Chapter

Quantum Algorithms for Fluid Simulations

René Steijl

Abstract

This chapter describes results of a recent investigation aiming to assess the potential of quantum computing and suitably designed algorithms for future computational fluid dynamics applications. For quantum computers becoming available in the near future, it can be expected that applications of quantum computing follow the quantum coprocessor model, where selected parts of the computational task for which efficient quantum algorithms exist are executed on the quantum hardware. For example, in computational fluid dynamics algorithm, this hybrid quantum/classical approach is discussed, and in particular it is shown how the approximate quantum Fourier transform (AQFT) can be used in the Poisson solvers of the considered method for the incompressible-flow Navier-Stokes equations. The analysis shows that despite the inevitable errors introduced by applying AQFT, the method produces meaningful results for three-dimensional example problems. A second example of a quantum algorithm for flow simulations is then described. This method based on kinetic modeling of the flow was developed to reduce the information transfer between quantum and classical hardware in the quantum coprocessor model. It is shown that this quantum algorithm can be executed fully on quantum hardware during a simulation. The conclusion summarizes further challenges for algorithm developments and future work.

Keywords: quantum algorithms, computational fluid dynamics, rarefied flows, kinetic modeling

1. Introduction

In recent years, the field of quantum computing [1] has developed into an active and diverse field of research, and significant progress has been made in a number of important areas. For a relatively small number of applications, quantum algorithms have been developed that provide a significant speedup relative to classical methods. Shor's algorithm for factoring composite integers and Grover's algorithm for quantum search were key developments in establishing quantum computing. More recently, significant progress has been made in the area of quantum chemistry and quantum physics. Beyond those two fields, only recently have quantum computing applications appeared in other areas of science and engineering, e.g., work in computational electromagnetics [2, 3], mixing in turbulent flow [4], and computational fluid dynamics [5]. More general 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 [6] and Poisson equation [7].

In recent years significant progress has been made in designing and constructing quantum computers. Currently available quantum computers are relatively small-scale and have become known as noisy intermediate-scale quantum (NISQ) computers. These machines have a limited number of qubits (expected to increase to 50–100 in coming years), a limited connectivity between these qubits, a small set of available quantum gates, and typically very little or no quantum error correction.

This chapter describes results of a recent investigation aiming to assess the potential of quantum computing and suitably designed algorithms for future computational fluid dynamics application, particularly for NISQ-type quantum hardware. In this work, the quantum circuit model is used for a "universal" or "digital" quantum computer, i.e., work on adiabatic quantum computing is not considered here. In the absence of the required quantum hardware, large-scale parallel simulations on parallel classical computers are required in developing such algorithms. In this work the recently developed quantum simulator [5] included in the MΦC multi-physics CFD framework is used [8, 9].

In the near future, the most likely scenario for the introduction of quantum computing hardware is through the quantum coprocessor model, i.e., where a quantum processing unit (QPU) is loosely coupled to a classical computer with one or more CPUs [10]. 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 application of this hybrid quantum/classical approach, the quantum processor acts like a coprocessor 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 in quantum circuits. 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 are needed to obtain classical information with acceptable levels of noise and uncertainty. It is important to recognize that, since initializing a particular quantum state in quantum computer can be a significant challenge, this hybrid approach can only be expected to lead to significant computational speedups 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 author 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 [5]. In this algorithm, 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. In this chapter, this algorithm and its application to example flow problems are investigated further. Specifically, the effect of applying an approximate QFT instead of the full QFT is analyzed for different levels of approximation or truncating of rotation gates in the quantum circuit implementation.

The second part of this chapter describes a more recent investigation into the development of quantum algorithms relevant for computational aerodynamics based on modeling at the kinetic level. The key innovation in these developments is the design targeting execution of the algorithm fully on the quantum processor. In particular, at the start of the simulation, multiple quantum states in the quantum processor would be initialized. Then, the quantum algorithms would perform a series of unitary transformations. Only at the end of the simulation would

measurements be performed to extract the required classical. A key question this study aims to answer is for which applications this approach is feasible.

This chapter is organized as follows. Section 2 describes key principles of quantum computing relevant to the quantum algorithms for fluid simulations described here. Section 3 describes the hybrid quantum/classical implementation of the vortex-in-cell method along with a number of example applications. The quantum discrete-velocity algorithm for kinetic flow modeling is described in Section 4. Conclusions and future research directions are presented in Section 5.

2. Elements of quantum computing relevant in current work

The fundamental unit of quantum computation is the qubit [1]. 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 [1]. A collection of nq qubits in a coherent state is termed a quantum register of size nq here. Its quantum state is defined by the wave function $|\Psi>$ resulting from the tensor product of the quantum states of each qubit in the coherent register. The superposition in this coherent register then creates 2^{nq} different states that can be found upon measurement of the quantum state. In simulating this quantum state on a classical computer, a storage space of 2^{nq} complex numbers is required.

In the present work, the quantum circuit model of quantum computing is used. In this case, the unitary operations 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 that performs a fixed unitary operation on selected qubits in a fixed period of time. Written in a matrix form, unitary means that the determinant of the transformation equals one.

2.1 Mapping a computational problem onto the quantum state vector

The quantum state $|\Psi>$ for a register with nq coherent qubits is represented by a Hilbert space of dimension 2^{nq}. In a quantum computer, different possibilities exist for the physical implementation of qubits, e.g., the "spin" of an electron (with the two possible states being "spin-up" and "spin-down") or the plane of polarization of linearly polarized photon has been used. The discrete energy levels in an atom excited by laser pulses present an alternative to electron-spin and photon-based qubit implementations.

We will now describe how this quantum state can be used to represent the storage space required for the computational problems of interest here, representing discretized partially differential equations. As a first step, consider 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 degrees of freedom in the quantum state vector.

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 $nq = \log_2 N$, 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 degrees of freedom. This superposition-based principle for

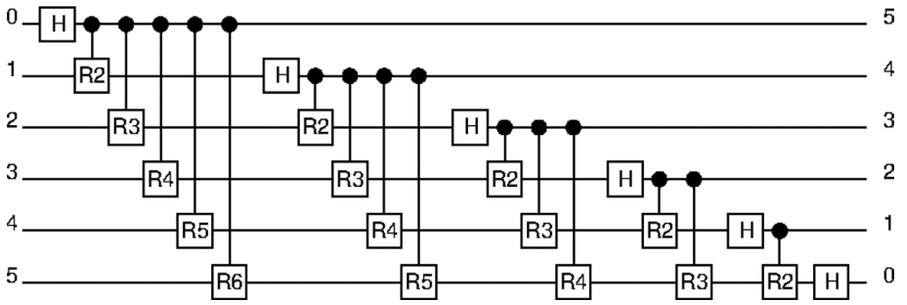

Figure 1.
Quantum circuit for QFT on six-qubit register.

storage of discrete data can be extended to multidimensional problems as well. For example, 24 qubits suffice to store a single discretized function on a 256^3 regular mesh. In application in which multiple variables are to be stored in each mesh point, as in the discrete-velocity method discussed in the second part of this chapter, we add further qubits to the quantum register. Specifically, for each qubit added in the coherent register, the number of degrees of freedom is doubled. Once a mapping of the considered computational problem onto the quantum state vector has been designed, calculations are then performed through application of quantum gates as in the quantum circuit model.

2.2 Approximate quantum Fourier transform (AQFT)

The ability to implement the quantum Fourier transform (QFT) efficiently on a quantum computer is of paramount importance for many quantum algorithms. **Figure 1** shows the "standard" quantum circuit implementation of the QFT for an example register with 6 qubits. In **Figure 1**, "H" represents the one-qubit Hadamard gate, and the "Rk" gates are controlled rotation gates over an angle defined by index "k," i.e., $2\pi/2^k$. In this circuit and all subsequent quantum circuits shown in this chapter, the qubit register is represented vertically, with the leftmost qubit in the register at the top. The horizontal direction determines the sequence of gates that are applied to the quantum state. In the QFT example shown, it can be seen that the qubit indices have been reversed in the output state (right-hand side) relative to the input, to represent that this standard QFT circuit returns the discrete Fourier transform in bit-reversed ordering.

A particular challenge is presented by the controlled rotation gates particularly those involving small angles. The QFT can be implemented approximately by removing all rotation gates with angles smaller than a certain threshold value, resulting in the approximate QFT (AQFT). In particular for fault-tolerant implementations, this is desirable as it greatly reduces the gate count. In the following, we define the approximation or "band-limiting" in the AQFT as follows. The rotation gates are eliminated above a limit value "k," i.e., for an angle smaller than $2\pi/2^k$, the rotation gate is not included in the quantum circuit.

3. Hybrid quantum/classical vortex-in-cell method

The vortex-in-cell method is a well-studied hybrid particle-mesh method for incompressible flows and is particularly well suited for flows in regular domains such that efficient Poisson solvers can be used. In the present work, the Fourier

DOI: http://dx.doi.org/10.5772/intechopen.86685

analysis approach to solving the problem in a fully periodic domain is used, using the QFT for the required discrete Fourier transforms.

The vortex-in-cell (VIC) method solves the incompressible-flow Navier-Stokes equations, transformed into the Helmholtz equations for vorticity evolution [5]:

$$\frac{\partial \vec{\omega}}{\partial t} + \vec{u}\frac{\partial \vec{\omega}}{\partial \vec{x}} = (\vec{\omega}.\nabla)\vec{u} + \nu\Delta\vec{\omega}; \vec{\omega} = \nabla x\ \vec{u} \tag{1}$$

In simulations using the VIC, the flow evolves through a (large) number of time steps. During each of these time steps, the velocity field is recomputed using solutions of three Poisson problems for stream function $\vec{A}$:

$$\Delta\vec{A} = -\vec{\omega}; \vec{u} = \nabla x\ \vec{A} \tag{2}$$

This part of the VIC is of particular interest here, as it represents the part that would be performed by the quantum processor in the quantum coprocessor model.

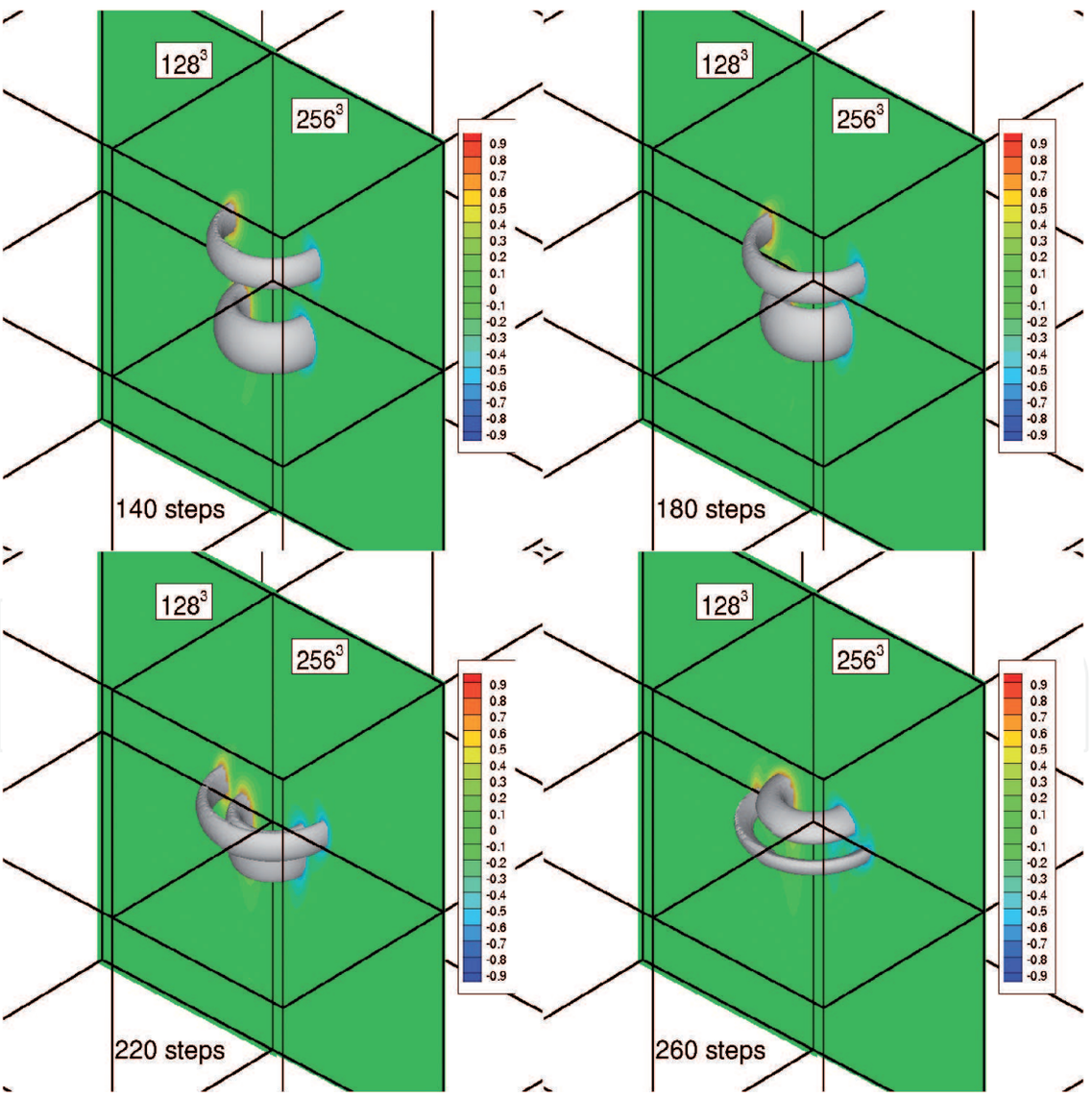

Figure 2.
Vortex-in-cell simulation of leapfrogging vortex rings. Effect of mesh refinement is shown. "Noiseless" simulation using full QFT.

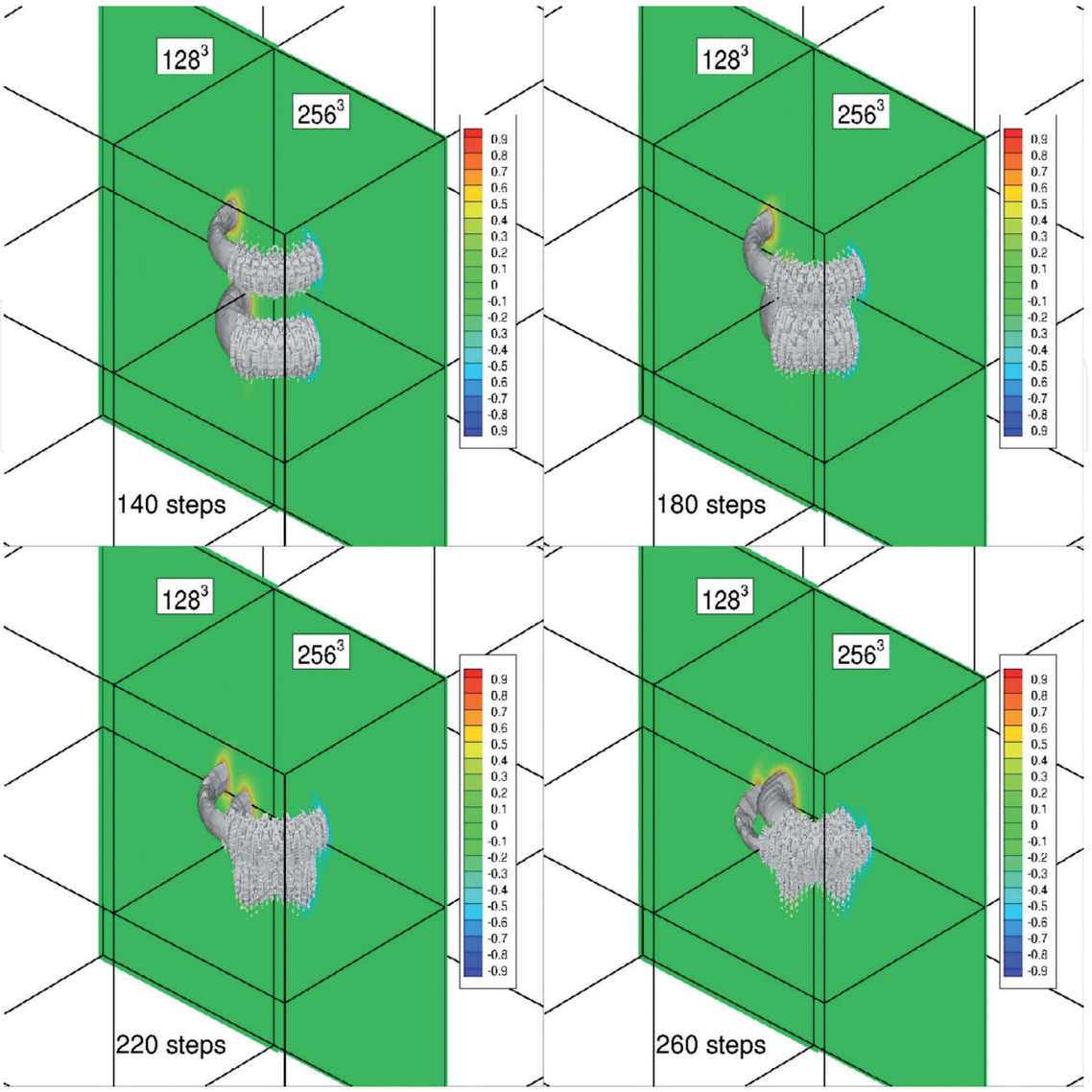

Figure 3.
Vortex-in-cell simulation of leapfrogging vortex rings. For two different mesh resolutions, the effect of applying AQFT is shown ("k" limit is 5).

Figure 2 shows an example of a VIC simulation of two leapfrogging vortex rings, i.e., flow structures of fundamental importance in fluid mechanics. The lower vortex ring is stronger than the ring above it, and it will therefore convect upward faster, leading to the interaction of the vortex rings as shown. The iso-surface represents vorticity strength, i.e., a direct indicator of the "strength" of the considered vortex. Results are compared for two different meshes, 128^3 and 256^3, to highlight the dependency of the solution on the chosen mesh size. Also, in the shown simulation, no quantum errors were simulated, and the full QFT was used. If we now replace the QFT with the AQFT, the results shown in **Figures 3** and **4** are obtained. In **Figure 3**, the "k" limit in the QFT is set to five for both meshes, showing that for the finer mesh this leads to unacceptable errors, while the coarser-mesh simulation still produces worthwhile results. If the "k" limit for the finer mesh is increased from 5 to 6, i.e., more controlled rotation gates are included in the AQFT circuit, the simulation on the finer mesh can also be made to produce similarly useful results. These example results show what level of approximation in the QFT is tolerable for application of the VIC method. For other QFT-based CFD solvers, a similar sensitivity study would need to be conducted.

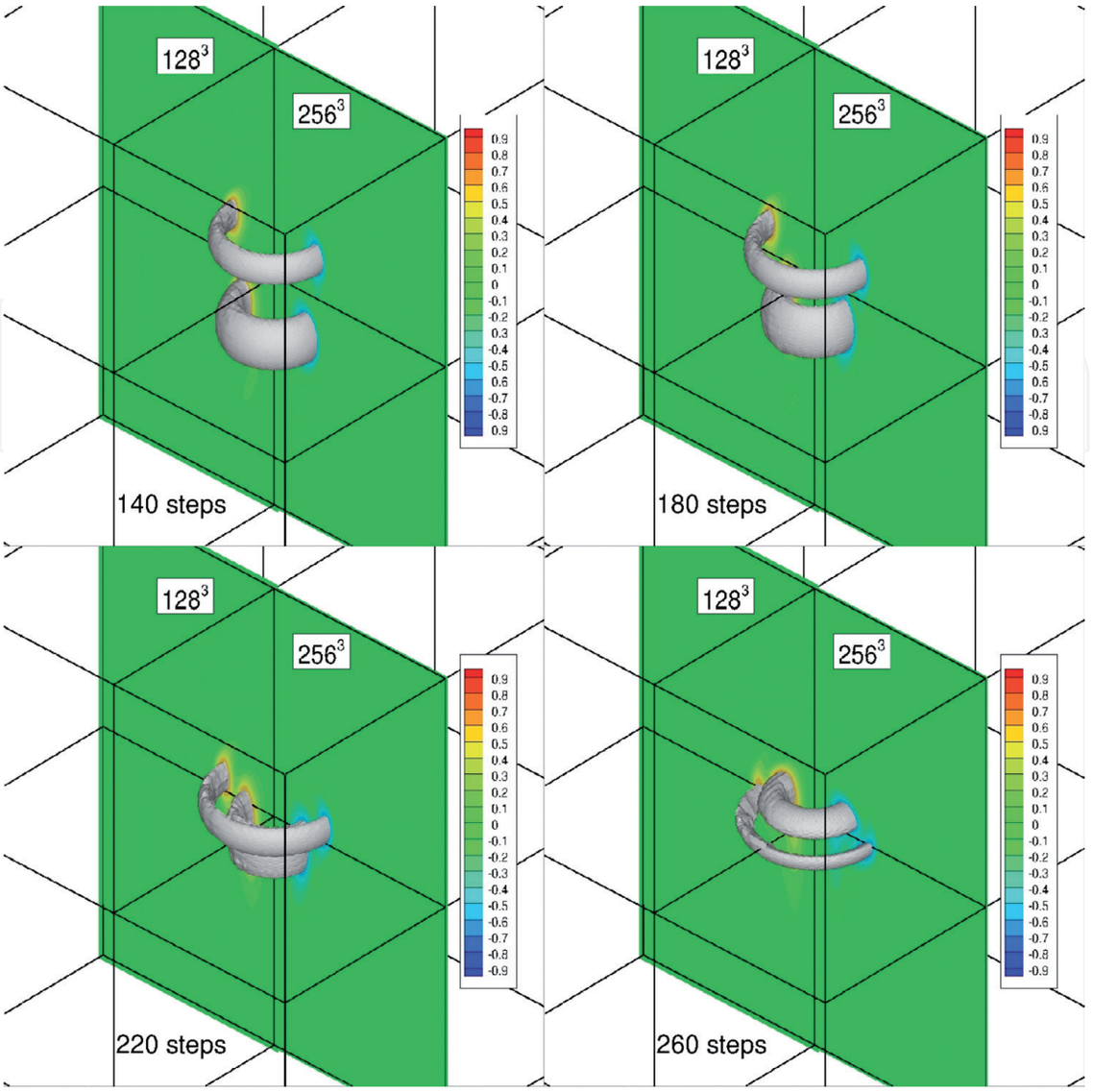

Figure 4.
Vortex-in-cell simulation of leapfrogging vortex rings. For 1283 mesh, AQFT ("k" limit 5) is used. For 2563 mesh, "k" limit in AQFT is 6.

4. Quantum algorithm for discrete-velocity method

In computational fluid dynamics, the most widely used methods involve solving the Navier-Stokes equations for a continuum fluid, i.e., where fluid density, velocity components, and energy in each location in the computational domain are to be found from conservation equations. The vortex-in-cell method used in the previous section employs the Navier-Stokes equations in a transformed form involving vorticity rather than velocity, but importantly it still uses the continuum flow assumption.

An alternative approach to the Navier-Stokes-based modeling is a description of the flow at a more detailed level, i.e., at the kinetic level [11]. Instead of governing equations for mass, momentum, and energy conservation, the flow is now described by the Boltzmann equation governing a particle distribution function in state space (or 3D velocity space for a 3D monatomic gas flow) for each location in the considered domain [10].

For a monatomic gas, the distribution function $f(\vec{x}, \vec{c}; t)$ with $\vec{x}$ defining the coordinates in (physical) space and $\vec{c}$ the molecular velocity (defined in "velocity space") is governed by the Boltzmann equation:

$$\frac{\partial f}{\partial t} + \vec{c}\frac{\partial f}{\partial \vec{x}} = Q(f, f) \tag{3}$$

The distribution function defines for each point $\vec{x}$ the likelihood that molecules have velocity $\vec{c}$. On the right-hand side of the equation, the collision term $Q(f,f)$ represents the effect of collisions between particles [12]. For each point in space, we can take moments of the distribution function to obtain the local fluid density, velocity components, and energy.

The key advantage of this approach is that non-continuum flows, i.e., flows for which the density is so low that we cannot assume it to act as a continuum, can also be modeled. However, the main problem is the large computational cost when using a direct discretization approach due to the high dimensionality, i.e., for a 3D flow problem, we have a six-dimensional solution space (or seven when including time).

A further main challenge is the cost of evaluating the collision term. For the free-molecular flows considered here, the collision term can be discarded, and we will use the **collisionless Boltzmann equation** instead.

In the discrete-velocity method used here, the velocity space is discretized using a uniformly spaced Cartesian mesh. A maximum molecular velocity magnitude is defined, c_{max}, so that the left and right domain boundaries in velocity space are at $-c_{max}$ and $+c_{max}$, respectively, with a uniform spacing Δc separating each point in velocity space. These limits are problem dependent.

The discrete-velocity approach used here has a number of characteristics facilitating a quantum implementation:

1. A uniformly spaced Cartesian mesh is used for the spatial discretization as well as for the discretization of the velocity space.

2. In case solid objects are present in the computational domain, these are rectangular, and its edges align with the mesh lines in the mesh. Specifically, solid bodies can be defined by "tagging" selected groups of cells in the mesh.

3. A constant velocity-space discretization is used in each point in space, i.e., the velocity-space boundaries defined earlier as well as the number of discrete velocities are identical in each cell.

4. The convection part of the Boltzmann equation (i.e., the second term on the left-hand side in the shown equation) along with gas-solid interactions determines the time evolution of the distribution function in the absence of interparticle collisions.

5. The time-integration method used here is based on the reservoir technique [13], such that during the time integration the convection step always exactly involves the distribution function defined in a cell of computational mesh to move to a cell that is a nearest neighbor. This is commonly referred to as "streaming" of data.

In the examples shown, problems are restricted to two-dimensional flows. For this case, the collisionless Boltzmann equation originally defined for 3D can be reduced to two kinetic equations for two reduced distribution functions:

$$\frac{\partial f}{\partial t} + \vec{c}\frac{\partial f}{\partial \vec{x}} = 0; \frac{\partial g}{\partial t} + \vec{c}\frac{\partial g}{\partial \vec{x}} = 0 \tag{4}$$

where the velocity and coordinate vectors are now defined in 2D.

Inspired by the previous work on the Dirac equation [14], an efficient quantum circuit implementation of the streaming operations in x- and y-direction on a Cartesian 2D mesh can be created as shown in **Figure 5**. The example shows how a 12-qubit register is used for a discretized function with 6 qubits defining the indices of 64 grid points in x- and y-coordinate directions. The circuits with the filled circles define streaming to "right" neighbors, i.e., when each control qubits has the |1> state, the target qubit gets negated ("X" symbol used here). Similarly, the left streaming operation employs multiple-control NOT gates with target qubits being negated when each control qubit is in state |0>.

For the streaming operations in quantum discrete-velocity method, further extensions are needed. First, the quantum register needs to be extended relative to that shown in **Figure 5** to account for the storage of two discretized reduced distributions defined in the discretized velocity space. The additional distribution function is accounted for using an additional qubit termed "g" in the following quantum circuit diagrams. The number of additional qubits needed for the discrete-velocity mesh clearly depends on the number of discrete velocities used. For example, for a 16 × 16 discrete-velocity mesh, we add 8 qubits (four for each direction in state space). The qubits are denoted by the "u0,"..., "v0,"... qubits in the quantum circuits. Finally, to account for solid objects, we use an additional qubit ("BC" in the diagrams) set to |1> to denote a cell within fluid and |0> for a cell within a solid. For a 64 × 64 Cartesian mesh and a 16 × 16 discrete-velocity mesh, **Figure 6** shows the quantum circuit used to simulate the free-molecular flow around a rectangular body, for which the evolution of the flow field starting from an initial uniform flow is shown in **Figure 7**. The key feature in the quantum circuits shown in **Figure 6** is the extended number of control qubits, i.e., beyond the checks on the qubits corresponding to spatial coordinates, control qubits also involve the "BC" qubit (fluid/solid flag) as well as the qubits related to the discrete-velocity indices. This least feature originates from the need to stream only data associated with selected discrete velocities in the used time-integration method.

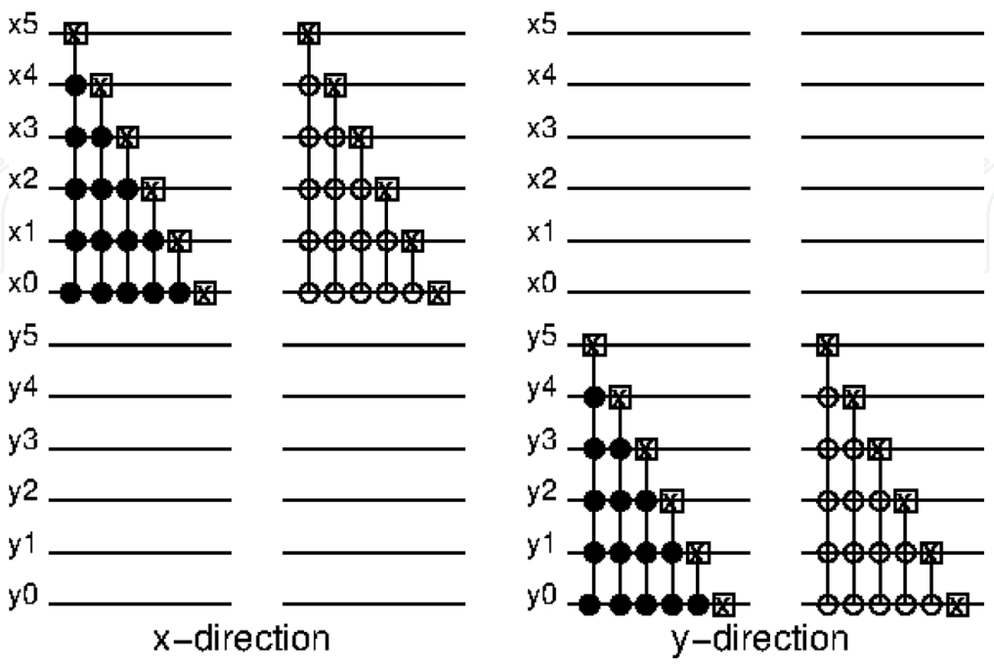

Figure 5.
Quantum circuit implementation of "left streaming" and "right streaming" in x- and y-direction on a 64 × 64 Cartesian mesh.

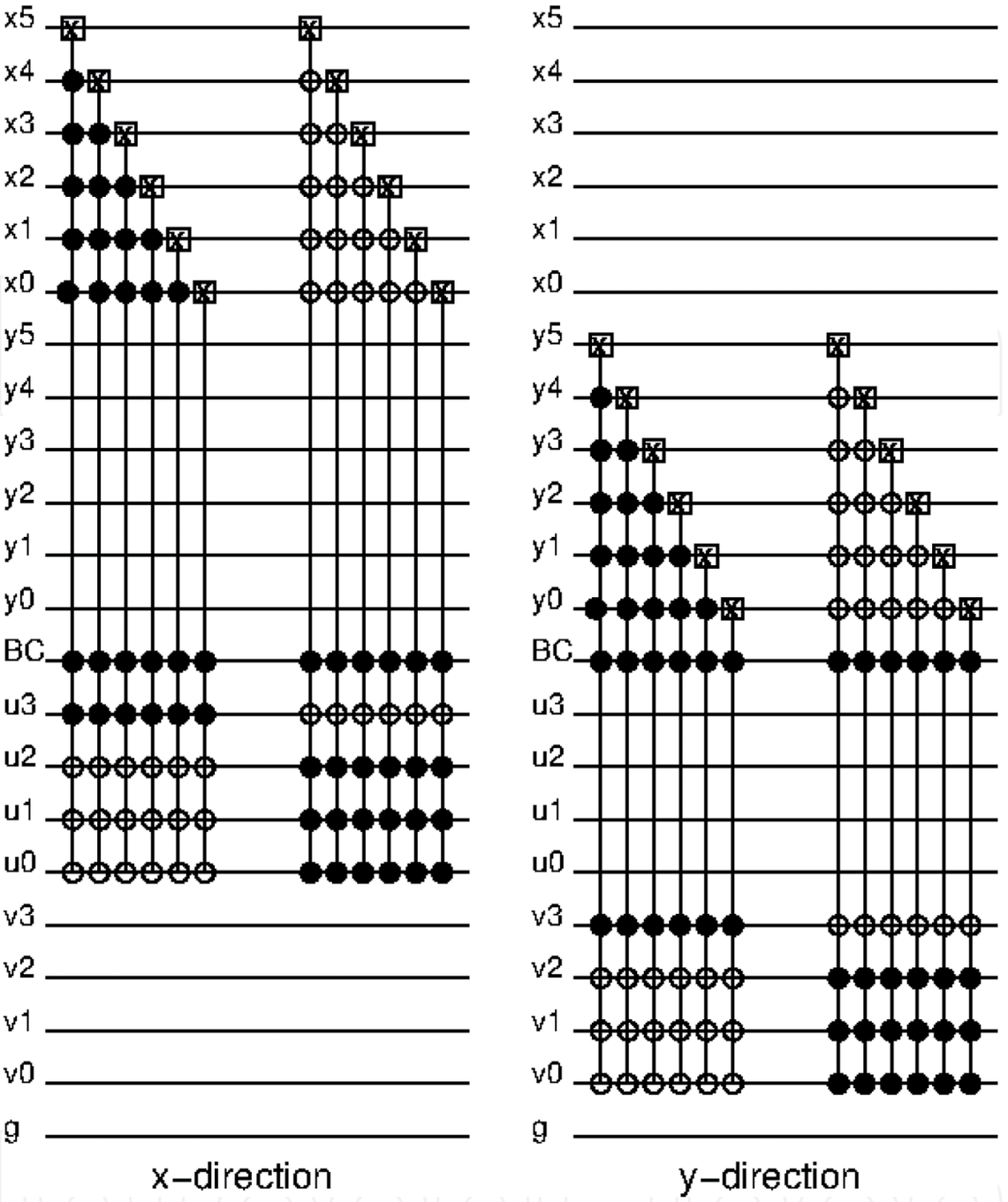

Figure 6.
Quantum circuit implementation of streaming operations for discrete-velocity method (with 16 × 16 velocity mesh). Two-dimensional domain with 64 × 64 Cartesian mesh.

4.1 Quantum circuit implementation of specular-reflection boundary conditions

A key aspect of the flows simulated by the quantum algorithm described here are gas-solid interactions. In this work, specular-reflection boundary conditions are assumed. This means that upon hitting a solid surface, a gas particle will bounce off this surface with the wall-normal velocity component effectively getting reversed and the tangential-flow component being preserved. In **Figure 7**, the time evolution of a Mach 2 free-molecular flow is shown, starting from an initial flow at uniform velocity everywhere.

As can be seen in **Figure 7**, the specular-reflection boundary conditions gradually make the velocity vectors align with the solid wall such that there is no longer a flow into the solid body. This is the physically correct behavior. In the quantum register,

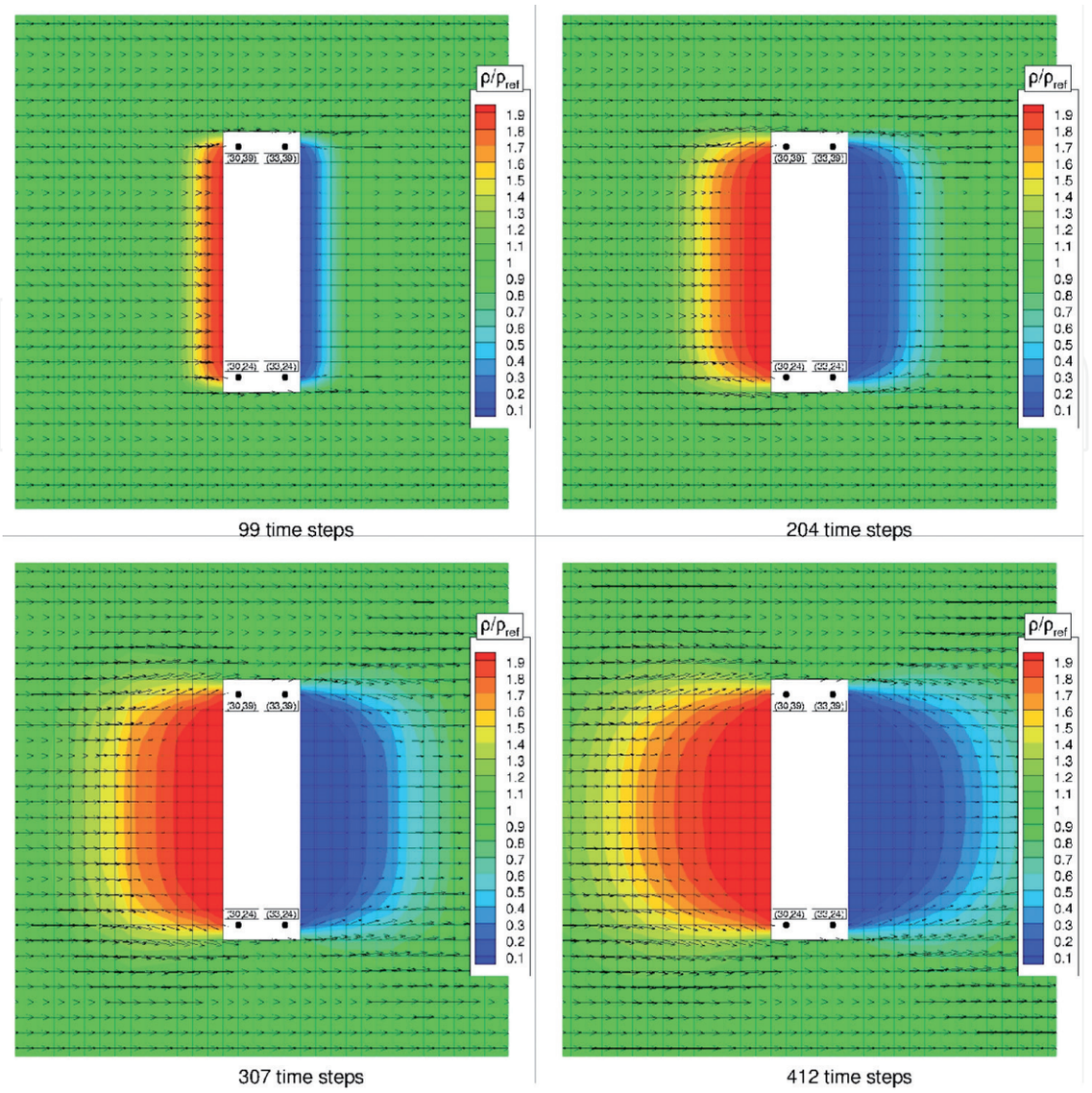

Figure 7.
Discrete-velocity simulation of Mach 2 flow around rectangular body. 64 × 64 Cartesian mesh, velocity-space mesh with 64 × 64 discrete velocities.

the indexing for the velocity-mesh data is designed such that a change of the sign of the discrete velocity implies a bit negation of qubits representing the index of the considered discrete velocity. As an example, **Figure 8** shows the quantum circuit implementation of the specular reflection for a rectangular body. In this case, only 16 × 16 discrete velocities are used for clarity. It can be seen that control qubits are used to "select" cells in space for which to apply the boundary condition. A further control qubit involves the "BC" qubit representing the solid/fluid flag. The negation operation ("X") is applied to the qubits representing the velocity-mesh index to create the "change of sign" of the considered discrete-velocity data. This circuit is shown as an illustration of the quantum algorithm design approach used here.

4.2 Circuit implementations using ancilla qubits

Although the circuits shown in **Figure 6** 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 Toffoli gate.

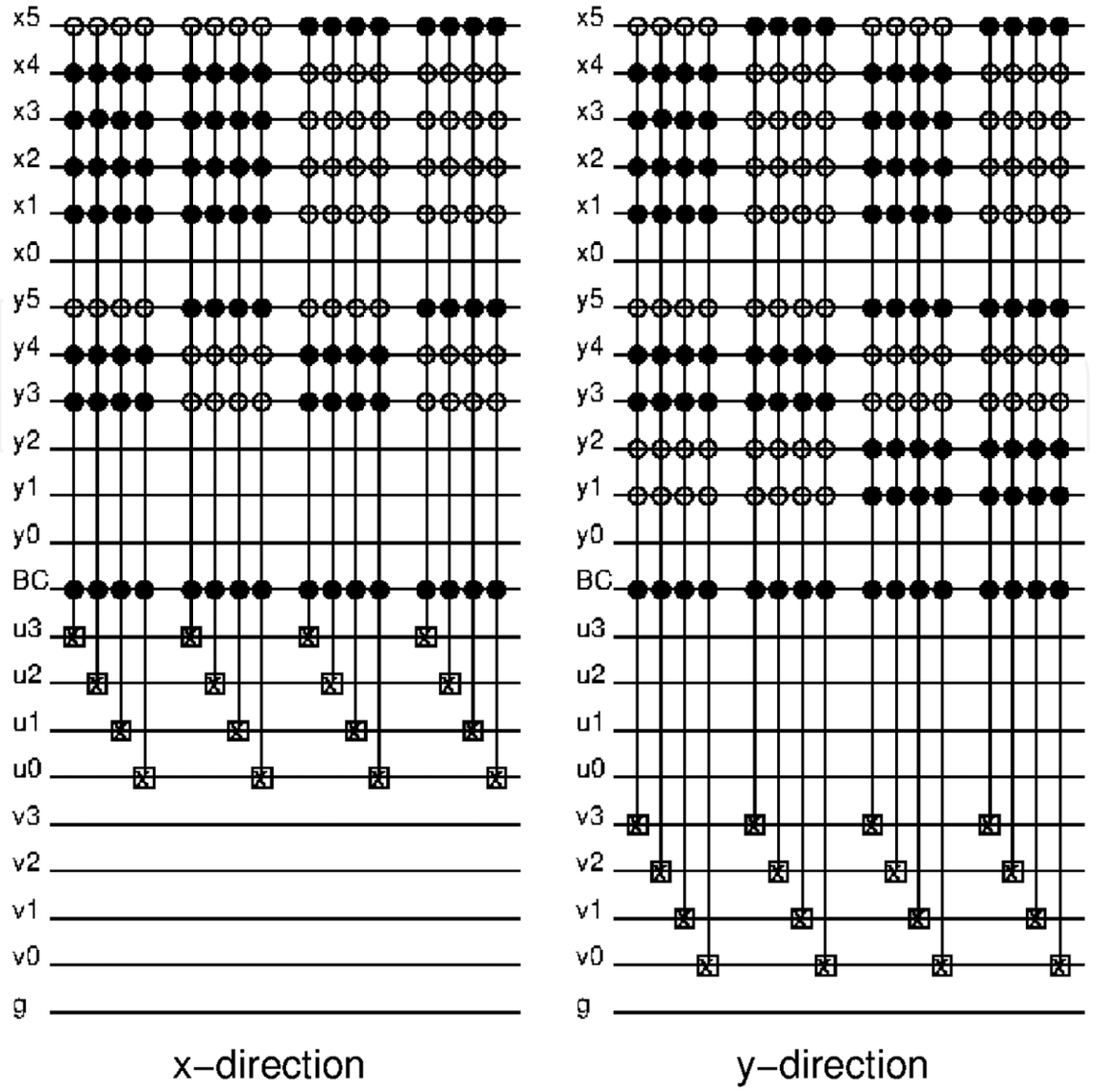

Figure 8.
Quantum circuit implementation of specular-reflection boundary conditions for rectangular body. 64 × 64 Cartesian mesh, 16 × 16 discrete-velocity mesh.

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. As is typical in this context, we assume that the ancilla qubits are initially in |0>, and since these are to be reused multiple times, the transformed circuits need to reset the ancilla qubits of this state at the end of the operations. For the quantum circuits implementing streaming in positive x-direction considered previously, **Figure 9** shows the required circuit transformations for the cases of 1, 2, or 3 ancilla qubits. Increasing the number of qubits will make the circuits more demanding to implement, so in the actual implementation there is an important trade-off between gate complexity (i.e., within the set of native gates available) and total number of qubits. For the 64 × 64 two-dimensional mesh and 16 × 16 discrete velocities considered in **Figure 6**, the transformed circuits show that with three ancilla qubits, the maximum number of control gates is reduced to four in a five-qubit-controlled negation gate. However, for most practical hardware implementations, further transformations would be required.

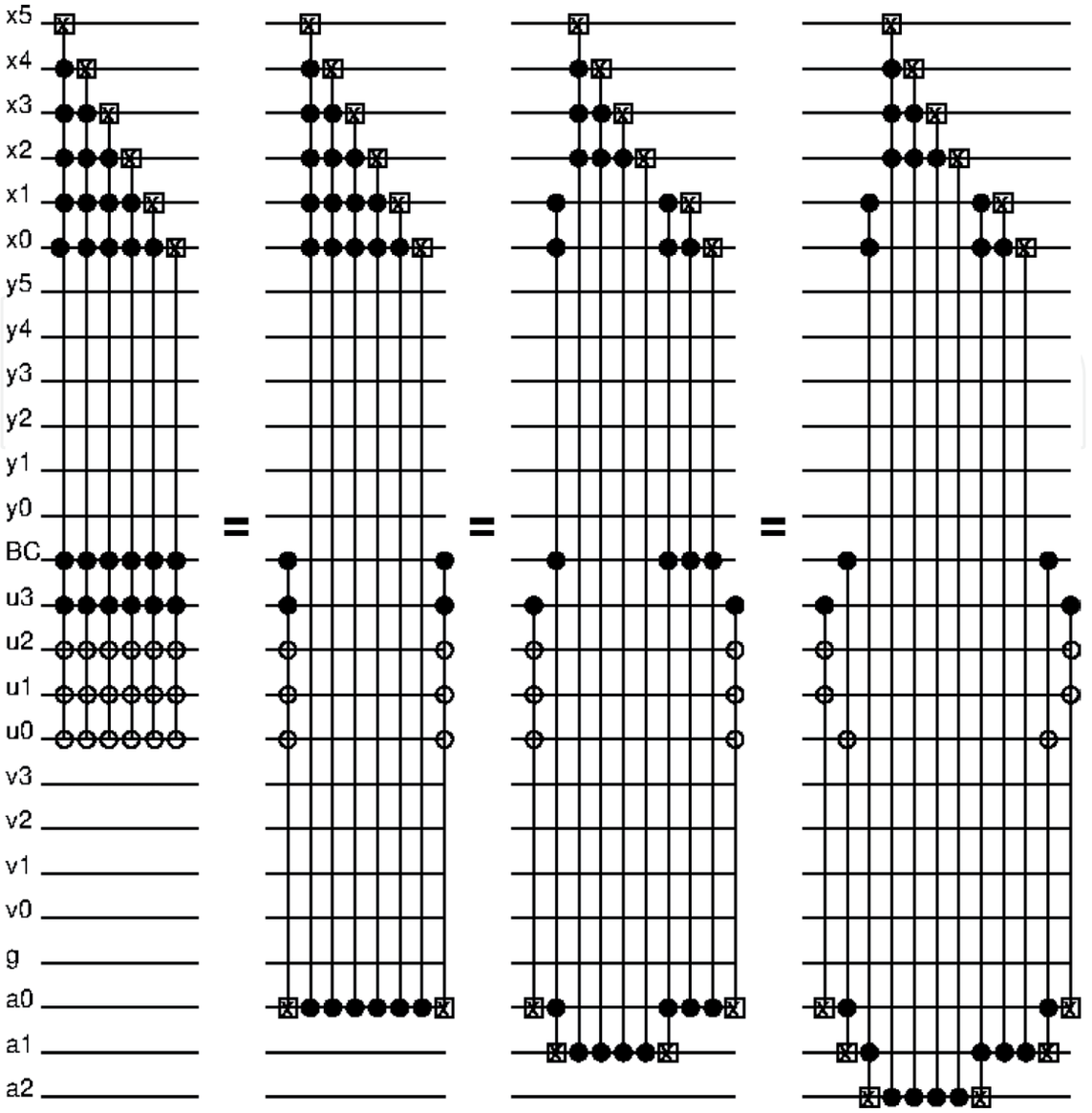

Figure 9.
Quantum circuit implementation of streaming operation. Circuit transformations are shown for addition of 1, 2, and 3 ancilla qubits.

5. Conclusions

This chapter presented two different quantum algorithms with possible applications in computational fluid dynamics. Beyond their very different areas of application, the key differences are the computational model with regard the quantum coprocessor model of quantum computing. The hybrid quantum/classical algorithm for the vortex-in-cell method involves repeated exchanges of information between classical and quantum hardware, i.e., at each time step in the time integration. In contrast, the quantum algorithm implementing a discrete-velocity method for kinetic flow modeling can be performed on the quantum processor for the duration of the simulation, with classical information exchange only required at the start and end of the simulation.

This work addressed a number of key challenges that remain to be investigated further. Firstly, the need for further efficient quantum algorithms as well as a further understanding of how to apply the quantum coprocessor model for this type of flow simulations was investigated. Secondly, the measurement-based extraction of

classical information fundamentally changes the way quantum algorithms for CFD application will most likely be used. Finally, obtaining detailed information on the full flow field will be a challenge, so applications for which only certain characteristics of the solution are desired would present a good choice for future applications.

Acknowledgements

A part of the simulation results presented were obtained using the EPSRC-funded ARCHIE-WeSt High Performance Computer (www.archie-west.ac.uk), EPSRC grant no. EP/K000586/1.

Author details

René Steijl
CFD Laboratory, School of Engineering, University of Glasgow, Glasgow, United Kingdom

*Address all correspondence to: rene.steijl@glasgow.ac.uk

References

[1] Nielsen MA, Chuang IL. Quantum Computation and Quantum Information: 10th Anniversary Edition. 2nd ed. Cambridge, UK: Cambridge University Press; 2010

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

[3] Scherer A, Valiron B, Mau SC, Alexander S, van den Berg E, Chapuran TE. Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2D target. Quantum Information Processing. 2017;**16**(3):1-65. DOI: 10.1007/s11128-016-1495-5

[4] Xu G, Daley AJ, Givi P, Somma RS. Turbulent mixing simulation via a quantum algorithm. AIAA Journal. 2018;**56**(2):687-699. DOI: 10.2514/1.J055896

[5] Steijl R, Barakos GN. Parallel evaluation of quantum algorithms for computational fluid dynamics. Computers & Fluids. 2018;**173**:22-28. DOI: 10.1016/.compfluid.2018.03.080

[6] Harrow AW, Hassidim A, Lloyd S. Quantum algorithm for linear systems of equations. Physical Review Letters. 2009;**15**:150502. DOI: 10.1103/PhysRevLett.103.150502

[7] Cao Y, Papageorgiou A, Petras I, Traub J, Kais S. Quantum algorithm and circuit design solving the Poisson equation. New Journal of Physics. 2013;**15**:013021. DOI: 10.1088/1367-2630/15/1/013021

[8] Britt KA, Humble TS. High-performance computing with quantum processing units. ACM Journal on Emerging Technologies in Computing Systems. 2017;**1**(1):1-39. DOI: 10.1145/3007651

[9] Steijl R, Barakos GN. Coupled Navier-Stokes-molecular dynamics simulations using a multi-physics flow simulation framework. International Journal for Numerical Methods in Fluids. 2010;**62**:1081-1106. DOI: 10.1002/fld.2641

[10] Steijl R, Barakos GN. Coupled Navier-Stokes/molecular dynamics simulations in nonperiodic domains based on particle forcing. International Journal for Numerical Methods in Fluids. 2012;**69**:1326-1349. DOI: 10.1002/fld.2053

[11] Cercignani C. The Boltzmann Equation and its Applications. 2nd ed. New York: Springer; 1987

[12] Vincenti WG, Kruger CH. Introduction to Physical Gas Dynamics. 2nd ed. New York: Wiley; 1967

[13] Alouges F, De Vuyst F, Le Coq G, Lorin E. The reservoir technique: A way to make Godunuv-type scheme zero to very low diffuse. Application to Colella-Glaz solver. European Journal of Mechanics-B/Fluids. 2008;**27**(6):643-664. DOI: 10.1016/j.euromechflu.2008.01.001

[14] Fillion-Gordeau F, MacLean S, Laflamme R. Algorithm for the solution of the Dirac equation on digital quantum computers. Physical Review A. 2017;**95**(4):042343. DOI: 10.1103/PhysRevA.95.042343