one cycle in the reservoir scheme for a one-dimensional problem with velocity space bounds $\pm 8\sqrt{2RT_r}$. For the smallest number of discrete velocities, 49 time steps constitute one cycle. It can be seen that for each doubling of the number of discrete velocities, the number of steps per cycle grows approximately by a factor 4. In all cases, the average non-dimensional time step Δt is significantly smaller than the CFL limit imposed by the largest discrete velocity, i.e. $\Delta x/c_{k,max}$ with $c_{k,min}$ representing the largest discrete velocity in absolute value. In the reservoir method, the reservoir values can be used in the interpolation of data between cells when output is required at a time step that does not correspond to the last step in a cycle. To facilitate the implementation as quantum algorithm, the reservoirs are actually not stored and when output at a time step which does not coincide with the end of a cycle is required, the interpolation of data from a location in between two neighboring grid points is omitted. This will have a small impact on the accuracy of the output, i.e. for certain steps within a cycle the output will not benefit from the smoothing effect of the data interpolation and will therefore be more likely to involve small oscillations.

4.4. Extension to higher dimensions

For two-dimensional problems in the work, it is assumed that the velocity space involves a square domain with a uniform and identical step size in each velocity-space direction. Furthermore, the number of steps and the upper and lower bounds are taken the same for both velocity-space directions. Then, the time integration of the two reduced distribution functions f and g for a two-dimensional problem proceeds by dimensionally splitting the convection step and using the method described for the one-dimensional case above. The number of steps per cycle then remains those shown in Table 1. It is clear that the assumption on velocity-space bounds, i.e. $c_{min} = -c_{max}$ as well as identical step sizes for both c_x and c_y discrete velocities is not ideal for some practical problems. However, for the case considered here involving specular-reflection wall boundary conditions, this choice fits well.

4.5. Extension to collisionless Boltzmann equation with homogeneous force field

So far the collisionless Boltzmann equation (as defined in Equation (6)) has been considered without the presence of a space-homogeneous force field. Including such a force field, the collisionless Boltzmann equation can be written as,

$$\frac{\partial F(\vec{x},\vec{c};t)}{\partial t} + \vec{c}\cdot\frac{\partial F(\vec{x},\vec{c};t)}{\partial \vec{x}} + \vec{F}^{hom}\cdot\frac{\partial F(\vec{x},\vec{c};t)}{\partial \vec{c}} = 0 \tag{15}$$

where $\vec{F}^{hom}$ defines the force field (space homogeneous) that acts on all particles. When integrating this extended equation in time, it can be seen that this force field effectively acts to move the distribution function within the velocity space in a direction defined by the direction of $\vec{F}^{hom}$. The time-integration method described in the present work can be extended to include this homogeneous force field. Specifically, the effect of the force field on the discretized distribution function can be accounted for using a streaming operation for the distribution function acting within velocity space, in contrast to the streaming operation in 'physical' space described in previous sections.

5. Quantum algorithm for collisionless Boltzmann equation

Building on the time-integration based on the reservoir-technique, a quantum algorithm implementing the discrete-velocity method for collisionless Boltzmann equation is introduced. One key aspect is the implementation of the convection step in the discrete-velocity method as a series of streaming operations, as described in the previous section. The quantum algorithm used to impose non-periodic boundary conditions is described in detail in this section, as it represents a key innovation of the present work. In the following, the quantum algorithm is detailed for two-dimensional problems. The implementation for one-dimensional problems follows a logical subset of this, while the extension to three-dimensional problems is also straightforward.

5.1. Data structure and mapping onto state vector

The quantum algorithm is based on the circuit model of quantum computing. More specifically, we encode the problem in a coherent quantum state $|\psi\rangle$ using n_q coherent qubits in the quantum register of a quantum computer and apply a series of unitary transformation using quantum logic gates. For the mapping of the problem onto this state the following approach is:

- For a 2D mesh with $N_x \times N_y$ grid points, we determine the numbers of qubits associated with this mesh size, i.e. $n_{q,x} = log_2(N_x)$ and $n_{q,y} = log_2(N_y)$;
- Assuming we have a uniformly spaced discrete-velocity mesh with $N_u \times N_v$ discrete-velocity mesh points, the number of qubits representing the state-space mesh is determined: $n_{q,u} = log_2(N_u)$ and $n_{q,v} = log_2(N_v)$
- For cases with solid-wall boundary conditions, we introduce an additional qubit working as a flag identifying parts of the domain that are with the fluid and parts of the domain with the solids. More specifically, this qubit is set to $|1\rangle$ when a cell is in the considered fluid domain, or $|0\rangle$ when the considered cell is part of a solid. This qubit is termed the 'BC' qubit in the quantum circuits presented here. Adding this qubit effectively doubles the size of the state vector. Only for the half of the state vector associated with 'BC' qubit in state $|1\rangle$, the streaming operations representing convection are performed. The other half of the state vector is not included in the streaming operations, and in the current implementation is used to represent the solution within solid bodies;
- For a single-species simulation, a further qubits is added to create the space for the storage of two reduced distribution functions. This qubit is termed the 'g' qubit in the quantum circuits presented here;
- For a multi-species simulation with N_{sp} species, a further $log_2(N_{sp})$ qubits is added;
- For practical implementations, typically a number of additional 'ancilla' qubits will be needed. This will be discussed later in this work.

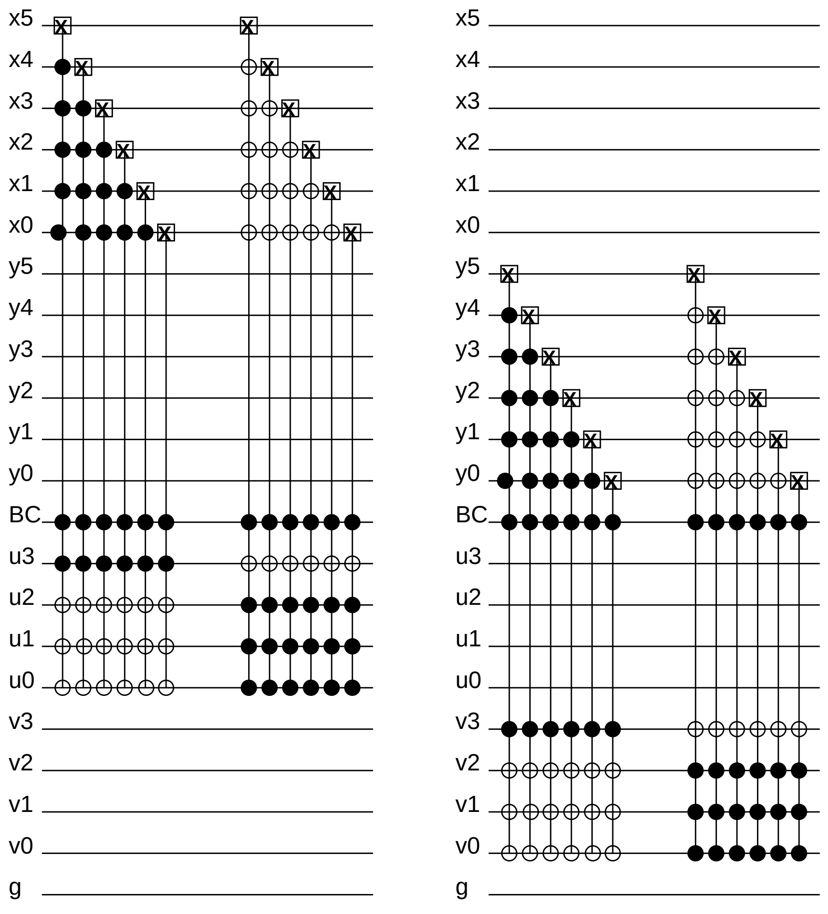
In the qubit register, starting from the left, first the $n_{q,x}$ qubits corresponding to the $x-$direction in the mesh are stored, followed the $n_{q,y}$ qubits for the y-coordinate direction. Then, the 'BC' qubit is stored. The next group of $n_{q,u}$ qubits correspond to the $x-$direction in the velocity-mesh, followed by the $n_{q,v}$ for the $y-$direction in the velocity-mesh. For single-species simulation, the next qubit will be the 'g' qubit. For multi-species simulations the additional $log_2(N_{sp})$ qubits are padded to the right. Any ancilla qubits will be added to the right-hand side of this register.

For example, a single-species flow on a 64×64 mesh with 16×16 discrete velocities and a single ancilla qubit will be represented by a qubit register as,

$$|q_{x,4}q_{x,3}q_{x,2}q_{x,1}q_{x,4}|q_{y,4}q_{y,3}q_{y,2}q_{y13}q_{y,0}|q_{BC}|q_{u,3}q_{u,2}q_{u,1}q_{u,0}|q_{v,3}q_{v,2}q_{v,1}q_{v,0}|q_g|q_{a0}\rangle \tag{16}$$

5.2. Implementation of the convection step in the quantum algorithm

Previously, the principles of performing a left- and right-streaming step on a regular mesh were discussed in Section 2.4 for a single function discretized on a 2D mesh. In the discrete-velocity method as implemented here, vector data representing the values of the distribution for each discrete-velocity in the considered cell needs to be streamed left or right at selected time steps in the time-integration process. Building on the circuits discussed in Section 2.4, the multiple-control NOT gates now need to be extended to also include a test on the qubits value representing the discrete-velocity indices. This enables streaming of selected discrete-velocities rather than streaming all data in one step. Figure 2 shows the quantum circuit implementation of the 'streaming' operation in both x- and y-direction for a two-dimensional flow problem on a 64×64 mesh and with 16×16 discrete-velocities in state-space. The circuits shown in Figure 2(a) represents the streaming for the data with qubit indices defined in Equation (16) with $|q_{u,3}q_{u,2}q_{u,1}q_{u,0}\rangle = |1000\rangle$ in the positive $x-$direction and with $|q_{u,3}q_{u,2}q_{u,1}q_{u,0}\rangle = |0111\rangle$ in the negative $x-$direction. As can be seen the qubits representing the x-coordinates form the target qubits, i.e. the cross represents a negation under the conditions that the control qubits have the required status: full circles mean that the control qubit should be in state $|1\rangle$, while open circles represent the control state $|0\rangle$. The First multi-qubit operation involves 10 control qubits, i.e. 5 from qubits representing x-coordinates, a further due to the 'BC' control qubits and finally the 4 qubits indexing the u-velocities. For the 4 subsequent operations, the target qubit shifts by one and the number of control qubits is reduced by one qubit representing x-coordinates. The difference in



(a) x-direction (b) y-direction

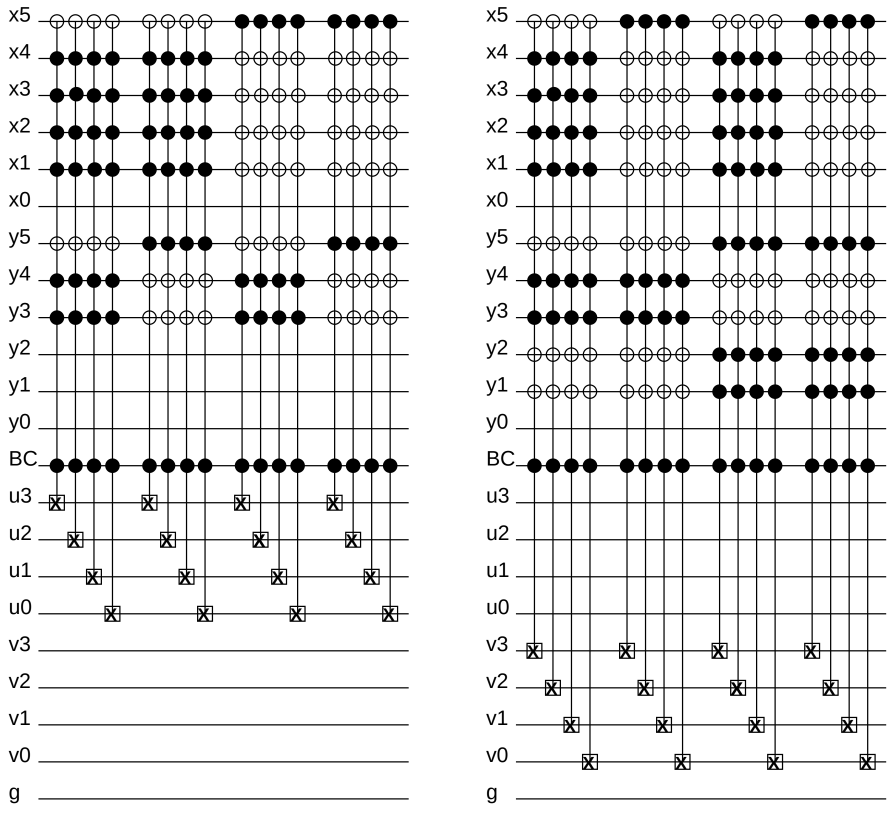
Figure 2: Quantum circuit representation of streaming operation in x-direction (a) and y-direction (b). Circuit for 64×64mesh in space and 16×16 discrete-velocities in state-space

streaming in positive or negative direction is manifested by the sign of the control qubits in the part of the register representing the considered coordinate direction. By moving the target qubits to the y-coordinate part of the qubit register and moving to the v-velocity qubits, the circuits representing streaming in y-direction are created. For the streaming in the y-direction, Figure 2(b) shows the required circuits. A key point that should be noted for the quantum circuits presented here, is that periodic boundary conditions will be imposed on the edges of the domain. Specifically, a quantum walk to the left applied to the left-most lattice point will move this data to the right-most lattice point. Similarly for the quantum walks moving right associated with a positive discrete velocity in the considered coordinate direction, data leaving the right boundary will move to the left-most lattice point. Although the circuits shown in Figure 2 will perform the correct convection operations, an important practical constraint needs to be considered. For the quantum computer implementations achieved so far and those foreseen for the near future, the kind of multi-qubit controlled NOT operations used here cannot be implemented. A small set of native gates will be available which most likely includes NOT, $CNOT$ and the three-qubit Toffoli gate. The solution around this limitation is the introduction of *ancilla qubits*, which can be regarded as the quantum equivalence of additional workspace in the memory of a classical computer. Then circuits involving multi-qubit operations involving a large number of control gates can be transformed into circuits with more qubits and a larger number of gate operations, however now with a smaller number of control qubits. For the presentation of the current algorithm, these details are not essential and are left for future work.

5.3. Implementation of specular-reflection boundary conditions

Since the flow in the free molecular limit is determined by collisions between the molecules and the boundary surfaces, the nature of this interaction between the molecule and the boundary is all important. However, the details of the reflection process vary in a complex manner with the nature of the surface and the velocity of the incident molecule, and it is necessary to treat idealized models[37]. In the present algorithm, the classical model of a specular-reflection wall is included. Specular reflection is an elastic process in which the velocity component normal to the wall is reversed and that parallel to the wall is retained. In contrast, in a diffusely reflected molecule its temperature adjusts towards that of the surface and it is re-emitted in a random direction with a Maxwellian speed distribution. In the present proof-of-concept implementation, it is assumed that the gas interaction with solid walls takes place through specular reflections. Imposing the diffuse boundary conditions is part of future research.

In the proposed algorithm, to approach used to impose specular-reflection boundary conditions can be summarized as follows:



(a) x-direction (b) y-direction

Figure 3: Quantum circuit imposing specular-reflection boundary conditions for blunt-body test case ($L/H = 4$). (a) x-direction and, (b) y-direction (b). Circuit for 64×64mesh in space and 16×16 discrete-velocities in state-space

- A solid object is defined with solid walls aligned with either $x-$ or $y-$ axis. The 'wall' locations are selected such that these fall halfway between two neighbouring lattice points;
- Based on the defined solid wall locations, all lattice point nearest to a solid wall, on the inside of a solid body, are selected. It is in these lattice points within the solid object that the wall-normal velocity will be reversed. An important part of the quantum circuits discussed here will therefore be use of the qubits representing $x-$ and $y-$ lattice coor-

dinates as control qubits in the multiple-input controlled-NOT gates. This ensures that any reversing operation will only take place in selected lattice points;

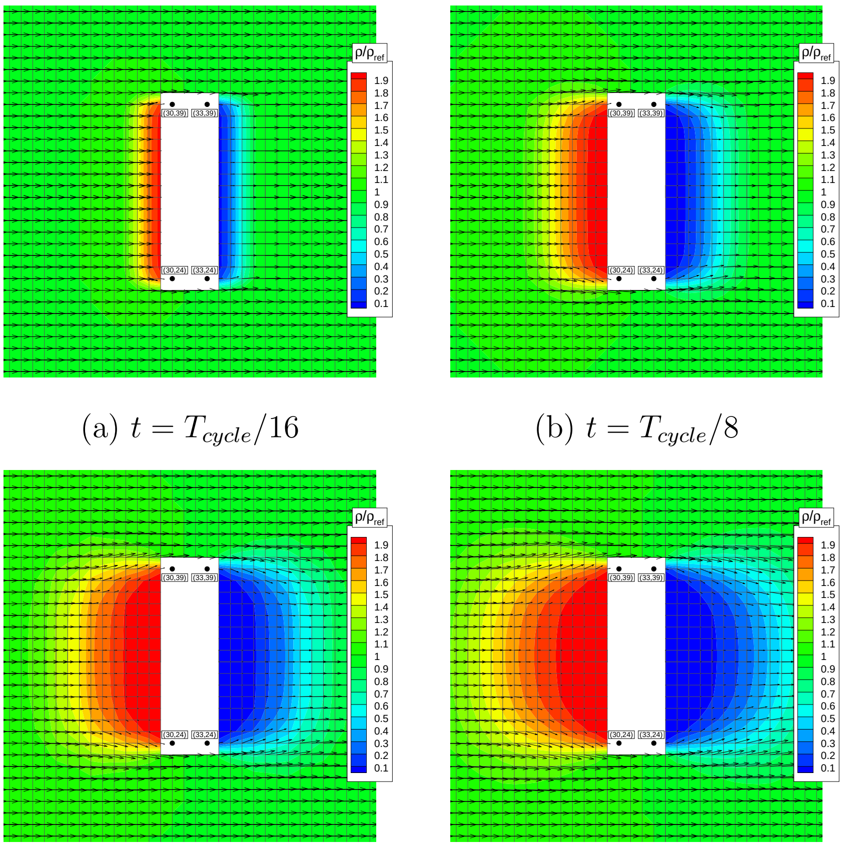
- As discussed in Section 5.1, a 'BC' qubit is used in the quantum register, such that the $|1\rangle$ state of this qubit defines the half of the state vector for which streaming operations are employed. For this half of the state vector, these streaming operations will also act on the lattice points within solid objects. Imposing specular-reflection boundary conditions will interfere with the solution within solid bodies. The current implementation was designed in such a way that for the regions 'within' solid bodies, the part of the state vector associated with 'BC' qubit in state $|0\rangle$ would be used when creating output of the algorithm. Clearly, alternative designs are also feasible.

- During the course of a simulation, discrete values for distribution functions for velocities pointing into a solid wall will stream into a solid body, i.e. the data will move to one of the lattice points highlighted above. For each time step of the reservoir-technique based integration method it is known which of the discrete velocities was streamed during that particular time step. For these discrete velocities, the specular-reflection condition is applied by reversing the wall-normal molecular velocity in the nearest lattice site within the solid body;

- In the implementation used here, the discrete velocity space is symmetric with respect the origin centered at zero velocity, as discussed previously in Section 4.1. The discrete velocities in each direction are numbered 0 (negative velocity with largest magnitude) to $N_{DV} - 1$ (positive velocity with largest magnitude), where N_{DV} represents the number of discrete velocities in the considered direction. Reversing the velocity is then achieved by applying NOT operation on all qubits defining the discrete-velocity in this direction. Effectively, the discrete distribution function value associated with a molecular velocity going into the wall has now been assigned to the corresponding opposite outgoing molecular velocity such that it can be used in the streaming operations during the next time step in the simulation.

For a 64×64 two-dimensional mesh and 16×16 discrete velocities, the specular-reflection boundary condition implementation based on quantum circuits is shown in Figure 3. The flow around a rectangular solid object with a height of 16 mesh spacings and with of 4 mesh spacings centered in the flow domain is considered. The flow field for a simulation using a finer 64×64 velocity-space discretization is shown in Figure 4 at four time instances following the initialization of the flow field with a uniform free-molecular flow at Mach 2. The figure shows the (i, j) index range involved in the internal part of the body where the velocity needs to be reversed. In the quantum circuit implementation, we use separate circuits to impose the boundary condition in x- and y-directions. Figure 3(a) shows the circuit imposing the boundary conditions in x-direction. In the considered circuit, 4 groups of gate operations (in the horizontal direction) can be observed. The first group imposes the data-swap in lattice sites with $i = 30, j = 24, \ldots 31$ and $i = 31, j = 24, \ldots 31$ representing the lower half of the front face of the solid body. The second group similarly represents the upper half of the front face, i.e. $i = 30, j = 32, \ldots 39$ and $i = 31, j = 32, \ldots 39$. The third and fourth group of gate operations similarly impose the boundary condition on the lower and upper half of the rear face of the body. In the quantum circuit, multiple-qubit controlled NOT gates are used to reverse the velocity in the $x-$direction within selected lattice points. These selected lattice points are defined using the qubits representing $x-$ and $-y$ coordinates as control qubits. Since the velocity reversal in these points is only to be performed when 'BC' qubit is $|1\rangle$, this 'BC' qubit also acts as control qubit. Figure 3(b) shows the circuit imposing the boundary conditions in y-direction, again consisting of 4 groups of gate operations. In this case, the first two groups represent the left and right halves of the lower wall of the body, while the boundary condition on the upper wall of the body is represent by the third and fourth group for the left and right halves, respectively.

Similar to the quantum circuit implementation for the 'streaming' operations, the implementation for the specular-reflection boundary conditions clearly involves multiple-qubit gates with a large number of control qubits. This means that in a practical implementation on a quantum computer with a realistic set of native quantum gates, a circuit transformation needs to be performed using additional ancilla qubits, as shown previously for the streaming operation.



(a) $t = T_{cycle}/16$ (b) $t = T_{cycle}/8$

(c) $t = 3T_{cycle}/16$ (d) $t = T_{cycle}/4$

Figure 4: Free-molecular Mach 2 flow field around blunt body, 64×64 discrete-velocity space. Specular-reflection boundary conditions are applied. Results show evolution of flow field at 4 time instances following initialization with uniform free-stream flow.

6. Complexity analysis

The application of the proposed quantum algorithms on quantum computer hardware involves three phases. Firstly, the quantum state needs to be initialized in an appropriate initial state. Then the main part of the algorithm is applied in terms of evolving the quantum state towards to the desired solution at a later time instance. Once this is completed, quantum measurement is applied to obtain the desired output in terms of classical data. All three phases are detailed in the following paragraphs. In the analyses, an d-dimensional domain (e.g. 2D in this work) with domain size D

in all considered direction is assumed with h the constant mesh width (cell dimension in finite-volume scheme), assumed the same in each considered spatial direction. In this section, N_v represents the number of discrete velocities for direction (assumed identical for each direction). For the time evolution a time T is considered, while for the quantum algorithm, a desired accuracy δ is specified.

6.1. Required memory

For a classical implementation (for the explicit time-integration methods considered), it can be assumed that the memory required scales directly proportional to the number of cells as well as the number of discrete velocities, i.e. memory cost is $O(N_v^d(D/h)^d)$. For the proposed quantum algorithm, the required number of qubits for memory storage scales as $log_2(N_v^d(D/h)^d)+n_{anc}$, with n_{anc} representing the number of required ancilla qubits. In the present work, n_{anc} will always be smaller than the theoretical minimum number of qubits $log_2(N_v^d(D/h)^d)$. Since the required repeated realizations do not involve storage within the same coherent state vector, an exponential improvement in complexity occurs in terms of memory.

6.2. Complexity of reservoir-technique method on classical computer

Before analyzing the complexity of the quantum algorithm, the complexity of a classical implementation of the reservoir-technique based DVM solver is analyzed, and compared to more 'conventional' DVM methods. For a conventional DVM method it is assumed that the discretized distribution functions for all discrete velocities are used for each considered time step (different than in the reservoir-technique). Stability requirements impose limits on the time-step mainly as function of the maximum discrete velocity used (in absolute terms), i.e. independent of number of discrete velocities. Therefore, the number of time step required scales directly proportional to the time interval T considered. The complexity of performing the flux and residual calculation is then $O(TdN_v^d(D/h)^d)$. Also, in a typical DVM method, the CPU time for flux and residual calculations dominate CPU time for calculation of updates to distribution functions as well as the cost of imposing boundary conditions. Therefore time complexity can be summarized as: $O(TdN_v^d(D/h)^d)$. In the reservoir-technique based implementation, it can be observed that the period of a single cycle scales as the reciprocal of the smallest discrete-velocity $(\pm\frac{\Delta c}{2})$, i.e. the period of one cycle increases directly proportional to N_v for fixed maximum values of discrete velocity. The number of time-steps within a cycle increases as N_v^2. Within a cycle, different discrete velocities are involved for different time steps. On average, the number of updates for each discrete velocity per cycle scales as N_v. Therefore, the number of flux and residual calculations required for each cycle scales as $dN_v^{d+1}(D/h)^d$. Since the period per cycle increases linearly with N_v, it can be seen that the time complexity for the reservoir-technique DVM method is the same as that for a 'conventional' DVM, i.e. $O(TdN_v^d(D/h)^d)$.

6.3. Complexity of time-evolution phase of simulation

In the quantum implementation of the reservoir method, streaming operations represent the flux evaluation, residual calculation and calculation of updates to discrete distribution function. Figure 1(a) shows how data for two discrete velocities in $x-$direction (i.e. with opposite signs) is streamed in $x-$direction. Data associated with all discrete velocities in the 'other' coordinate direction is included in these operations, a key factor in achieving efficiency. The number of multi-qubit gate operations required for this streaming in positive and negative direction scales as $log_2(D/h)$ for each discrete velocity considered. For all discrete velocities for one coordinate direction, a scaling as $N_v log_2(D/h)$ for the number of multi-qubit gate operations follows. For d-dimensional problems, the complexity becomes $O(dN_v log_2(D/h))$. Since time step used is not proportional to N_v, the time complexity for a time interval T becomes $O(TdN_v log_2(D/h))$. It is important to note that the multi-qubit gate operations required in the current work are multi-qubit controlled-NOT gates, e.g. C^pNOT, representing $p+1$-qubit gates with p control gates. Analysis of our circuit-based implementation shows that p scales as $log_2(D/h)+log_2(N_v)$. When compiling the current quantum circuits for actual quantum hardware, the used multi-qubit controlled-NOT gates need to be converted into native gates on the considered quantum hardware, leading to a further increase in the number of gate operations that strongly depends on the compilation technique employed.

6.4. Complexity of specular-reflection boundary condition

The quantum circuits used to impose specular-reflection boundary conditions were discussed in Section 5.3. The previously introduced 'BC' qubit can be regarded as representation of the memory overhead of this boundary condition. In terms of time complexity, we assume that the number of lattice points at which the specular-reflection boundary condition is to be applied is a constant fraction of the total number of lattice points when considering increasing problem size, i.e. the required number of multi-qubit gate operations scales as $dN_v log_2(D/h)$ for each time step. The time complexity of the specular-reflection boundary condition therefore matches that for the convection step when expressed in terms of required number of multi-qubit gate operations. Since the boundary conditions are imposed on a subset of all lattice points, the multi-qubit gate operations required will involve a larger number of control qubits than used in the convection step of the algorithm, since a subset of the qubits representing $x-$ and $y-$ coordinates are involved as well.

6.5. Complexity of initialization

As mentioned in Section 2.5 it is important to avoid the complexity of order N associated with the preparation of a completely arbitrary quantum state in an N-dimensional Hilbert space. For the initial solutions used here, i.e. the same Maxwellian distribution function in each of the lattice points in the considered domain, the complexity can be expected to be much lower, based on the fact that the Maxwellian initial solutions used here are log-concave with respect to the particle velocities defined by the corresponding qubits in the register. Since the same initial solution is set throughout the computational domain, there is no dependency on the qubit states defining lattice location. The 'BC' qubit does not introduce significant challenges, since the flow will be initialized in the same state for the 'BC' qubit in state $|0\rangle$ and $|1\rangle$. For the algorithm to work, it is required that ancilla qubits are initialized to $|0\rangle$. In future work, the complexity of initialization will be analyzed in more detail. However, from the initial analysis performed here and the findings of Grover and Rudolph[30], we can expect that initializing will not render the proposed quantum algorithm ineffective by requiring exponential resources. Instead, a complexity polynomial in the number of qubits appears to be possible.

6.6. Obtaining output from simulation

Obtaining output from the algorithm poses a major challenge. In general quantum state tomography can be applied to learn the full state. However, this leads to a scaling with the state space size. As mentioned previously in Section 2.5, the intended use of the proposed algorithm does not involve obtaining the full flow field. If we assume that N_{out} amplitudes are sought at end of a simulation, then using amplitude estimation, the number of realizations can be estimated to scale as N_{out}/δ, for a required accuracy of δ. The aspect of applying amplitude estimation in the current context needs further investigation in future work. However, the $1/\delta$ dependency of the number of realizations represents a key challenge to achieving a potential meaningful speed up relative to classical implementations. A key benefit of the quantum algorithm that remains despite the output/measurement challenges is the exponential reduction in memory when expressed in terms of qubits.

Table 2: Reservoir method parameters for blunt-body test case, for $\Delta x = 1$.

n_{DV}	Mach	n_{cycle}	Δc	$c_{k,min}$	$c_{k,max}$	T_{cycle}	ave. Δt
32	2	213	1/3	1/6	31/6	6.0	0.02830
32	6	213	2/3	1/3	31/3	3.0	0.01415
64	2	825	1/6	1/12	63/12	12.0	0.01456
64	6	825	1/3	1/6	63/6	6.0	0.007282
128	2	3327	1/12	1/24	127/24	24.0	0.007216
128	6	3327	1/6	1/12	127/12	12.0	0.003608

7. Validation and demonstration of method

In the literature, the problem of a collisionless gas expanding into a vacuum has been studied extensively, e.g. Narasimha[39], and the key differences with an equivalent gas dynamics solution includes the absence of shock waves and the resulting absence of interacting shocks when the expansion takes place in a confined domain. The initial validation of the developed method included the quantitative comparison of the analytical solutions from Narasimha[39] one-dimensional expansions into vacuum with computed results at different time instances. This established that for this expansion test case, 64 discrete velocities were sufficient to provide a good quantitative agreement, while a further velocity-space refinement mainly provided smoother density and velocity variations. For 32 discrete velocities, only in the early stages was a smooth density profile obtained, while at later stages, significant oscillations occurred in the density profile. In the interest of brevity, this validation is not further detailed here.

As a further validation, we consider the free-molecular flow around a rectangular blunt body. It is assumed that, initially the gas is at rest and is in thermal equilibrium at temperature T. Then, at $t = 0$, the gas acquires a

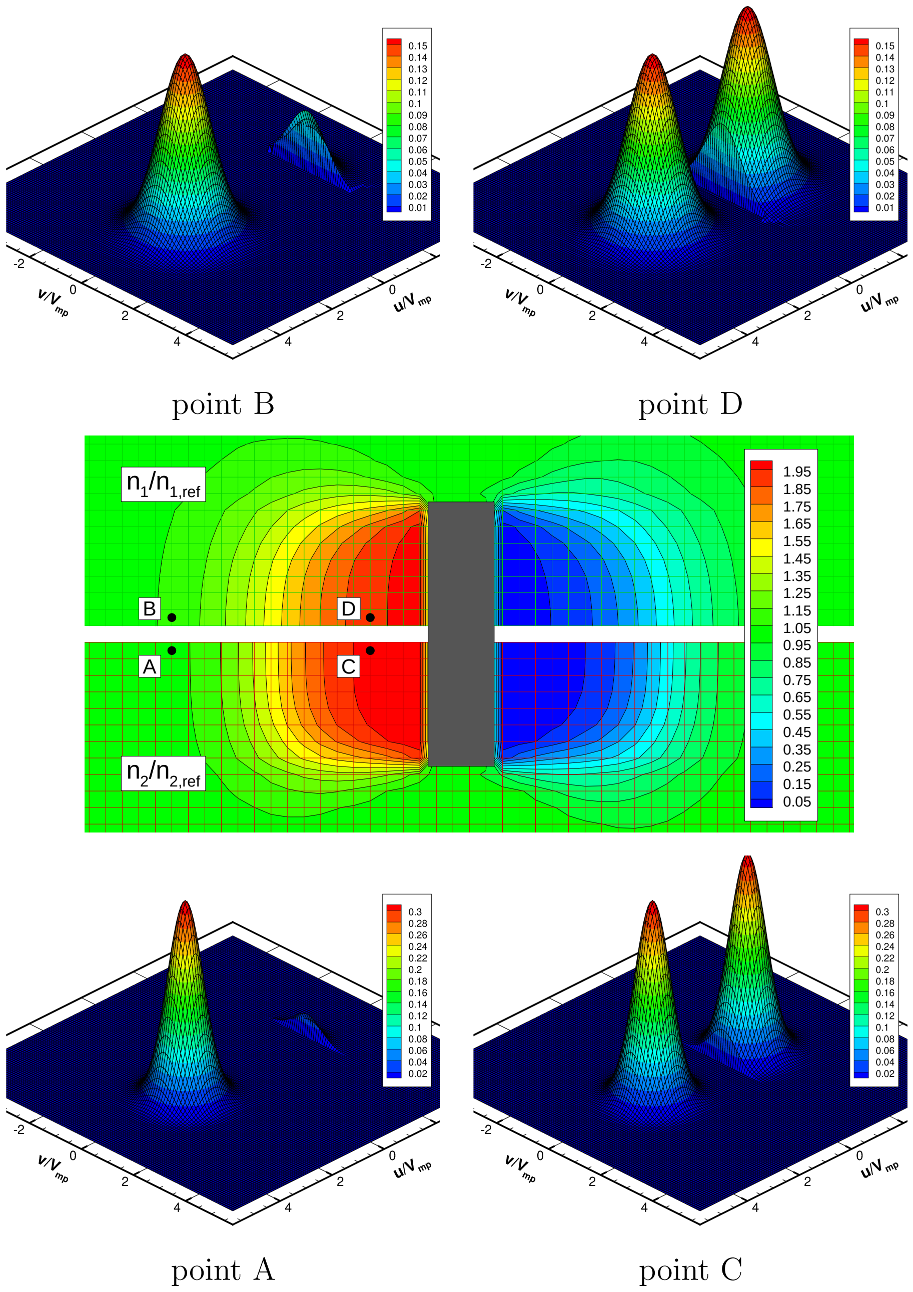


Figure 5: Mach 2 flow of binary gas mixture ($m_2/m_1 = 2$) around rectangular body. Cartesian mesh (64×64) around plate, 128×128 discrete-velocity mesh.

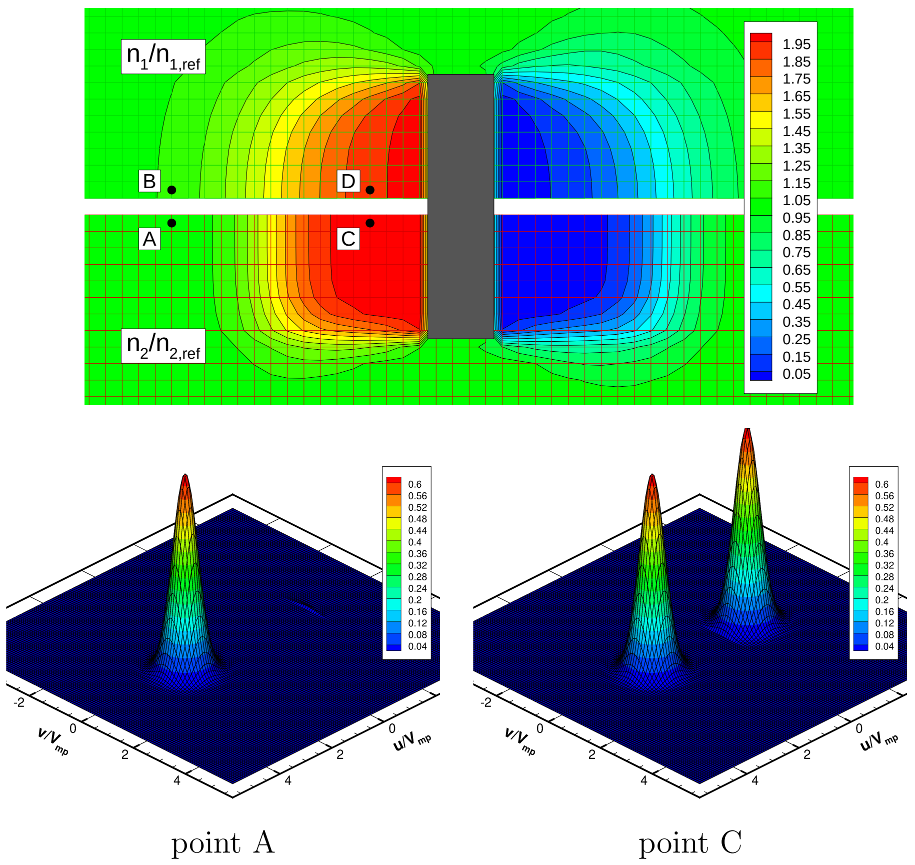
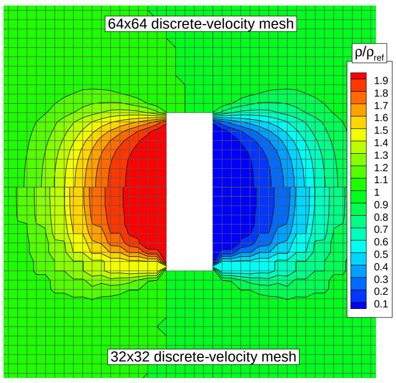


Figure 6: Mach 2 flow of binary gas mixture ($m_2/m_1 = 4$) around rectangular body. Cartesian mesh (64×64) around plate, 128×128 discrete-velocity mesh.

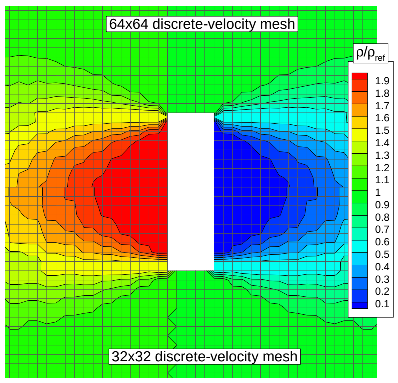
velocity U_∞ towards this blunt body. Before presenting validation cases, Section 7.1 first highlights the typical flow features encountered when applying the proposed method to a free-molecular flow of a binary gas mixture around a solid body.

7.1. Free-molecular flow around solid body

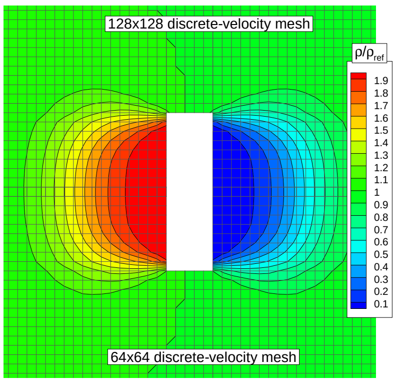
As an illustration of the binary mixture flows considered here, Figure 5 shows an example for a Mach 2 flow with $m_2/m_1 = 2$ around a rectangular body. In the centre, the number densities for both species are shown. For two locations upstream of the body, the particle distribution functions (f_1 and f_2) for both species are presented in velocity space. For species 1 at the top of the figure, while the two bottom plots show distribution functions for



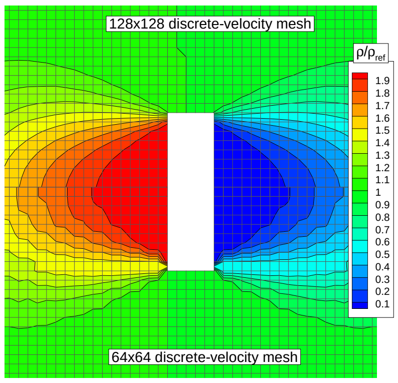
(a) 32 vs 64: $t = T_{cycle}/4$



(b) 32 vs 64: $t = T_{cycle}/2$



(c) 64 vs 128: $t = T_{cycle}/4$



(d) 64 vs 128: $t = T_{cycle}/2$

Figure 7: Free-molecular Mach 2 flow field around blunt body. Specular-reflection boundary conditions are applied. Effect of velocity-space discretization is shown for 2 time instances following initialization with uniform free-stream flow. T_{cycle} used here is for 64×64 discrete-velocity mesh.

species 2. In the simulations, the distribution functions everywhere in the domain were initialized with the Maxwellian distributions for the considered mean velocity, temperature and molecular mass. For the heavier species the direct result of the increased molecular mass is a more 'pointy' distribution function (i.e. on average molecular velocities are closer to the mean value than for a lighter species) for the same temperature as compared to species 1. Figure 5 highlights the main effect of the considered flows, i.e. the reflection of particles from the solid surfaces, leading to a secondary 'spike' forming in the particle distribution functions in velocity space. The formation of these secondary spikes is time-dependent starting from the initial uniform flow, i.e. for parts of the distribution function representing the faster moving molecules this spike forms more quickly than for slower-moving molecules. Also, for a point further upstream of the body, this process takes more time since it takes longer to reach the considered location for particles reflecting off the solid walls, as can be seen in the different results for point A and C and, similarly, B and D. In case the ratio of the molecular masses of the species is increased, the effect of the more confined particle distribution functions in velocity space for the heavier species becomes more pronounced, as can be seen from the example results for a Mach 2 flow with $m_2/m_1 = 4$ around the rectangular body. The results shown in Figure 5 and 6 highlight the importance of the boundary conditions imposed and their effect on the particle distribution functions even at large distances from the solid walls. Due to the absence of inter-particle collisions, the often non-smooth distribution functions do not relax towards a local Maxwellian distribution function.

7.2. Discrete-velocity mesh convergence

Figure 7 shows the effect of refining the velocity-space discretization for the blunt-body flow at Mach 2. Table 2 shows the parameters used for the test case involving the flow around the rectangular object. For the 32×32 and 64×64 velocity-space meshes, Figure 7(a)-(b) show the comparison of the computed density for the time instances, $t = T_{cycle}/4$ and $t = T_{cycle}/2$. It can be seen that although overall the results agree well, the finer discretization can be seen to lead to smoother derived quantities such as density obtained from numerically evaluation the moment of the distribution function in velocity space using a trapezoidal integration. As similar comparison is shown in Figure 7(c)-(d) for the 64×64 and 128×128 velocity-space discretizations. Here, the differences between the computed densities for two time instances are significantly smaller, illustrating the velocity-space mesh convergence for meshes of 64×64 and finer.

7.3. Comparison with analytical solution for piston-driven free-molecular flow

For the free-molecular flow around the rectangular blunt body considered here, the flow solution along the stagnation streamline that is created in the ensuing two-dimensional flow can be tested against the analytical solution obtained by Bird[37] for a piston-driven free-molecular flow. In Bird's solution, a free-molecular flow driven by an infinite plane piston is considered.

Initially, the piston is at rest and the gas is in thermal equilibrium at temperature T. Then at time $t = 0$ the piston acquires a velocity U_∞ in the direction normal to its face. For convenience, a coordinate system moving with the piston was considered with the x-direction normal to the plane of the piston. As discussed by Bird[37], the problem is equivalent to a plane wall being instantaneously inserted into a uniform stream of velocity $-U_{mean}$. The flow upstream of the wall corresponds to the compression side of the piston, and this will be affected by the molecules reflected from the wall and also by the absence of the molecules which would have previously come from positions downstream of the wall.

For specular reflection boundary conditions on the piston, the number density as function of x and t can then be written as,

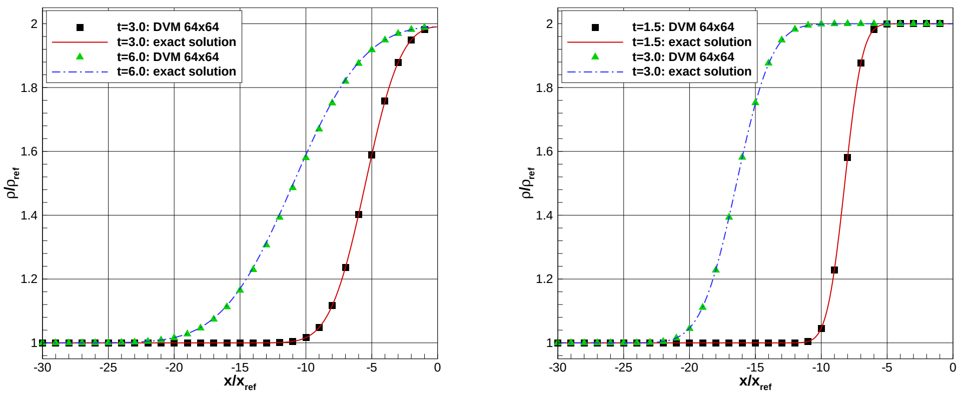
$$\begin{aligned} n(x,t) &= n_\infty + \frac{n_a}{\sqrt{2\pi RT}} \int_{c_x = x/t}^{\infty} \Big\{ \mathtt{exp}\Big[-\frac{(U_\infty - c_x)^2}{2RT} \Big] - \mathtt{exp}\Big[-\frac{(U_\infty + c_x)}{2RT} \Big] \Big\} dc_x \\ &= n_a \Big[1 + \frac{1}{2} \Big\{ \mathtt{erf}\Big(\frac{x/t + U_\infty}{\sqrt{2RT}} \Big) - \mathtt{erf}\Big(\frac{x/t - U_\infty}{\sqrt{2RT}} \Big) \Big\} \Big] \end{aligned} \tag{17}$$

where n_∞ is the number density of the ambient molecules. For the two-dimensional flow around the rectangular blunt body, we can now use Eq.17 for validation of the computed number densities along the stagnation streamline.

Figure 8 shows this comparison for two different Mach numbers, i.e. Mach 2 and Mach 6, obtained using a 64×64 velocity mesh. For different time instances, the build up of the number density upstream of the body can be seen as a result of the particles bouncing back from the solid surface. The comparison shows that for both Mach numbers and time instances considered the computed flow along the stagnation streamline is in excellent agreement with the analytical solution, confirming the capability of the quantum algorithm to accurately resolve flows of this type.

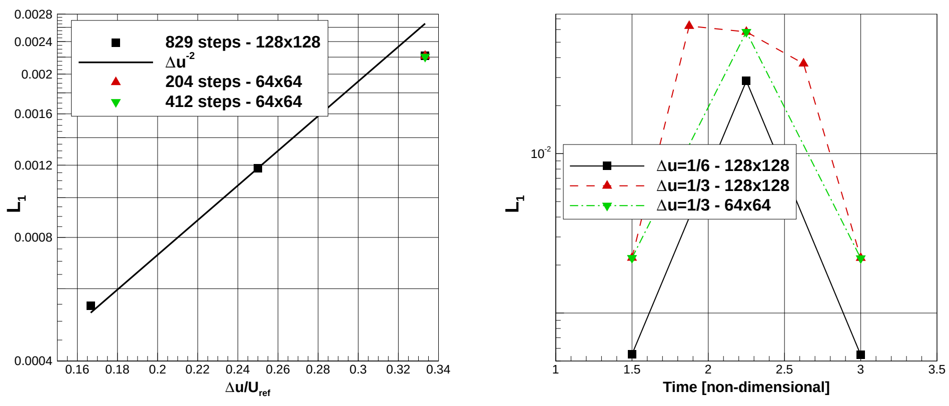
7.4. Error analysis

For Mach 6, a more detailed, quantitative convergence analysis is now performed. The reservoir-technique based method is first-order accurate in space. It is based on a discretization method assuming that function values are constant within each cell of the finite-volume mesh. For the discretized collisionless Boltzmann equation we effectively find a set of linear advection equations, i.e. a separate equation for each considered discrete velocity. The accuracy of the methods stems from the "CFL=1"-like exact propagation



(a) Mach 2 - 64×64 velocity-space (b) Mach 6 - 64×64 velocity-space

Figure 8: Comparison of computed gas density along stagnation streamline for Mach 2 and Mach 6. DVM results for 64×64 velocity mesh are compared with analytical solution



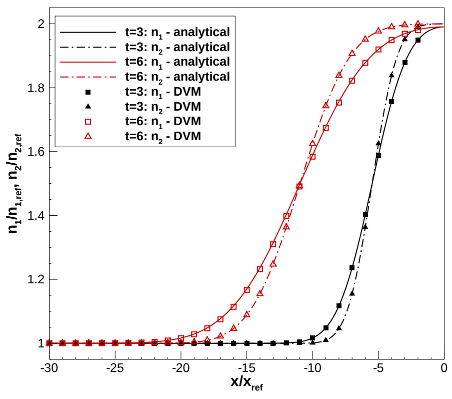
(a) L_1 as function of Δu (b) L_1 as function of evolved time

Figure 9: Comparison of computed gas density along stagnation streamline for Mach 6 with analytical solution. (a) L_1 norm has second-order dependency on velocity-mesh spacing, (b) Evolution of L_1 norm as function of elapsed (non-dimensional) time during simulation

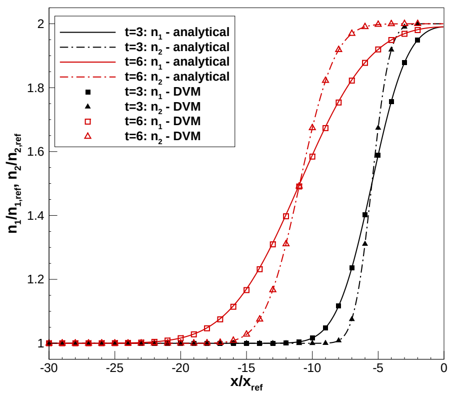
that is found for a first-order upwind method for the linear advection equation on a uniform mesh. At the end of a 'full cycle' (n_{cycle} steps and T_{cycle} non-dimensional period), as discussed in Section 4.1, an exact propagation of all discrete-velocity data has occurred. During a simulation, therefore, whenever the current time step is a multiple of the n_{cycle}, the only discretization error will be due to the discretization of the velocity-space. Specifically, for the discretized distribution functions that have propagated exactly, a discretization error will arise when moment integrations are used to obtain number densities, velocities, etc. In performing the error analysis for the reservoir-technique based method used here, the effect of obtaining output at other instances than multiples of n_{cycle} needs to be considered, as well as, the effect of the velocity-space discretization. The dependency of the error on when during a cycle output is created is mainly caused by the simplifications introduced in the reservoir technique to facilitate implementation as quantum algorithm. Therefore, it needs particular attention. The analysis of the effect of spatial mesh refinement (as opposed to refining velocity-space mesh) for the present time-accurate method for the collisionless Boltzmann equation differs markedly from more conventional mesh-refinement studies. A key factor is that the only 'characteristic' length scale occurring in the collisionless Boltzmann equation is the ratio of characteristic velocity (typically most-probable molecular speed at reference conditions) and the time-scale considered. In effect, if the mesh-spacing in each coordinate direction is halved, the simulated time needs to be doubled to achieve a solution that can be compared to the 'coarser-mesh' result in a mesh refinement study. First, the order of the truncation error in discretizing the velocity-space are studied for the case of 128×128 discrete velocities and a Mach 6 flow around the solid body considered previously. By increasing the velocity-domain size, three simulations were performed with different Δu (same spacing in $u-$ and $v-$ direction). Figure 9(a) shows the L_1 norm of the difference between the simulated density along the stagnation streamline and the analytical result. It is important to mention that the analytical result does not assume periodic boundary conditions at upstream- and down-stream ends of the domain. Therefore, only a limited amount of time can be considered, for which the flow disturbance created by the solid object has not yet reached these domain boundaries. The L_1 reduction for decreasing mesh spacing Δu indicates that the method has a second-order L_1 error with respect to Δu. The results shown are those after 829 time steps, i.e. 1/4 of a full cycle. For $\Delta u = 1/6$ and $\Delta u = 1/3$, this corresponds to an elapsed non-dimensional time of 1.5 and 3.0, respectively. The figure also shows the results at these time instances from a simulation with a 64×64 discrete-velocity mesh, with a non-dimensional $\Delta u = 1/3$. Non-dimensional times 1.5 and 3.0 are obtained after 204 (1/4 of n_{cycle}) and 412 (1/2 of n_{cycle}) time steps, respectively. Clearly, the obtained L_1-norms are almost identical than those found for the 128×128 discrete-velocity simulation with $\Delta u = 1/3$. This shows in that the present method, the discretization method used in the velocity space is the dominant effect in the error in the computed density for the considered time instances. To analyze the errors at different time instances (different fractions of n_{cycle}), the L_1 norm of the error of the predicted stagnation streamline density is plotted in Figure 9(b) as function of the elapsed non-dimensional time. As before, two cases with 128×128 discrete velocities are considered. For the case with $\Delta u = 1/3$, $T_{cycle} = 6$, while for $\Delta u = 1/6$, $T_{cycle} = 12$. Therefore, the shown results with $T = 1.5$ and $T = 3.0$ correspond to 1/4 cycle and 1/2 cycle for the $\Delta u = 1/3$ case. For the case with 64×64 discrete velocities ($\Delta u = 1/3$, $T_{cycle} = 6$), $T = 1.5$ and $T = 3.0$ also correspond to 1/4 cycle and 1/2 cycle. The results shown in Figure 9(b) clearly show that 'optimum' points within the cycle exist at which output has smallest errors. This can also be analyzed using the variation of the 'reservoirs' during a cycle, as defined in Section 4.1. At these 'optimum' time instances, the errors are dominated by velocity-space discretization, as shown previously. Detailed further analysis for the results at $T = 1.5$ and $T = 3.0$ shows that the difference in L_1 norm at these different times is approximately $1 \cdot 10^{-5}$ for the 128×128 discrete velocity cases, i.e. two orders of magnitude smaller than the L_1 norm itself. A similar analysis for the 64×64 discrete-velocity case showed a difference of approximately $3 \cdot 10^{-5}$ between the L_1 norms at $T = 1.5$ and $T = 3.0$, again two orders smaller than the L_1 norm itself. Clearly, for the selected time instances, the solution is close to that achieved by exact propagation of the discretized particle distribution function. For other fractions of n_{cycle}, the error can be orders of magnitude larger as demonstrated in Figure 9(b). In terms of refining the spatial mesh, this has important implications: if the mesh spacing is halved, the solution time needs to be doubled to achieve the same flow field on the finer mesh (as mentioned previously). If the solution output corresponds to optimum points within the cycle, then we see that for finer mesh, the flow field is better resolved, as expected. However, for the lattice points than coincide with those on the coarser mesh, a (nearly) identical solution is found, based on the behaviour shown in Figure 9(b).

7.5. Extension to binary gas mixture

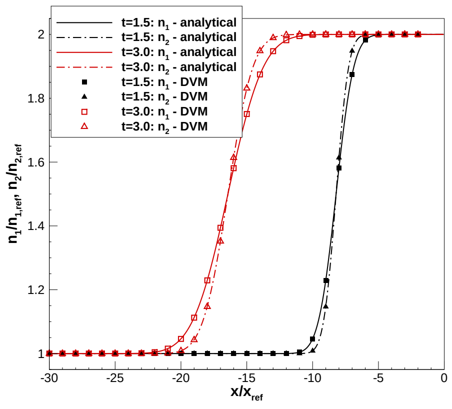
For a free-molecular flow of a binary gas mixture around the rectangular body, an analytical solution for the number densities along the stagnation streamline can be obtained as an extension of Bird's solution for a single-species gas. In this work, we use species 1 as the 'reference' , with m_1 the



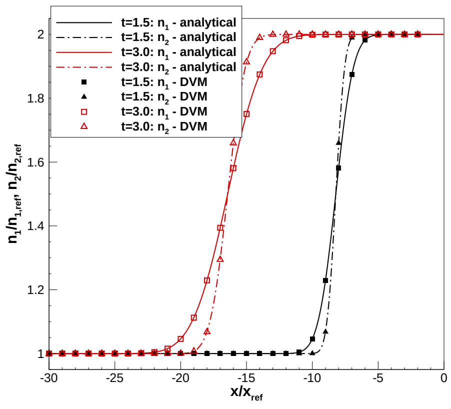
(a) Mach 2 - $m_2/m_1 = 2$



(b) Mach 2 - $m_2/m_1 = 4$



(c) Mach 6 - $m_2/m_1 = 2$

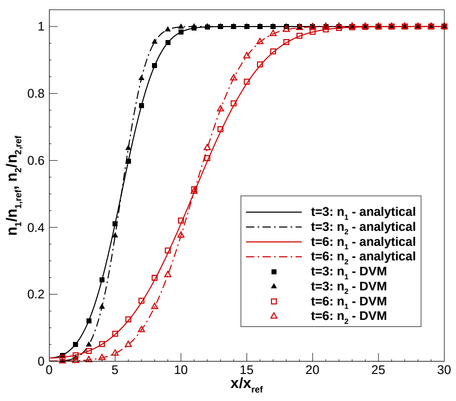


(c) Mach 6 - $m_2/m_1 = 4$

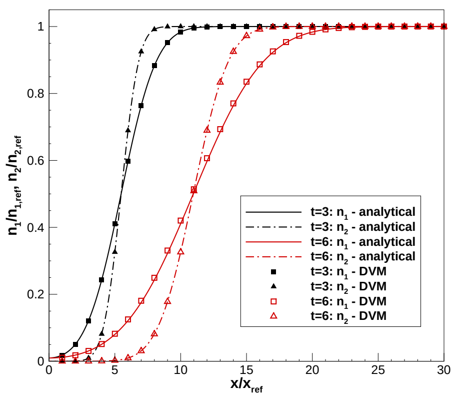
Figure 10: Non-dimensional species number densities along stagnation streamline for Mach 2 and Mach 6 flow around rectangular body. DVM results for 128×128 velocity mesh are compared with analytical solution. Effect of species mass ratio is shown for both Mach numbers.

molecular mass of species 1. Here we assume that species due is the heavier of the two species $m_2 > m_1$. The total number density of the ambient molecules $n_\infty = n_{1,\infty} + n_{2,\infty}$ is used as reference number density. Using specular-reflection boundary conditions as before, the number densities as function of x and t for the flow upstream of the body can then be written as,

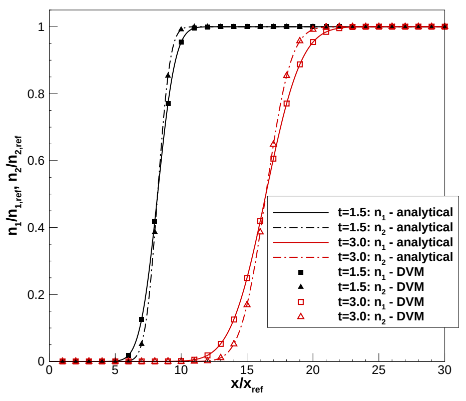
$$n_1(x,t) \quad = \quad n_{1,\infty} + \frac{n_{1,\infty}}{\sqrt{\pi T}} \int_{c_x=x/t}^{\infty} \left\{ \mathtt{exp}\Big[-\frac{(U_\infty - c_x)^2}{T} \Big] - \mathtt{exp}\Big[-\frac{(U_\infty + c_x)}{T} \Big] \right\} dc_x$$



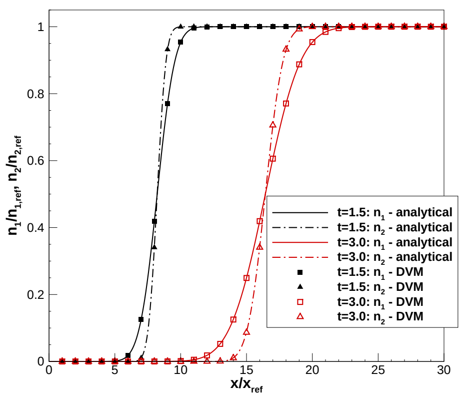
(a) Mach 2 - $m_2/m_1 = 2$



(b) Mach 2 - $m_2/m_1 = 4$



(c) Mach 6 - $m_2/m_1 = 2$



(c) Mach 6 - $m_2/m_1 = 4$

Figure 11: Non-dimensional species number densities along centre line in lee side of rectangular body for Mach 2 and Mach 6. DVM results for 128×128 velocity mesh are compared with analytical solution. Effect of species mass ratio is shown for both Mach numbers.

$$
\begin{aligned}
&= n_{a1}\Big[1 + \frac{1}{2}\Big\{\texttt{erf}\Big(\frac{x/t + U_\infty}{\sqrt{T}}\Big) - \texttt{erf}\Big(\frac{x/t - U_\infty}{\sqrt{T}}\Big)\Big\}\Big] \qquad (18)\\
n_2(x,t) &= n_{2,\infty} + \frac{n_{2,\infty}\sqrt{m_2/m_1}}{\sqrt{\pi T}} \int_{c_x = x/t}^{\infty} \Big\{\texttt{exp}\Big[-\frac{(U_\infty - c_x)^2}{T}\Big(\frac{m_2}{m_1}\Big)\Big] \\
&\quad -\texttt{exp}\Big[-\frac{(U_\infty + c_x)}{T}\Big(\frac{m_2}{m_1}\Big)\Big]\Big\} dc_x \\
&= n_{2,\infty}\Big[1 + \frac{1}{2}\Big\{\texttt{erf}\Big(\frac{x/t + U_\infty}{\sqrt{T}}\sqrt{\frac{m_2}{m_1}}\Big) - \texttt{erf}\Big(\frac{x/t - U_\infty}{\sqrt{T}}\sqrt{\frac{m_2}{m_1}}\Big)\Big\}\Big] \qquad (19)
\end{aligned}
$$

A similar analytical solution can also be obtain for the region behind the rectangular body. In this case, as a result of the solid body blocking particles from the upstream direction, a growing region with a very low-density forms.

Figure 10 shows the non-dimensional species number densities along stagnation streamline for a Mach 2 and a Mach 6 flow around the rectangular body. With species 1 used as reference in the scaling of the equations, the increased molecular mass of species 2 creates an increased challenge in representing the distribution function in velocity space (i.e. as a result of the more 'pointy' equilibrium distribution function in the initial solution). Therefore, DVM results for a 128×128 velocity mesh are compared with the analytical solution. For both mass ratios $m_2/m_1 = 2$ and $m_2/m_1 = 4$, excellent agreement can be observed. For the flow field behind the rectangular body, Figure 11 shows the non-dimensional species number densities along the centerline for the Mach 2 and Mach 6 flow conditions. It can be seen that the numerical results predict the formation of the low-density region accurately, with a excellent agreement with the analytical solutions obtained for this flow region, demonstrating the capability of the quantum algorithm to resolve the considered binary-mixture flows accurately.

7.6. Free molecular flow escaping from rectangular container

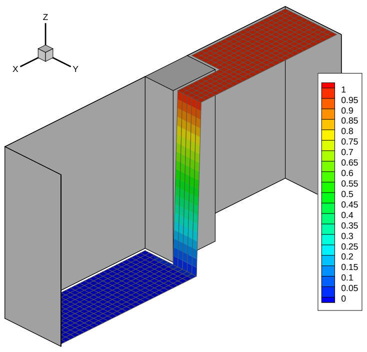
As a final test case, the time-dependent flow out of a rectangular container is considered. A 64×16 uniformly spaced mesh was used. Figure 12(a) shows the initial condition, where a uniform gas density is assumed within the container as well as in the rectangular throat, while the rest of the domain is initialized with a vacuum. One half of the overall domain is considered, with a symmetry boundary condition applied on the 'open' right-hand side of shown domain. The initial density and temperature are used as reference values, so that initially gas density $\rho = 1$ and $T = 1$. The velocity-space discretization employs a uniform 128×128 mesh, with boundaries $[-4, 4]^2$ scaled with most probable speed at reference temperature. The non-dimensional time $T_{cycle} = 32$ for this integration, i.e. within one cycle a particle travelling at the most probable speed will cover $32\Delta x$. Figure 12 shows the flow developing between $T = 0$ and $t = T_{cycle}/4$, while the development up to $t = T_{cycle}/2$ is shown in Figure 13. For all time instances, the non-dimensional density and $u-$velocity are shown. As expected, the flow evacuates the container, forming a jet coming out of the throat. Also, a rarefaction wave runs in the opposite direction into the reservoir. Because of the collisionless nature of the gas, no shock waves are formed. Furthermore, it can be seen that for the 128×128 velocity-space mesh, the output at timesteps not coinciding with the end of a cycle does not lead to significant oscillations in the gas density and velocity field. Therefore, the simplification made to facilitate implementation on a quantum computer is clearly acceptable.

8. Conclusions and future work

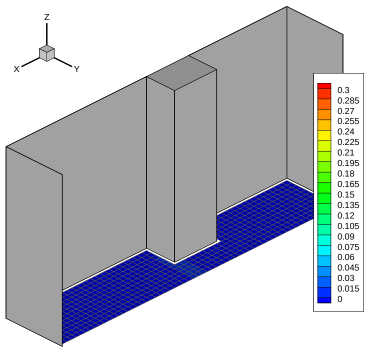
The development of a quantum algorithm for the collisionless Boltzmann equation was presented. A time-integration method based on the reservoir technique was described in detail for one- and two-dimensional flows. Design choices regarding the data structure were discussed along with quantum circuits implementing the convections steps in both $x-$ and $y-$directions on a uniform mesh. For the case of specular-reflection boundary conditions, the quantum circuit implementation was detailed next. The extension to simulations involving multiple species was discussed, highlighting a key benefit of the quantum algorithm in this context: no additional quantum gate operations were found to be required when adding more species. Validation of the developed algorithm was then presented for the free-molecular flow at different Mach numbers around a rectangular body showing excellent agreement with exact solutions applicable to the stagnation streamlines for this problem. As an example for a relevant time-dependent flow, the flow out of a rectangular domain into vacuum is considered. It was shown that despite the simplifications made to the reservoir time-integration technique, results with good accuracy and largely free of oscillations could be achieved. In future work, the aspects of initializing the quantum state vector and obtaining output efficiently need to be investigated further and in more detail. Furthermore, the application to more complex flow problems will be considered, along with a more detailed investigation of applications to design problems. The challenging problem of extending the current method to kinetic problem with inter-particle collisions forms a further important line of future work.

Acknowledgments

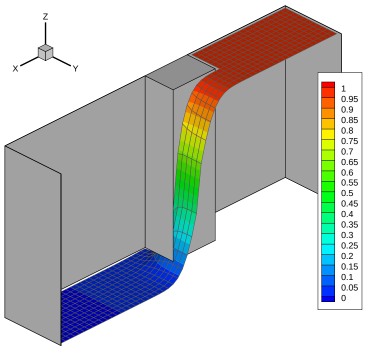
The authors would like to acknowledge the financial support provided by the University of Glasgow.



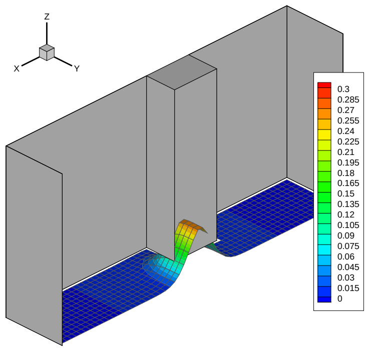
(a) ρ at $t = 0$



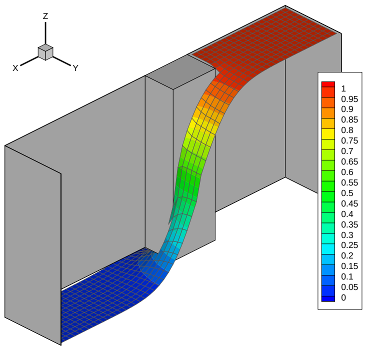
(b) u velocity at $t = 0$



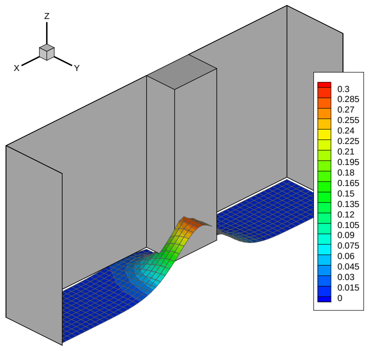
(c) ρ at $t = T_{cycle}/8$



(d) u velocity at $t = T_{cycle}/8$

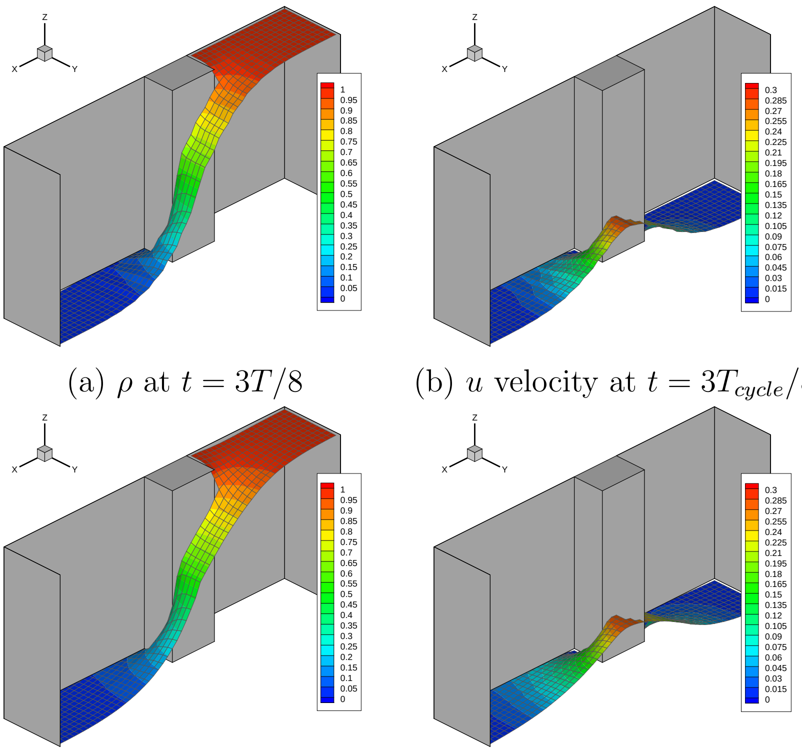


(e) ρ at $t = T_{cycle}/4$



(f) u velocity at $t = T_{cycle}/4$

Figure 12: Formation of jet out of rectangular container in two-dimensional domain simulated with quantum algorithm for collisionless Boltzmann equation. A 64×16 uniformly spaced mesh was used. Discrete-velocity method used 128×128 discrete-velocities. Initial gas region has ρ and T at 1.0. Density is non-dimensionalized with initial density and speed with most probably molecular speed at initial temperature.



(a) ρ at $t = 3T/8$ (b) u velocity at $t = 3T_{cycle}/8$

(c) ρ at $t = T/2$ (d) u velocity at $t = T_{cycle}/2$

Figure 13: Formation of jet out of rectangular container in two-dimensional domain simulated with quantum algorithm for collisionless Boltzmann equation. A 64×16 uniformly spaced mesh was used. Discrete-velocity method used 128×128 discrete-velocities. Initial gas region has ρ and T at 1.0. Density is non-dimensionalized with initial density and speed with most probably molecular speed at initial temperature.

References

[1] M. Nielsen, I. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press, 2010 (2010).

[2] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124–134 (Nov 1994). doi:10.1109/SFCS.1994.365700.

[3] L. Grover, Quantum mechanics helps in searching for a needle

in a haystack, Phys. Rev. Lett. 79 (1997) 325–328 (Jul 1997). doi:10.1103/PhysRevLett.79.325.

[4] S. Sinha, P. Russer, Quantum computing algorithm for electromagnetic field simulation, Quantum Information Processing 9 (3) (2010) 385–404 (Jun 2010). doi:10.1007/s11128-009-0133-x.

[5] A. Scherer, B. Valiron, S. Mau, S. Alexander, E. vanden Berg, T. Chapuran, Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2D target, Quantum Information Processing 16 (3) (2017) 60 (Jan 2017). doi:10.1007/s11128-016-1495-5.

[6] G. Xu, A. Daley, P. Givi, R. Somma, Turbulent Mixing Simulation via a Quantum Algorithm, AIAA Journal 56 (2) (2018) 687–699 (2018). doi:10.2514/1.J055896.

[7] R. Steijl, G. Barakos, Parallel evaluation of quantum algorithms for computational fluid dynamics, Computers & Fluids 173 (2018) 22–28 (2018). doi:10.1016/j.compfluid.2018.03.080.

[8] A. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. 103 (2009) 150502 (2009). doi:10.1103/PhysRevLett.103.150502.

[9] S. Jordan, Fast quantum algorithm for numerical gradient estimation, Phys. Rev. Lett. 95 (2005) 050501 (2005). doi:10.1103/PhysRevLett.95.050501.

[10] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, S. Kais, Quantum Algorithm and Circuit Design solving the Poisson Equation, New Journal of Physics 15 (2013) 013021 (2013). doi:10.1088/1367-2630/15/1/013021.

[11] P. Costa, S. Jordan, A. Ostrander, Quantum algorithm for simulating the wave equation, Phys. Rev. A 99 (2019) 012323 (2019). doi:10.1103/PhysRevA.99.012323.

[12] R. Steijl, G. Barakos, Coupled Navier-Stokes-Molecular Dynamics simulations Using a Multi-physics Flow Simulation Framework, International Journal for Numerical Methods in Fluids 62 (2010) 1081–1106 (2010). doi:10.1002/fld.2053.

[13] R. Steijl, G. Barakos, Coupled Navier-Stokes/Molecular Dynamics Simulations in Nonperiodic Domains Based on Particle Forcing, International Journal for Numerical Methods in Fluids 69 (2012) 1326–1349 (2012). doi:10.1002/fld.2641.

[14] J. Yepez, Quantum lattice-gas model for computational fluid dynamics, Phys. Rev. E 63 (2001) 046702 (2001). doi:10.1103/PhysRevE.63.046702.

[15] G. Berman, A. Ezhov, D. Kamenev, J. Yepez, Simulation of the diffusion equation on a type-II quantum computer, Phys. Rev. A 66 (2002) 012310 (2002). doi:10.1103/PhysRevA.66.012310.

[16] M. Micci, J. Yepez, Measurement-based quantum lattice gas model of fluid dynamics in 2+1 dimensions, Phys. Rev. E 92 (2015) 033302 (2015). doi:10.1103/PhysRevE.92.033302.

[17] A. Childs, Universal computation by quantum walk, Phys. Rev. Lett. 102 (2009) 180501 (2009). doi:10.1103/PhysRevLett.102.180501.

[18] I. Karafyllidis, Quantum walks on graphene nanoribbons using quantum gates as coins, Journal of Computational Science 11 (2015) 326 – 330 (2015). doi:10.1016/j.jocs.2015.05.006.

[19] S. Venegas-Andraca, Quantum walks: a comprehensive review, Quantum Information Processing 11 (5) (2012) 1015–1106 (2012). doi:10.1007/s11128-012-0432-5.

[20] B. Douglas, J. Wang, Efficient quantum circuit implementation of quantum walks, Phys. Rev. A 79 (2009) 052335 (2009). doi:10.1103/PhysRevA.79.052335.

[21] F. Fillion-Gourdeau, S. MacLean, R. Laflamme, Algorithm for the solution of the dirac equation on digital quantum computers, Phys. Rev. A 95 (2017) 042343 (Apr 2017). doi:10.1103/PhysRevA.95.042343.

[22] F. Fillion-Gourdeau, E. Lorin, Simple digital quantum algorithm for symmetric first-order linear hyperbolic systems, Numerical Algorithms (Dec 2018). doi:10.1007/s11075-018-0639-3.

[23] N. Crouseilles, M. Mehrenberger, E. Sonnendrücker, Conservative semi-Lagrangian schemes for Vlasov equations, Journal of Computational Physics 229 (2010) 1927 – 1953 (2010). doi:10.1016/j.jcp.2009.11.007.

[24] Y. Güclü, W. Hitchon, A high order cell-centered semi-Lagrangian scheme for multi-dimensional kinetic simulations of neutral gas flows, Journal of Computational Physics 231 (2012) 3289 – 3316 (2012). doi:10.1016/j.jcp.2012.01.008.

[25] G. Dimarco, R. Loubere, Towards an ultra efficient kinetic scheme. Part I: Basics on the bgk equation, Journal of Computational Physics 255 (2013) 680 – 698 (2013). doi:10.1016/j.jcp.2012.10.058.

[26] G. Dimarco, R. Loubere, Towards an ultra efficient kinetic scheme. Part II: The high order case, Journal of Computational Physics 255 (2013) 699 – 719 (2013). doi:10.1016/j.jcp.2013.07.017.

[27] I. Karafyllidis, Simulation of entanglement generation and variation in quantum computation, Journal of Computational Physics 200 (1) (2004) 383 – 397 (2004). doi:10.1016/j.jcp.2004.04.008.

[28] C. Zalka, Simulating quantum systems on a quantum computer, Proc. R. Soc. Lond. A. 454 (1998) 313–322 (1998). doi:10.1098/rspa.1998.0162.

[29] I. Georgescu, S. Ashhab, F. Nori, Quantum simulation, Rev. Mod. Phys. 86 (2014) 153–185 (Mar 2014). doi:10.1103/RevModPhys.86.153.

[30] L. Grover, T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions, Arxiv.org (2002) 1–2 (2002). URL `http://www.arxiv.org/abs/quant-ph/0208112`

[31] A. Soklakov, R. Schack, Efficient state preparation for a register of quantum bits, Phys. Rev. A 73 (2006) 012307 (Jan 2006). doi:10.1103/PhysRevA.73.012307.
URL `https://link.aps.org/doi/10.1103/PhysRevA.73.012307`

[32] G. Ortiz, J. E. Gubernatis, E. Knill, R. Laflamme, Quantum algorithms for fermionic simulations, Phys. Rev. A 64 (2001) 022319 (Jul 2001). doi:10.1103/PhysRevA.64.022319.
URL `https://link.aps.org/doi/10.1103/PhysRevA.64.022319`

[33] R. Somma, G. Ortiz, J. Gubernatis, E. Knill, R. Laflamme, Simulating physical phenomena by quantum networks, Phys. Rev. A 65 (2002) 042323 (Apr 2002). doi:10.1103/PhysRevA.65.042323.

[34] G. Brassard, P. Hoyer, M. Mosca, Quantum amplitude amplification and estimation, in: Quantum Computation and Information, Contemporary Mathematics, Vol. 305, 2002, pp. 53–74 (2002). doi:10.1090/conm/305/05215.

[35] W. Vincenti, C. Kruger, Introduction to physical gas dynamics, 2nd Edition, John Wiley and Sons, New York, 1967 (1967).

[36] C. Cercignani, The Boltzmann Equations and its Applications, 2nd Edition, Springer-Verlag, New York, 1987 (1987).

[37] G. Bird, One-dimensional Compression of a Collisionless Gas, Journal of Fluid Mechanics 21 (1) (1965) 183–191 (1965). doi:10.1017/S0022112065000125.

[38] F. Alouges, F. De Vuyst, G. Le Coq, E. Lorin, The reservoir technique: a way to make godunov-type schemes zero or very low diffuse. Application to ColellaGlaz solver, European Journal of Mechanics - B/Fluids 27 (6) (2008) 643 – 664 (2008). doi:10.1016/j.euromechflu.2008.01.001.

[39] R. Narasimha, Collisionless Expansion of a Gas into Vacuum, Journal of Fluid Mechanics 12 (2) (1962) 294–308 (1962). doi:10.1017/S0022112062000208.