Computers and Fluids 173 (2018) 22–28

Contents lists available at ScienceDirect

Computers and Fluids

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

Parallel evaluation of quantum algorithms for computational fluid dynamics☆

René Steijl[1,*], George N. Barakos[2]

University of Glasgow, School of Engineering, James Watt South Building, G12 8QQ, UK

ARTICLE INFO

Article history:
Received 24 January 2017
Revised 21 March 2018
Accepted 28 March 2018
Available online 29 March 2018

MSC:
00-01
99-00

Keywords:
Quantum computing
Quantum computer simulator
Vortex-in-cell method

ABSTRACT

The development and evaluation of quantum computing algorithms for computational fluid dynamics is described. A hybrid classical/quantum hardware approach is assumed where selected computationally intensive parts of the solver are implemented as quantum circuits. The vortex-in-cell method is considered as an example where the Quantum Fourier Transform is used to build a Poisson solver. Computational aspects of simulating the required quantum circuits on a classical parallel computer are discussed including an analysis of the required data exchanges for a distributed-memory parallelization using message-passing. The effect of errors and noise in the quantum algorithm on the flow solution is analyzed and it is shown that despite inevitable noise and uncertainties, meaningful flow simulations can be performed using a hybrid classical/quantum hardware approach. An improved version of the vortex-in-cell method with increased resilience to noise is also discussed along with suggestions for future steps. The presented work is limited to a single CFD algorithm. However, building on this work, a broader range of algorithms will be considered in future work.

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 and two decades after their invention, Shor's algorithm for factoring composite integers and Grover's algorithm for quantum search are still among the main applications. 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 [2], numerical gradient estimation [3] and the Poisson equation [4].

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. A hybrid classical/quantum hardware approach is assumed where selected computationally intensive parts of the solver are implemented as quantum circuits.

Quantum computer applications aim to achieve a computational speed-up relative to classical computers by using unique quantum effects. The first and most important for the algorithms considered here is quantum parallelism, based on the use of qubits representing a superposition of infinitely many possible states of a binary system. Quantum entanglement and quantum teleportation are the other main quantum effects used to create computational capabilities not available to classical computers.

Quantum parallelism used in quantum computing achieves potential performance benefits over conventional hardware and algorithms by representing 2^{n_q} simultaneous values in a qubit register with n_q qubits. A calculation using this register calculates all possible outcomes for the 2^{n_q} input values. However, in order to read out the results of a calculation, the output of the calculation needs to be *measured*. This measurement forces all the qubits to a particular value thereby destroying the parallel state.

With current technologies, implementing quantum computers with many qubits is extremely challenging. One of the main obstacles is decoherence, in effect an interaction of quantum systems with their environment that destroys the superposition on which

☆ An earlier version of this work was presented at the ParallelCFD2017 conference

* Corresponding author.
E-mail addresses: Rene.Steijl@glasgow.ac.uk (R. Steijl), George.Barakos@glasgow.ac.uk (G.N. Barakos).
[1] Senior lecturer
[2] Professor

https://doi.org/10.1016/j.compfluid.2018.03.080
0045-7930/

most of the potential performance gains of quantum algorithms over classical algorithms are built. Furthermore, there are *operational errors* caused by quantum gates in a quantum circuit implementation not performing the intended transformations perfectly.

A key aspect of work in quantum algorithm development and implementation of quantum computers is therefore the analysis of performance and robustness of quantum circuits in the presence of decoherence and operational errors. For a realistic implementation, quantum error-correcting codes will typically be required and the analysis of the effectiveness of these codes forms another reason for the detailed analysis of quantum algorithms and hardware on classical computers.

In the literature, a range of general-purpose parallel simulators for quantum algorithms and quantum circuits have been presented that typically have been applied to the Shor and Grover algorithms, e.g. Niwa et al. [5] and Miquel et al. [6]. The effect of decoherence on the Quantum Fourier Transform (QFT) was analyzed by Cirac and Zoller [7] and Barenco et al. [8]. The present work is a targeted investigation of quantum algorithms for flow simulations, employing QC simulation techniques directly implemented into the C++ framework MΦC [9,10], that forms the basis of the CFD methods studied here. Demonstrating how simulation of quantum circuits relevant to CFD can be performed on conventional parallel computers and a first example application to a relevant simulation method are the aims of the present work. The vortex-in-cell method is considered and applied to a test case involving two colliding vortex rings.

Recently, the vortex-in-cell method has been investigated in detail for efficient simulations on GPUs showing a good speed-up of a single GPU relative to a CPU [11]. This study showed that the method could be well suited to computing hardware beyond classical CPU-based systems.

In particular, the focus in the present work is on the QFT-based Poisson solver and the sensitivity of the developed vortex-in-cell method to measurement and operational errors in the solution of the Poisson problems. The QFT is simulated as a quantum circuit involving up to 24 qubits facilitating the solution of the Poisson equation for a uniform mesh of up to 256^3. To account for measurement and operational errors, numerical noise is injected into the simulated outcome of the Poisson equation and its effect on the flow solution considered. Since measurement and operational errors are inevitable in practical quantum computer implementations, the present work also considers a modification of the vortex-in-cell method in which the velocity field is derived from spectral space rather than through finite differences applied to the velocity potential function. The present work therefore focusses on a single CFD algorithm, and future work will expand this to a broader range of algorithms used in CFD applications. This work starts with a brief review of quantum computing principles, followed by a detailed analysis of the simulation of relevant quantum circuits on parallel classical computers. Then, the vortex-in-cell method is described. Finally, results for a colliding vortex pair with different levels of noise representing measurement and operational errors are discussed. Finally, conclusions and future lines of work are discussed.

2. Quantum computing principles

Before describing the quantum algorithms considered here for computational fluid dynamics, a number of definitions commonly used in quantum computing literature is briefly reviewed. A more detailed description of quantum computing principles can be found in the textbook by 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 $|c_0|^2 + |c_0|^2 = 1$. For a quantum system with n_q coherent qubits, measurement of the quantum system can lead to 2^{n_q} possible outcomes. A collection of n_q qubits is called a *register* of size n_q. Information is stored in the registers in binary form. For n_q qubits, a superposition is created as $\left|c_{n_q-1}\right\rangle \otimes \left|c_{n_q-2}\right\rangle \otimes \ldots |c_1\rangle \otimes |c_0\rangle$, often written as $\left|c_{n_q-1}c_{n_q-2}\ldots c_1c_0\right\rangle$, which represents a quantum register prepared with the value $c = 2^0c_0 + 2^1c_1 + \ldots + 2^{n_q-1}c_{n_q-1}$. Here $c_0, \ldots, c_{n_q-1}$ are complex numbers representing quantum wave number amplitudes, defining the state vector of the system $\psi = (c_0, \ldots, c_{n_q-1})^T$. In the following the qubits are numbered 0 to $n_q - 1$ running left to right, i.e. with qubit 0 representing the most significant bit and qubit $n_q - 1$ the least significant bit. 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. Here 1-qubit and 2-qubits gates are considered. A quantum network consists of quantum logic gates whose computational steps are synchronized in time. The output of some of the gates are connected to the input of others. The present work considers the quantum circuit model of quantum computing, i.e. the quantum computer is considered as an implementation of a quantum network.

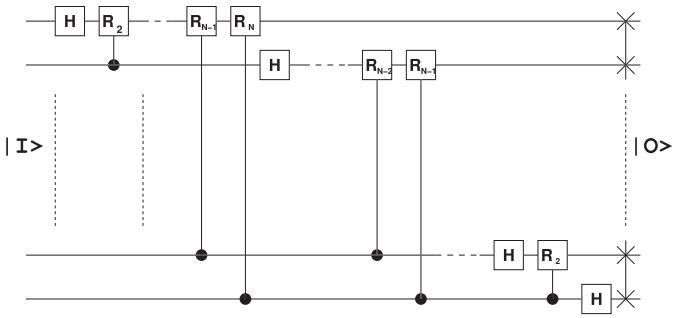

Fig. 1. Circuit representation of the Quantum Fourier transform. The input quantum state $|I\rangle$ is transform into an output quantum state $|O\rangle$. In the output state vector represented by 2^{n_q} complex numbers, vector elements appear in bit-reversed order. This is represented by the crosses in the circuit output channels.

3. Quantum Fourier Transform

The Quantum Fourier Transform (QFT) provides an efficient method for computing the Discrete Fourier Transform

$$QFT(|\psi\rangle) = \sum_{j=0}^{N-1} b_j|j\rangle \; ; \; b_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} c_k \cdot \exp\left(\frac{2\pi i \cdot j \cdot k}{N}\right) \qquad (1)$$

Using the quantum circuit model of quantum computing, the QFT can be represented by the circuit sketched in Fig. 1. This circuit takes an input quantum state $|I\rangle$ and outputs the result in the quantum state $|O\rangle$. As can be seen, the circuit involves a series of gate operations, e.g. H and R_n, where H represents the single-qubit Hadamard operation and R_n the controlled-rotation acting on two qubits. R_n performs a conditional phase shift, i.e. it applies a phase factor $\exp(2\pi i/2^n)$ only if the considered two qubits are both in their $|1\rangle$ state, leaving the other three basis states unaffected. The single-qubit Hadamard transformation and two-qubit

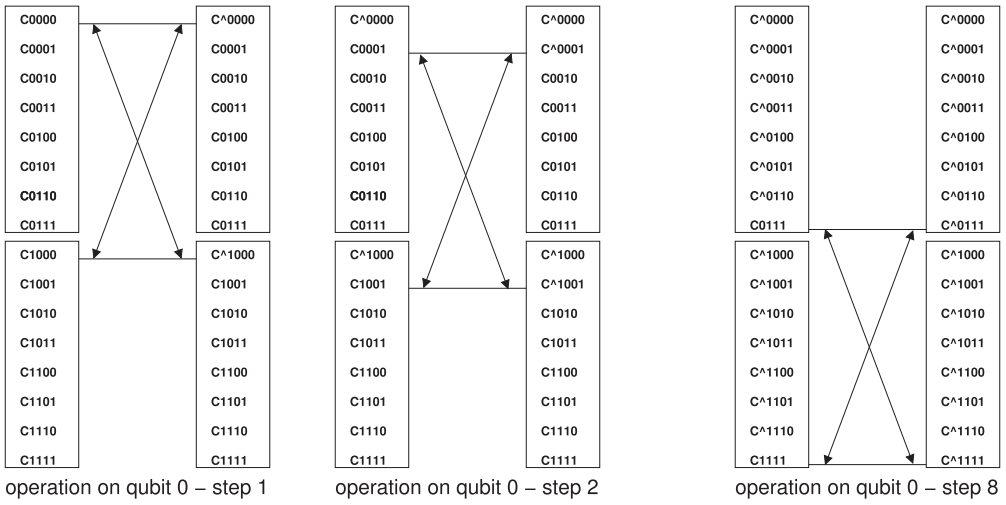

Fig. 2. Single-gate operation on the first qubit in a 4-qubit register. Examples shows the required steps for the state vector distributed over two processes of a distributed-memory computer. As can be seen, the single gate operation involves 8 steps modifying the 2^4 complex numbers in the state vector. Each step requires data from another processor.

controlled rotation operations are defined as,

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad ; \quad R_n = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^n} \end{pmatrix} I \tag{2}$$

Similar to the results of the classical FFT algorithms employing recursion, the output state represented by 2^{n_q} complex numbers occurs in 'bit-reversed' order.

4. Parallel quantum computer simulation

The number of processors is assumed as $np = 2^p$ with $p < n_q$. The state vector is defined as $\psi = (c_0, \ldots, c_{n_q-1})^T$ as is equally split over the processes within an MPI Communicator, i.e. each process stores n^{n_q-p} amplitudes, with amplitudes $c_0, \ldots, c_{n_q-p-1}$ allocated to process 0, etc. Qubit 0 is the left-most qubit in the register and represents the most significant bit, and single-qubit operations on this qubit will thus affect all 2^{n_q} amplitudes in the register. For qubits further to the right in the register, a single-qubit operation will involve increasingly smaller subsets of the register. For np processes, it can then be shown that for qubit index i_q a single-qubit operation does not need exchange of data between processors if $2^{i_q} > np$. For an example register with 4 qubits this is illustrated in Fig. 2 for a single-qubit operation on the 1st qubit and Fig. 3 for the 2nd qubit. For a two-qubit operation, a 4×4 matrix represents the operation and for the controlled-rotation gates used here, the matrix is a diagonal matrix, e.g. R_n in Eq. (2). This means that independent of the qubits considered, data exchange between processes is not required. If we assume i_{q1} to be the index in the register of the control-qubit and $i_{q2} > i_{q1}$ that for the qubit to be transformed, then application of the two-qubit controlled-rotation gates will involve multiple processes when $2^{i_{q1}+1} > np$.

4.1. Multiple realizations in noise analysis

For the considered algorithms designed to perform computational fluid dynamics simulations, it is envisaged that part of the simulation is conducted on a classical computer, while particular, computationally costly parts are executed on a quantum computer. The results of the 'quantum' part of the simulation need to be coupled to the remainder of the algorithm and this process involves measuring the quantum state of the qubit register. The collapse of the wave function involved in the process will result in a solution in which each qubit takes on one of two possible value with a probability defined by the corresponding amplitude of the wavefunction. This means that typically multiple realizations will be needed with statistical sampling to obtain the required output.

The analysis of the effect of operational, decoherence and statistical measurement errors requires multiple realizations to be considered and sampled. The present work considered multiple realizations conducted sequentially. However, the quantum simulation method developed also allows parallel execution of multiple realizations by creating multiple MPI Communicators in which separate realizations are run in parallel. A similar approach has been previously used by Tabakin and Julia–Diaz [12] for analysis of Shor and Grover algorithms. For more complex algorithms considered here, this has not yet been published previously, and such an analysis for a more complex CFD-related algorithm forms a key innovation of the present work.

Table 1
Wall clock time for discrete Fourier transform for one direction.

n_q	$nlog_2n$	t_{QFT}(s)	t_{FFT}(s)	t_{QFT}/t_{FFT}
18	384	$1.71 \cdot 10^{-4}$	$1.25 \cdot 10^{-5}$	13.6
21	896	$4.57 \cdot 10^{-4}$	$2.63 \cdot 10^{-5}$	17.4
24	2048	$1.16 \cdot 10^{-3}$	$6.03 \cdot 10^{-4}$	19.2

4.2. Analysis of QFT-based Poisson solver

To analyze the computational cost of the QFT-based solver when executed on a classical computer, the wall-clock times for different parts of the current solver running on a single core of a 4-core Intel Xeon processor were measured. Different mesh sizes were considered, i.e. 64^3, 128^3 and 256^3 corresponding to 18, 21 and 24 qubits. Table 1 presents the CPU times for the execution of the discrete Fourier transform for points along a single line in one coordinate direction. As can be seen, the computational complexity of the QFT implementation still scales largely as $Nlog_2N$ like the standard *FFT* implementation shown for comparison, while the required time is up to 20 times longer.

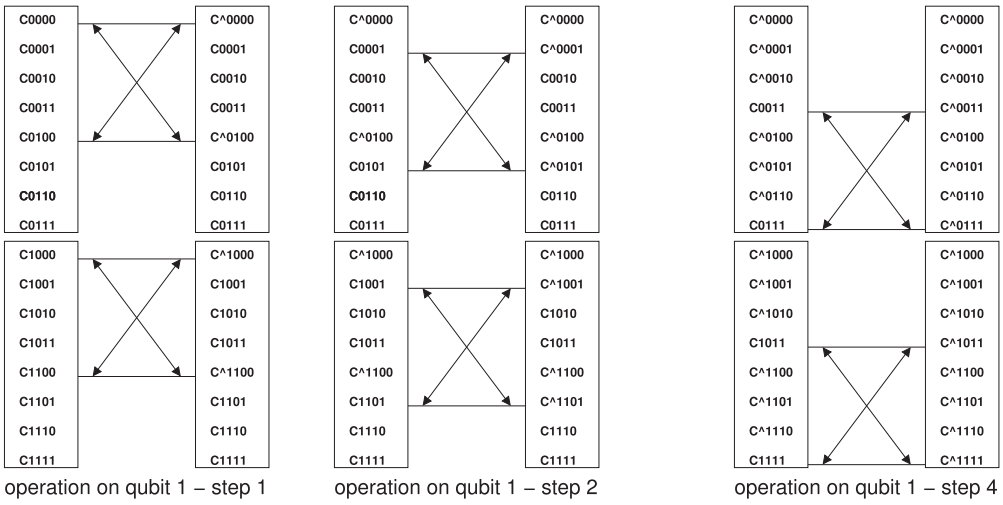

Fig. 3. Single-gate operation on the second qubit in a 4-qubit register. Examples shows the required steps for the state vector distributed over two processes of a distributed-memory computer. As can be seen, the single gate operation involves 4 steps independently performed on both processors modifying the 2^4 complex numbers in the state vector. For this case, no data exchange between processors is required.

The findings of relative CPU times for FFT and QFT obtained here are consistent with previous work by Niwa et al. [5] who analyzed the QFT using their general-purpose parallel simulator and found that the execution time of the QFT in the simulator is 20 – 30 times slower than that of a classical FFT algorithm.

5. Vortex-In-Cell method

The vortex-in-cell method [11,13] 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 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 method, solves the incompressible-flow Navier–Stokes equations transformed into the Helmholtz equation for vorticity evolution,

$$\frac{\partial \omega}{\partial t} + \big(\mathbf{u}\cdot\nabla\big)\omega = \big(\omega\cdot\nabla\big)\mathbf{u} + \nu\Delta\omega \;;\; \omega = \nabla\times\mathbf{u} \tag{3}$$

A viscous splitting algorithm is employed, which first solves the inviscid equation,

$$\frac{\partial \omega}{\partial t} + \big(\mathbf{u}\cdot\nabla\big)\omega = \big(\omega\cdot\nabla\big)\mathbf{u} \tag{4}$$

where the right-hand side represents the vortex-line stretching term. Next, the viscous effects are introduced by solving the diffusion equation,

$$\frac{\partial \omega_i}{\partial t} = \nu\Delta\omega_i \tag{5}$$

where $\boldsymbol{\omega}_i$ is the vorticity field obtained after the inviscid step. The advection of vorticity represented by the second term on the left-hand side of Eq. (4) is modeled using Lagrangian motion of vortex particles on a regular background mesh. The vortex-line stretching term is evaluated on this regular mesh at the start of the time-step. In the current implementation, the vorticity is transferred from the particle to the mesh at the end of each time-step. Since this is done after each step of the inviscid equation, the viscous effects can be introduced by Eq. (5), discretized on the regular background mesh using second-order accurate central differences. This is followed by a regularization step in which the particles are placed at the nodes of the regular background mesh. The vorticity represented on this mesh is the used to re-initialize the strength of the vorticity in each of the vortex particles. The velocity field is then evaluated from a vector potential **A**, which follows from solving the following three Poisson problems,

$$\Delta A_i = -\omega_i \quad ; \quad i = 1,2,3 \;;\; \mathbf{u} = \nabla\times\mathbf{A} \tag{6}$$

with i representing the coordinate directions. The solution of the Poisson problems forms the quantum part of the present hybrid classical/quantum algorithm. For the time-integration, a predictor-correction approach is used, such that for each time step, the steps described above need to be executed twice. For the transfer of vorticity from particles-to-mesh and mesh-to-particles, the following interpolation kernel (M_4') is used,

$$\zeta(r/\sigma) = \begin{cases} 1 - \frac{5}{2}\big(r/\sigma\big)^2 + \frac{3}{2}\big(r/\sigma\big)^3 & \text{if } 0 \le r/\sigma \le 1 \\ \Big[2 - \big(r/\sigma\big)^2\Big]^2\Big(1 - \big(r/\sigma\big)\Big)/2 & \text{if } 1 \le r/\sigma \le 2 \\ 0 & \text{if } 2 \le r/\sigma \end{cases} \tag{7}$$

where r is the distance from the center of the kernel and σ is the mesh width. For the present 3D cases, a tensor-product of this kernel is used.

In the first version of the vortex-in-cell method considered here, the velocity field is obtained from the vector potential as shown in Eq. (6), where the gradients of the potential are evaluated on the Cartesian mesh using second-order accurate finite-differences. In the considered hybrid quantum/classical computing approach, the vector potential obtained from the quantum-based solution of the Poisson equation will include noise when used in the classical part of the algorithm as a result of measurement and operational errors. In the considered algorithm, the finite-difference computation of the velocity is performed on the classical computer, so that noise levels are amplified relative to the levels in the original vector potential as a result of applying finite-differences to noisy data. An improved version of the vortex-in-cell method with increased resilience to noise was derived in the present work. The underlying principle was the elimination of the finite-differencing of the vector potential. Since the Poisson solver used in the considered rectangular Cartesian domains is based on the discrete Fourier transform, here implemented using the QFT, the second version of the vortex-in-cell method was designed to evaluate the velocity field from the Fourier transforms of the solution to the Poisson equation of Eq. (6). The derivatives of the vector

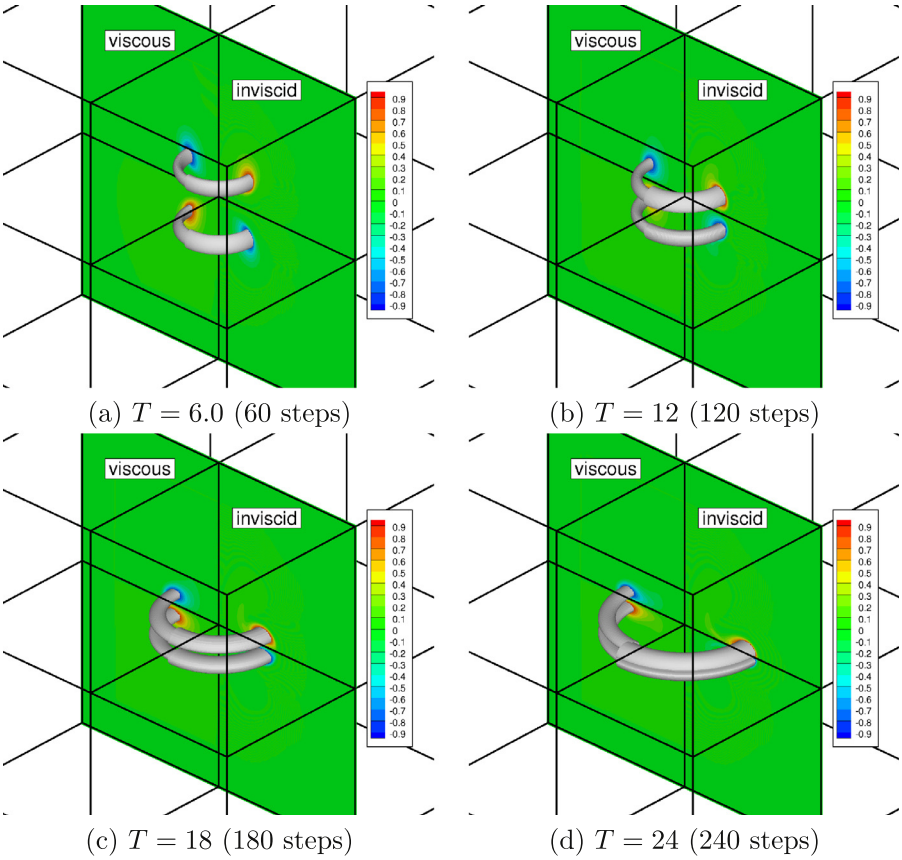

Fig. 4. Colliding vortex rings simulated with and without viscosity. Vorticity isosurface at $\omega = 1$. Uniform 128^3 background mesh was used.

potential in one of the three coordinate directions are obtained after the Discrete Fourier Transform has been applied using the QFT. In the 3D domains considered, the data is stored using $n_q/3$ qubits for each coordinate direction, e.g. in the current implementation the first $n_q/3$ qubits represent the $x-$direction, followed by $n_q/3$ qubits each for both $y-$direction and $z-$direction. As a first step, the following single-qubit operator is applied to the first qubit in the relevant range of qubits,

$$\begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix} \tag{8}$$

This step can be efficiently executed on a quantum computer, since it only involves a single-qubit operation. The remaining work needed was designed to be executed on classical hardware and involves multiplying the complex coefficients resulting from the QFT and the single-qubit operation described above, with real-valued numbers representing wave numbers. An inverse QFT transform will then return the derivative in the considered coordinate direction in the 3D domain. In summary, the second implementation of the velocity computation compares to the finite-difference approach as follows:

- the finite-difference method applies finite-difference operations on a 'noisy' set of data (representing the vector potential) defined on the three-dimensional mesh. This process will typically amplify the noise levels of the derivative as compared to the original signal
- the QFT-based approach for derivatives avoids this noise-amplifying step
- the single-qubit operation required for the derivative-calculation can be done efficiently on a quantum computer
- the main drawback of the QFT-based approach as implemented here, is that it assumes that the multiplication with the wave numbers can be done efficiently on the classical hardware in the envisaged hybrid approach. Furthermore, it has so far been assumed that no additional noise is created in the transfer of the Fourier coefficients from quantum to classical hardware. This aspect will be studied in more detail in future work.

It follows that the proposed QFT-based approach for the velocity-field evaluation would mainly be of interest for quantum computer applications in case the multiplication with the wave numbers can be performed completely on a quantum computer. At present it is not yet clear if and how this can be done efficiently.

6. Simulation of colliding vortex rings

A pair of colliding vortex rings is considered with equal but opposite-signed strength. A non-dimensional formulation is used with the flow domain a unit cube and the ratio of vortex core radius (r_c) to ring radius (R) equal to $r_c/R = 1/5$. For a uniform 128^3 mesh the vortex core radius is 5 mesh widths, while the vortex core radius is 10 mesh widths for a uniform 256^3 mesh. The vertical spacing between the centers of the vortex rings at the start of the simulation is 1.5 times the ring radius. The vortex rings are initialized with a Gaussian profile with a unit (non-dimensional) peak vorticity. Inviscid results and results for a circulation based Reynolds number of 10^5 are compared in Fig. 4, showing the increasing radius for both rings due to mutual velocity induction. For the inviscid result the vorticity magnitude increase due to vortex-line stretching effect is somewhat more pronounced than the viscous simulation where part of the increase is cancelled by viscous dissipation. Results on a 256^3 mesh were also computed. However,

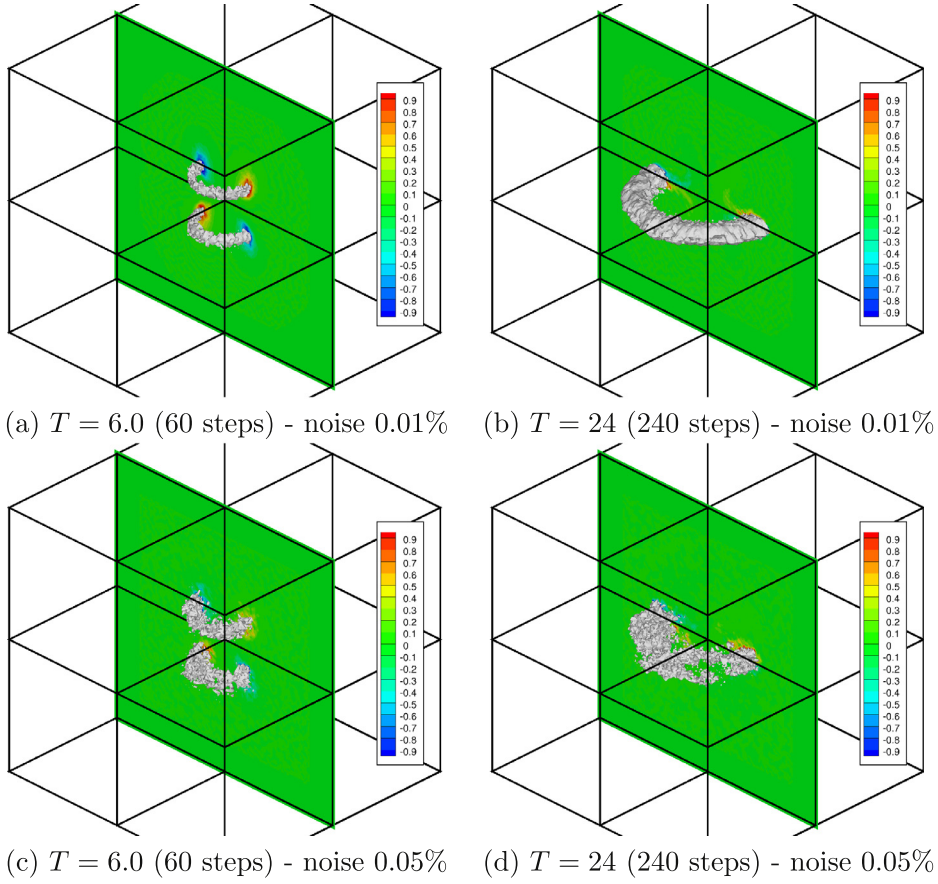

Fig. 5. Colliding vortex rings with simulated noise. (a)-(b) scaled noise magnitude 0.0001, (c)-(d) scaled noise magnitude 0.0005. Vorticity isosurface at $\omega = 1$. Uniform 128^3 background mesh was used. Inviscid flow simulations.

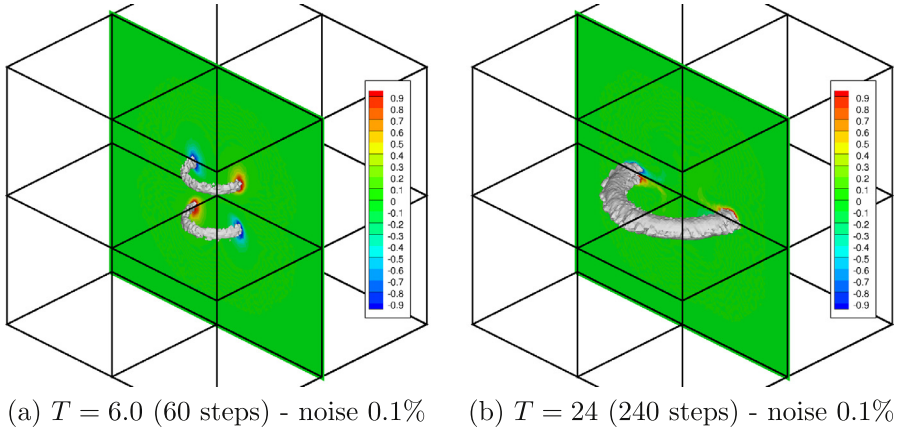

Fig. 6. Colliding vortex rings with simulated noise. Velocity field evaluated in spectral space. Scaled noise magnitude 0.001. Vorticity isosurface at $\omega = 1$. Uniform 128^3 background mesh was used. Inviscid flow simulations.

for the time span considered the increased spatial resolution created negligible differences, so for the remaining simulations a 128^3 mesh was used, with approximately 884,736 vortex particles located in mesh points centered around the vortex rings. The domain was decomposed in 8 subdomains for parallel simulation with a maximum of 8 processes. Due to the need for multiple realizations in the quantum simulation part of the simulation, and the design of the simulator created to perform these simultaneously in separate MPI Communicators, a larger number for processes for each realization has so far not been considered.

The sensitivity to operational errors of the quantum gates and statistical sampling errors due to measurement is investigated. The vector potential is computed from the solution of a Poisson with the vorticity forming the source term as in Eq. (6). Since ideal quantum gate operations involve unitary operations, the algorithm used here first normalizes the system and stores the scaling factor involved. Then random noise is added to the solution of the Poisson system before the scaling factor is applied in the reverse direction. The vortex-in-cell method then applies second-order finite differences to create an updated velocity field defined on the background mesh, which is subsequently transferred to the vortex

particles for the advection step in the next time step. Here two amplitudes of the random noise were considered, e.g. 0.0001 and 0.0005 relative to the normalized state vector in the quantum network representing the QFT based Poisson solver. More elaborate noise scenarios with more detailed gate operational errors and decoherence errors will be considered in future work. Fig. 5 show the results for both noise amplitudes at two non-dimensional time instances. Clearly, the larger noise amplitude considered leads to significant errors early in the simulation and at later stages both vortex rings have lost their coherence. In contrast, the smaller noise amplitude enables the simulation to maintain the coherence of the vortex rings as well as the main fluid dynamics aspects of the interaction. Clearly, relative to the noiseless simulations shown previously, the effects are still significant but for simulations focussing on the main large-scale dynamics, the introduced noise could be acceptable. In the conventional vortex-in-cell formulation used so far, the noise in solution for the vector potential is enhanced for the velocity field since this is obtained from finite-differences from this potential field. A novel formulation of the vortex-in-cell method was also developed, using the fact that the Poisson solver is based on Fourier analysis. The new approach involves evaluation of the velocity field directly from the Fourier modes in the Poisson solver and therefore avoid the need for finite-differencing the potential field. Fig. 6 shows results for this formulation, where the noise level is increased to 0.001, and which show a coherence of the computed vortex system similar to that for the lowest noise level, i.e. 0.0001, for the conventional velocity evaluation. These results clearly show that with suitable modifications to this particular vortex-in-cell implementation the sensitivity to noise and uncertainty can effectively be reduced.

7. Conclusion

The application of the Quantum Fourier Transform in a Poisson solver applied to a vortex-in-cell method was described and analyzed. A hybrid classical/quantum hardware approach was assumed where the computationally intensive Poisson solver was executed in quantum circuits. In applying this approach, the key aspects of errors introduced in the quantum circuit, here focussed on operational errors, as well as uncertainty introduced by measurement of the quantum system needed to extract classical information, were analyzed. It was demonstrated what levels of errors the developed vortex-in-cell method can tolerate to still produce meaningful answers to the example application considered, i.e. the collision of two vortex rings. In terms of simulating quantum algorithms on parallel computers, the presented work discussed the required data exchange required for 1-qubit and 2-qubit operations and its dependency on the position of the qubit within the register as well as the number of processors used in the simulation. Based on these initial findings, we believe that for future applications of quantum hardware the type of parallel simulations discussed here will play an important role and that the hybrid classical/quantum approach to computing represents a promising line of development. For a range of CFD algorithms, devising changes to increase resilience to noise will facilitate the introduction of quantum computing techniques. Future work will consider a more detailed error analysis and a wider range of algorithms used in computational fluid dynamics. Decoherence errors were not modelled in the present work, it will be considered in future work.

References

[1] Nielsen M, Chuang I. Quantum computation and quantum information: 10th anniversary edition. Cambridge University Press; 2010.
[2] Harrow A, Hassidim A, Lloyd S. Quantum algorithm for linear systems of equations. Phys Rev Lett 2009;15:150502.
[3] Jordan S. Fast quantum algorithm for numerical gradient estimation. Phys Rev Lett 2005;95:050501.
[4] Cao Y, Papageorgiou A, Petras I, Traub J, Kais S. Quantum algorithm and circuit design solving the poisson equation. New J Phys 2013;15:013021.
[5] Niwa J, Matsumoto K, Imai H. General-purpose parallel simulator for quantum computing. Phys Rev A 2002;66:062317.
[6] Miquel C, Paz J, Perazzo R. Factoring in a dissipative quantum computer. Phys Rev A 1996;54(4):2605–13.
[7] Cirac J, Zoller P. Quantum computations with cold trapped ions. Phys Rev Lett 1995;74(20):4091–4.
[8] Barenco A, Ekert A, Suominen K, Törmä P. Approximate quantum fourier transform and decoherence. Phys Rev A 1996;54(1):139–46.
[9] Steijl R, Barakos G. Coupled Navier-Stokes-molecular dynamics simulations using a multi-physics flow simulation framework. Int J Numer Methods Fluids 2010;62:1081–106.
[10] Steijl R, Barakos G. Coupled Navier-Stokes/Molecular Dynamics Simulations in Nonperiodic Domains on Particle Forcing. Int J Numer Methods Fluids 2012;69:1326–49.
[11] Kosior A, Kudela H. Parallel computations on GPU in 3D using the vortex particle method. Comput Fluids 2013;80:423–8.
[12] Tabakin F, Julio-Diaz B. QCMPI: a parallel environment for quantum computing. Comput Phys Commun 2009;180:948–64.
[13] Cottet G, Poncet P. Simulation and control of three-dimensional wakes. Comput Fluids 2004;33(5):697–713.