

An Introduction to Algorithms in Quantum Computation of Fluid Dynamics

Sachin S. Bharadwaj and Katepalli R. Sreenivasan

Department of Mechanical and Aerospace Engineering

New York University

370 Jay Street, Brooklyn

New York, 11201

United States of America

sachin.bharadwaj@nyu.edu

katepalli.sreenivasan@nyu.edu

ABSTRACT

Studies of strongly nonlinear dynamical systems such as turbulent flows call for superior computational prowess. With the advent of quantum computing (QC), a plethora of quantum algorithms have demonstrated, both theoretically and experimentally, more powerful computational possibilities than their classical counterparts. However, for QC to emerge as an indispensable tool for practical applications, the exigency is not just for novel protocols that process quantum information but also for extracting it wisely in classical formats that cater to the solution of practical problems. Here we draw attention to potential methods of conducting fluid mechanics research using QC, which we call Quantum Computation of Fluid Dynamics (QCFD). Starting with a brief introduction to QC, we will distill a few key tools and algorithms from the huge spectrum of methods available and evaluate possible approaches of QC in fluid dynamics. Further, as an example, we demonstrate an end-to-end implementation of modified Quantum Linear System Algorithms (QLSA) to study problems such as the Poiseuille flow. We also introduce here a new, high performance QC simulator, specific to fluid dynamics, which we call “QuOn”, designed to simulate most standard quantum algorithms. We shall present results using both QuOn and the IBMQ–Qiskit tools and shed light on essential contributions needed to make QCFD simulations practicable.

Contents

I Introduction and the Big Picture of Quantum CFD	4
1.0 Introduction	4
2.0 Motivation	5
3.0 Quantum Computation - Some Fundamentals	6
3.1 What is Quantum Computation?	6
3.2 Qubits	6
3.3 Quantum Gates, Circuits and Algorithms	9
3.4 Quantum Parallelism	10
4.0 QCFD: The Big Picture	11
4.1 Quantum Systems	11
4.2 Classical Systems	11
4.3 Computational Tools	12
II Overview of Methods for QC in Fluid Dynamics	13
5.0 Lattice Simulations	13
5.1 LQC 1: Quantum Lattice Gas Automaton and Phase Coherent Quantum Networks	13
5.2 LQC 2: Dirac Equation and the Pseudo Spin Boson system	16
5.3 LQC 3: Quantum to Classical Mapping	17
6.0 Continuum Simulations	17
6.1 Data Loading	18
6.2 Output Measurements	21
6.3 Quantum State Tomography	22
6.4 Data Processing	23
6.5 Quantum Fourier Transform	26
6.6 Quantum Turbulence	27
6.6.1 Quantum simulation	28
6.6.2 The two-fluid model	28
6.6.3 Gross-Pitaevskii model	29
6.6.4 Vortex filament model	29
6.6.5 HVBK model	29
7.0 Variational Solvers and Quantum Annealers	29
8.0 Quantum Programming and Machines	32
III Introducing <i>QuOn</i> - A Quantum Simulator and Quantum Linear System Algorithms	34

9.0 QuOn - A High Performance Quantum Simulator	34
9.1 The basic structure	34
9.2 Features	35
10.0 Quantum Linear System Algorithms and an Example	41
10.1 The HHL Algorithm	41
10.1.1 A simplified outline	41
10.1.2 The algorithm in a bit more detail	42
10.2 An Example: 2D Hele-Shaw Flow	43
10.2.1 Numerical procedure	45
10.2.2 Results	45
11.0 Discussion and Concluding Remarks	51
11.1 Summary	53

Part I

Introduction and the Big Picture of Quantum CFD

1.0 INTRODUCTION

Fluid mechanics as a field poses a vast array of interesting questions that relate to almost everything we see around us. Apart from theory and experiments, computational methods have greatly aided fluid mechanics research over the past few decades; indeed, with the growth in computers of increasingly higher computational power, fluid mechanical simulations have become highly realistic. But with increasing sophistication comes new generations of questions. For instance, even with the great advances seen in High Performance Computing (HPC), and despite the progress being made continually by very large Direct Numerical Simulations (DNS), one cannot say that long standing questions relating to the separation of scales in turbulence have been addressed fully. Without necessarily making the explicit case that computer technology development has hit obstacles, we simply note that the computational challenges being faced at present are so enormous that simply making supercomputers more powerful cannot catch up with the demands. Not only manufacturing smaller transistors face quantum effects, but also their integration into massively complex systems poses numerous challenges. To break this barrier, one needs a change of paradigm in computing. Enter quantum computing!

In quantum computing, we manipulate quantum systems to perform calculations and simulations. We are thus entering an era in which computations are becoming more “physical”. In fact, it was a dream of Richard Feynman [1] to simulate a quantum system by using another quantum system. We are now in the NISQ (Noisy Intermediate Scale Quantum) era [2], where we have quantum computers of sizes ranging from 50 qubits to a few hundreds of them. (Qubits are essentially the quantum analogue of classical bits and will be described later; it suffices to say here that their number characterizes the power and size of a quantum computer.) The word “noisy” indicates that quantum devices are still prone to errors from external and internal noises, and are not yet perfect. Yet, with quantum devices of the size just emerging, quantum computing (QC) can outperform many operations that current supercomputers strain to achieve. Quantum Computers have already started demonstrating their practicability in various fields such as finance strategies, medicine, quantum materials and chemical simulations, resource management, optimization and cryptography. What we wish to investigate here is its utility for performing Computational Fluid Dynamics (CFD) research.

In these lectures, which are in part derived from [3], presents an outlook on doing CFD quantum mechanically, which we term Quantum Computation of Fluid Dynamics (QCFD). It introduces and motivates researchers who wish to study fluid mechanics or dynamical systems, in general, to the new possibility of using quantum computers. Part 1 includes section 3, which presents a brief overview of QC and its differences from classical computing, and section 4, which provides the big picture of how fluid mechanics study can be viewed in the QC context. This is followed by Part 2 where one can find a description of methods that are lattice based (section 5) and continuum based (section 6). Section 6 also touches on the possibility of studying quantum turbulence and reviews existing methods and proposes newer directions. Sections 6 and 7 list, from the horde of QC algorithms, a few key ones that are deemed important for our purposes, and provide a few specific demonstrations. Section 8 briefly mentions the currently available commercial quantum machines and quantum programming tools. Part 3 includes the following topics: section 9 which introduces our own new and high performance Quantum Simulator called “QuOn”, followed by section 10, which briefly discusses Quantum Linear Systems Algorithm (QLSA) and its implementation using QuOn and Qiskit with a simple fluid flow example. Finally, these lectures end with section 11 on a few conclusions relevant to QCFD.

2.0 MOTIVATION

Several physical systems such as turbulent flows, glassy systems, protein folding and chemical reactions, are formidably hard to simulate “efficiently” on even the biggest supercomputers. However, with the growth in high performance computing (HPC) technology and equally “efficient” algorithms, computational fluid dynamics (CFD) in particular, has progressed immensely over the last several decades. Some state-of-the-art Direct Numerical Simulations (DNS) [4–6] of turbulent flows governed by the Navier-Stokes equations, simulate flows at overwhelmingly large Reynolds numbers (Re) with high resolutions. These simulations not only reveal the fine details of the flow physics [5, 6], but also test the limits of supercomputers they are run on. Let’s for instance refer to Figure 1, which shows how HPC, isotropic DNS simulations have evolved with time and increasing computational power, all the way from 32^3 to $\approx 18000^3$ grid point simulations (a-f) [7–12]. We see that it roughly scales with Moore’s law. However, even with the upcoming exa-scale machine upgrades, reaching significantly higher Reynolds number seems almost impossible in the near future.

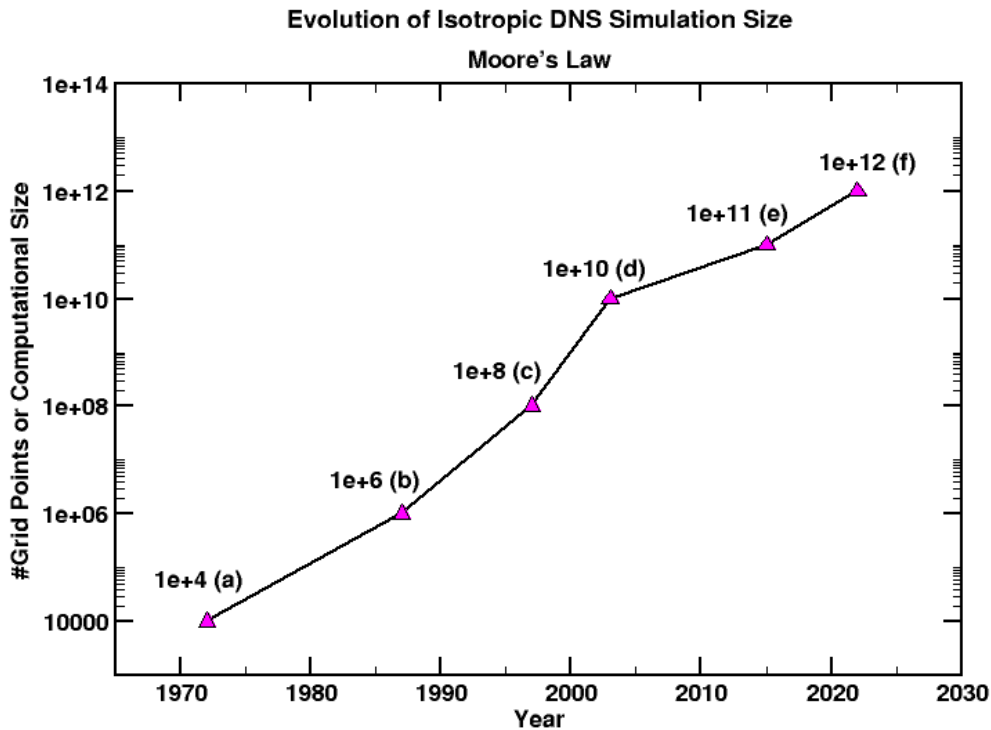


Figure 1: Evolution of Isotropic DNS simulations with time and computational power. The capacity scales as Moore’s Law.

For instance, to simulate flows with resolutions (or Re) high enough to settle unresolved questions, or to simulate physical systems with extremely high Re such as the Sun or a cyclone, one would need supercomputers that are several orders of magnitude higher in power than the current ones. Thus a paradigm shift into quantum computing might, after all, become necessary to attain such computational scales at exponentially faster rates as shown in Figure 2 (not to scale, representative only).

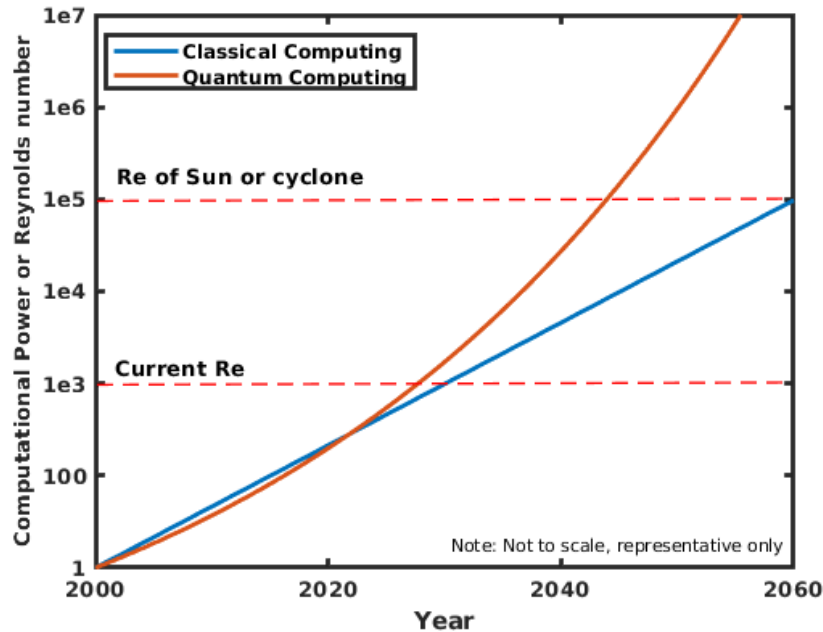


Figure 2: A schematic of quantum vs classical scaling of computational power or Reynolds number growth with time

3.0 QUANTUM COMPUTATION - SOME FUNDAMENTALS

The purpose of this section is to provide a brief overview of how the working rules of QC differ from those of classical computing. It is intended for readers with minimal background in quantum computing; a detailed account can be found in [13].

3.1 What is Quantum Computation?

It is a form of computation centered on quantum mechanics, manipulating information in the form of quantum bits called “qubits”, by designing appropriate “quantum algorithms” that comprise “quantum gates and circuits”, which, in turn, act on these qubits to yield the intended result. This sounds similar to classical computation, except that every word or phrase is prefixed by the word “quantum”. We shall explain each of these terms below.

3.2 Qubits

Qubits form the work horse of quantum computation. Similar to classical bits, quantum bits are objects that hold information describing quantum physical systems, which are eventually manipulated to perform a computational task. In reality, these qubits represent the state of an actual quantum physical system, governed by laws of quantum mechanics. Mathematically, it is given by the wavefunction Ψ , which completely encodes all the details describing the state of a quantum object. As a working example, a qubit could represent the two spin states of an electron. There exist several physical realizations of qubits such as Quantum Electro-Dynamic (QED) Optical Cavities, Ultra Cold atoms and Rydberg ions, Superconductors and Topological Materials (Majorana

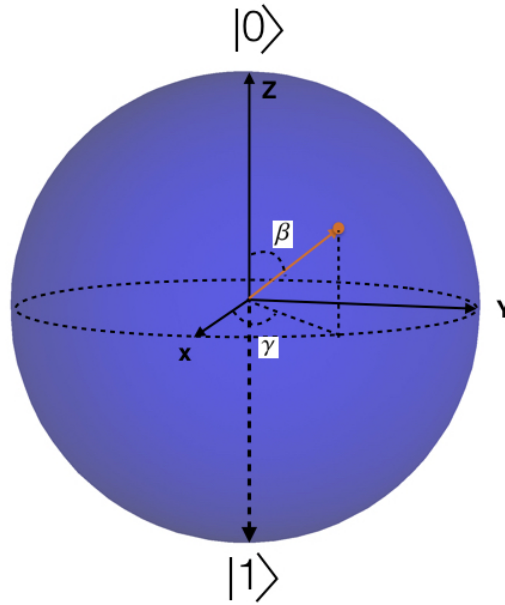


Figure 3: Bloch Sphere. Here α is the global phase, β is the polar angle and the azimuthal angle γ is the relative phase

fermions), Photons, Quantum dots, Nuclear Magnetic Resonance (NMR), etc. For the rest our discussion, however, we shall simply describe qubits as abstract mathematical objects. For now and for all practical purposes, we shall denote qubits as wavefunctions, which are vectors in a complex vector space called the Hilbert Space \mathbb{H} . In Dirac's bra-ket notation, it is represented as a "ket" vector $|\Psi\rangle$ in \mathbb{H} ($\in \mathbb{C}^n$)

$$|\psi\rangle = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}; c_i \in \mathbb{C} \quad (1)$$

while "bra", given by $\langle\Psi|$, is the vector dual $= |\Psi\rangle^\dagger$. These wavefunctions obey all the rules of a complex vector space. An obvious extension to this concept is to multiple qubits by taking tensor products of individual wavefunctions, which together lie in a tensor-product Hilbert space of corresponding wavefunctions: $|\Psi\rangle = (|\psi_1\rangle \otimes \dots \otimes |\psi_n\rangle) \in \mathbb{H}^{\otimes n}$. For instance, the two spin states (spin-up \uparrow and spin-down \downarrow) of an electron, could correspond to the state eigenvectors $|0\rangle$ and $|1\rangle$, respectively. The wavefunction of such two level or two state systems is a complex vector in $\mathbb{H} \in \mathbb{C}^{\otimes 2}$ which, when expressed mathematically as a linear combination of the basis vectors, has the form

$$|\Psi\rangle = c_1|0\rangle + c_2|1\rangle, \quad (2)$$

where

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (3)$$

Physically, the complex numbers c_1 and c_2 represent the *probability amplitudes* of the electron being in a given basis state, whose squares, $|c_1|^2$ and $|c_2|^2$, according to Born's principle, give the *probability* of the electron being in either state, $|0\rangle$ or $|1\rangle$. This only implies, at any given time, that the electron has a finite

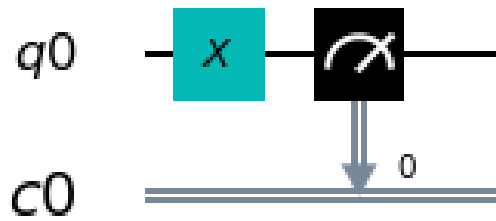


Figure 4: Circuit for a simple NOT operation

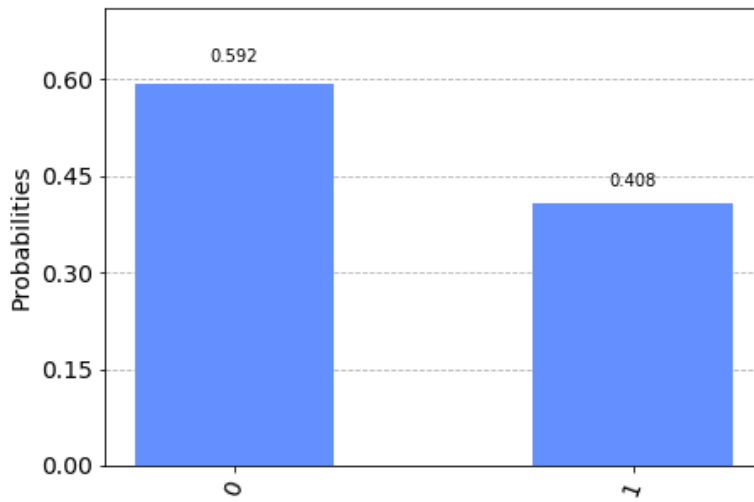


Figure 5: Probabilities before NOT operation

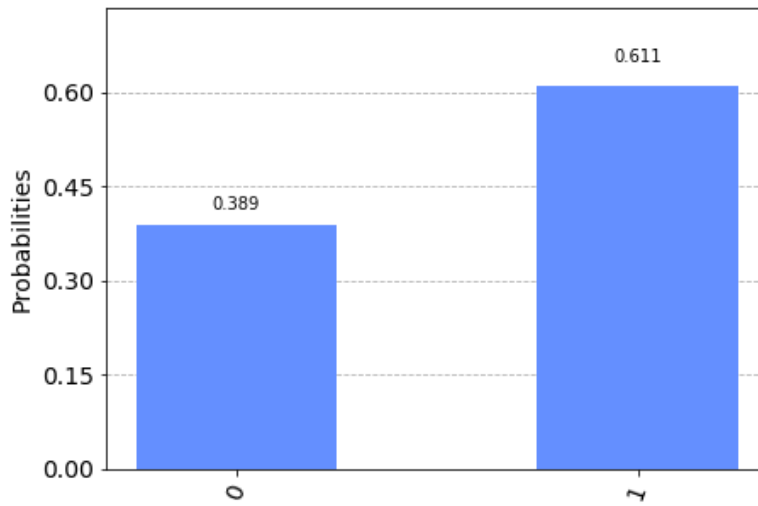







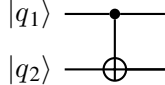
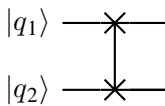
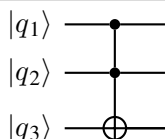
Figure 6: Probabilities after NOT operation

probability of being in both states simultaneously, unlike an unbiased coin. (Such superposition states sum to unity as probabilities should.) This is what distinguishes a classical state from a quantum one: a quantum state prior to measurement or observation can exist in a superposition of two different states, while a classical state can be in only one of them at a given time. This gives us access to multiple basis states simultaneously; this is the quantum parallelism which we shall examine subsequently. Apart from the Hilbert space representation, another useful visualisation of a qubit is the Bloch Sphere representation shown in Figure 3. Every possible state described by equation (2), can be represented as a vector on this unit sphere via the relation

$$|\Psi\rangle = e^{i\alpha} \left[\cos\left(\frac{\beta}{2}\right)|0\rangle + e^{i\gamma} \sin\left(\frac{\beta}{2}\right)|1\rangle \right], \quad (4)$$

where α is the global phase, γ the relative phase (azimuthal angle) and β the polar angle. All transformations and actions of “quantum gates” on qubits are to be regarded as rotational affine transformations on the Bloch vector.

Table 1: Quantum Logic Gates

Quantum Logic Gate	Circuit Symbol	Operation	
X (Pauli X)		$ 0\rangle \rightarrow 1\rangle$	$ 1\rangle \rightarrow 0\rangle$
Y (Pauli Y)		$ 0\rangle \rightarrow i 1\rangle$	$ 1\rangle \rightarrow -i 0\rangle$
Z (Pauli Z)		$ 0\rangle \rightarrow 0\rangle$	$ 1\rangle \rightarrow - 1\rangle$
H (Hadamard)		$ 0\rangle \rightarrow \frac{ 0\rangle+ 1\rangle}{\sqrt{2}}$	$ 1\rangle \rightarrow \frac{ 0\rangle- 1\rangle}{\sqrt{2}}$
R_ϕ (Phase Shift)		$ 0\rangle \rightarrow 0\rangle$	$ 1\rangle \rightarrow e^{i\phi} 1\rangle$
CNOT		$ q_1, q_2\rangle \rightarrow q_1, q_2 \oplus q_1\rangle$	
SWAP		$ q_1, q_2\rangle \rightarrow q_2, q_1\rangle$	
Toffoli		$ q_1, q_2, q_3\rangle \rightarrow q_1, q_2, q_3 \oplus q_1 \cdot q_2\rangle$	

3.3 Quantum Gates, Circuits and Algorithms

In analogy to classical computing, where we write algorithms to manipulate information, and accomplish them fundamentally via logic gates such as AND, NOT, OR, NAND and Toffoli gates, information manipulation is accomplished by quantum algorithms using quantum logic gates and circuits, as explained below.

Quantum gates are fundamentally unitary operators U ($UU^\dagger = \mathbb{I}$), which cause affine rotational transformations on qubits. These unitaries are linear and reversible operations (unlike classical gates such as NOT), and

are also norm-preserving. It is obvious that an infinitely many such unitary transforms can exist but, among them, the fundamental and important ones are listed in Table 1. As in classical computing there are a set of quantum gates which are universal, and a detailed description can be found in [13].

Now, a combination of such gates forms a quantum circuit. For instance, consider the quantum version of the classical NOT gate acting on a qubit. This is the X gate, given by σ_x , the Pauli operator. To see this in action, let us prepare a single qubit (q0) in the state $|\psi\rangle = \sqrt{0.6}|0\rangle + \sqrt{0.4}|1\rangle$ and then apply the

X gate = $\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. This yields

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{5}$$

That is, this gate just flips the state $|0\rangle \rightarrow |1\rangle$ and $|1\rangle \rightarrow |0\rangle$. After operating this gate, we measure the probabilities ($|c_1|^2$ and $|c_2|^2$) associated with the basis states and store them in a classical register (c0), as shown in Figure 4. The horizontal lines or circuit wires represent the time evolution of a qubit, and the double lines represent a classical bit. The meter symbol represents a measurement operation in the computational basis, while the X symbol denotes the quantum NOT gate.

This circuit is now run on IBMQ Qiskit quantum simulator platform; from the results shown in Figures 5 and 6, it is clear that the states have flipped. It is worth noting that the NOT operation is applied on both $|0\rangle$ and $|1\rangle$ simultaneously, i.e., we now have the new state $|\psi'\rangle = \sqrt{0.4}|0\rangle + \sqrt{0.6}|1\rangle$. We shall explore the associated notion of quantum parallelism further but note here simply that a quantum circuit is basically a construction of quantum gates and wires that together act on a given set of qubits and perform the desired transformation, while a quantum algorithm is a collection of linked quantum circuits that performs a computational task. We shall outline the important algorithms in the sections to follow.

3.4 Quantum Parallelism

With a simple block diagram, we shall briefly outline quantum parallelism and the subtle difference between quantum and classical computing.

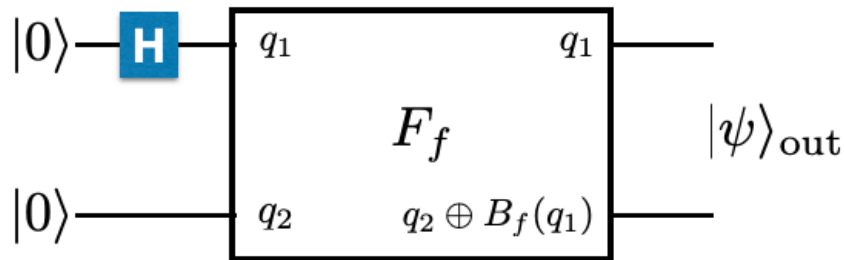


Figure 7: Quantum Parallelism

Consider a Boolean function, $B_f: \{0,1\} \mapsto \{0,1\}$. We wish to evaluate this function for both 0 and 1 via quantum processing. For this, we take a 2-qubit state $|\psi_{in}\rangle = |q_1 q_2\rangle$, where $q_1, q_2 \in \{0, 1\}$. To keep our discussion brief, let us accept the existence of an oracle function (black-box) F_f that basically performs $|q_1, q_2\rangle \xrightarrow{F_f} |q_1, q_2 \oplus B_f(q_1)\rangle$, where \oplus represents modulo 2 addition (refer to [13] for the details of the black-box). To compute both $B_f(0)$ and $B_f(1)$ classically, we would have to do the computation twice. Now let us

look at the quantum circuit in Figure 7, whose action is as follows: First, a Hadamard gate is applied on the first qubit: $|0\rangle \otimes |0\rangle \xrightarrow{H \otimes I} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle$. This state now forms the input to the black-box, which finally produces the output state by applying F_f , as

$$|\Psi\rangle_{out} = \frac{|0, B_f(0)\rangle + |1, B_f(1)\rangle}{\sqrt{2}}. \quad (6)$$

Now, we have evaluated in just one shot both $B_f(0)$ and $B_f(1)$. This is quantum parallelism. The concept can be extended to more qubits, and also be used to extract information about some global properties of the function B_f , so as to verify whether or not a given function is a constant; algorithms such as Deustch-Josza and Simon's algorithm can do that and more [13]. This inherent parallelization of QC at the physical level demonstrates one of the many subtle and fundamental differences that sets QC apart from classical computations. In the sections to follow, we shall look at algorithms, gradually narrowing our scope to a discussion of fluid mechanics.

4.0 QCFD: THE BIG PICTURE

We now attempt to address the task at hand: Analyzing the possible utility and advantages of quantum computing to study physical systems, fluid mechanical in particular. To this end, a slight digression towards a broader picture of QC study, shown in Figure 8, is useful for classifying the problems and methods for fluid mechanics. There are primarily three possible sets of problems which could be addressed by QC: (1) quantum systems, (2) classical systems and (3) quantum algorithms. Each of them is described below.

4.1 Quantum Systems

Quantum systems are obvious candidates for using QC. Though all quantum systems are legitimate candidates, problems that are currently being explored, or could be explored, fall in two categories:

(a) *Lattice based systems*: Most hard quantum condensed matter systems such as the Hubbard problem or the quantum lattice gas fall in this category. Here, one can look at the lattice based Hamiltonians to either perform a quantum simulation or compute observables and properties via specific algorithms.

(b) *Continuum problems*: On the other hand, some problems such as quantum turbulence and quantum liquids, would require the integration of the many-body Schrödinger equation followed by a mapping to macroscopic observables. One could also use quantum algorithmic numerical tools to integrate model equations such as Gross Pitaevski equations (in the case of quantum turbulence) or do a quantum Monte Carlo study, etc.

4.2 Classical Systems

A slightly harder but an interesting avenue would be to compute classical systems using quantum computing. Most of the effort here would be spent in translating classical dynamics into the quantum language. One can then harness the quantum advantage from here on. Once again, a similar classification could be done, where one looks at lattice systems, Lattice Boltzmann Methods and molecular dynamics, or one can start using quantum based mathematical tools such as ODE solvers and eigenvalue solvers or optimization methods for integrating and solving classical governing equations such as the Navier-Stokes equations.

4.3 Computational Tools

Though the development of quantum algorithms would need new mathematical tools, this step can proceed independently up to a certain point. Here one would be interested primarily in developing, quantum mechanically, the numerical solvers or methods available on classical machines, such as optimization, ODE/PDE solvers, factorizations, data search, eigenvalue solvers, etc.

With this background, we now proceed to examine each of these methods and provide real quantum computational demonstrations. From these methods, we shall focus primarily on two methods suitable for studying fluid dynamics: (1) lattice based methods, and (2) continuum quantum simulations and quantum algorithms.

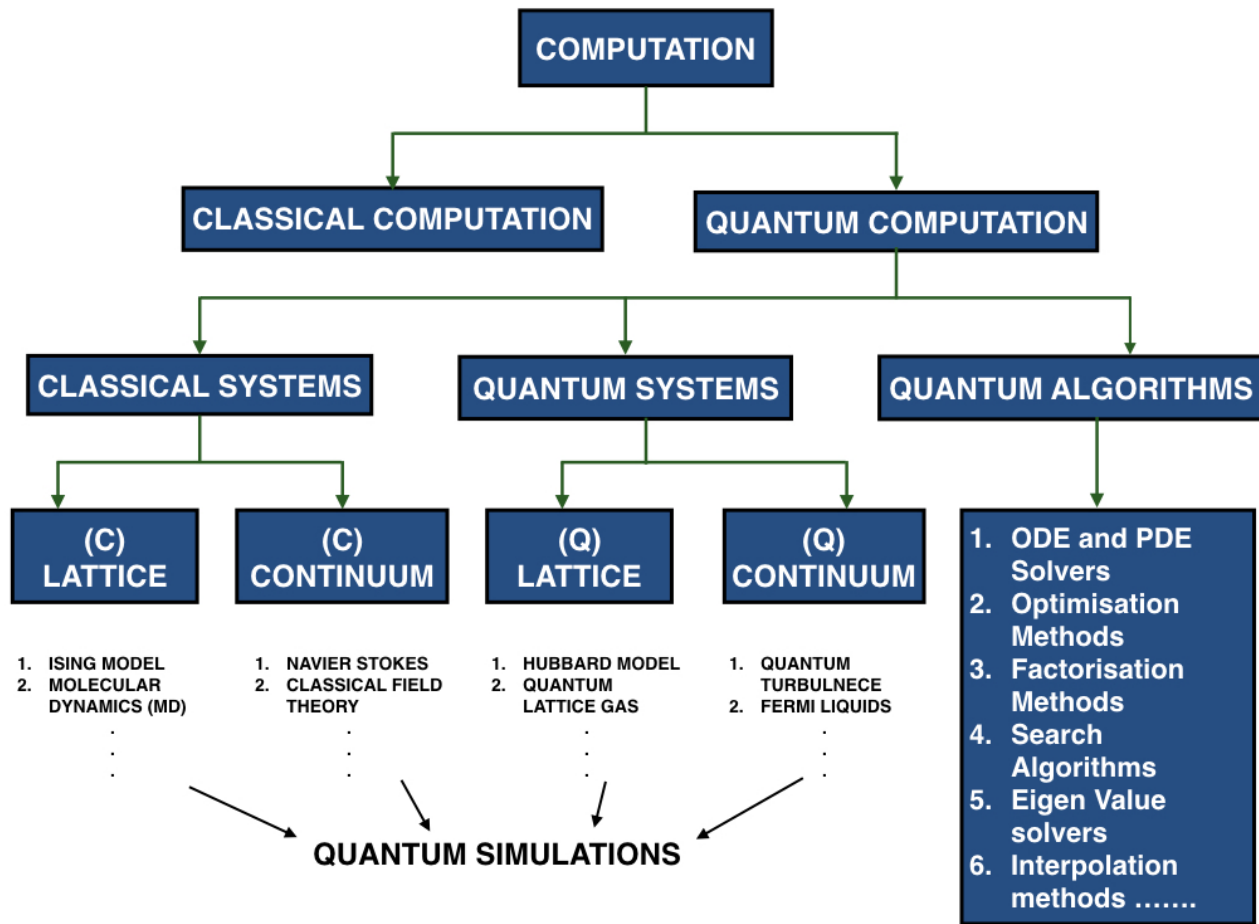


Figure 8: Classification of problems based on QC methods

Part II

Overview of Methods for QC in Fluid Dynamics

5.0 LATTICE SIMULATIONS

In addition to the popular computational methods such as (DNS) [5, 7, 14–16], Large Eddy Simulations (LES) [15, 17, 18], Reynolds Averaged Navier-Stokes (RANS) [15, 19] and other modelling techniques, the Lattice Boltzmann Method (LBM) [20, 21] has also been used recently to model fluid dynamical problems. The underlying principle governing LBM stems from the classical Boltzmann kinetic transport mechanism, which models the fluid as an ensemble of a large number of fictitious “fluid particles” placed on a uniform lattice. These fluid particles advect in some allowed velocity directions and collide with each other resulting in a scattering-relaxation type process, which results to a net momentum transfer, as shown in Figure 9. The main advantage of this model is the large reduction in the number of degrees of freedom with which one would otherwise have to deal in the continuum case. The basic LBM equation is

$$\underbrace{(\partial_t + \mathbf{v}_\alpha \cdot \nabla)}_{\text{Advection}} \rho_\alpha(\mathbf{r}, t) = \underbrace{S_{\alpha\beta}(\rho_\beta^{eq}(\mathbf{r}, t) - \rho_\beta(\mathbf{r}, t))}_{\text{Scattering-Relaxation}}. \quad (7)$$

where \mathbf{v} is the velocity, ρ is the mass density, S is the scattering matrix and ρ^{eq} is the equilibrium distribution of the mass density (for a detailed review refer to [21]). One could naively say that, since quantum mechanical problems inherently deal with quantized “particles”, problems that involve particle tracking (e.g., discrete Lagrangian dynamics), would be a good method for the application of QC to fluid dynamics.

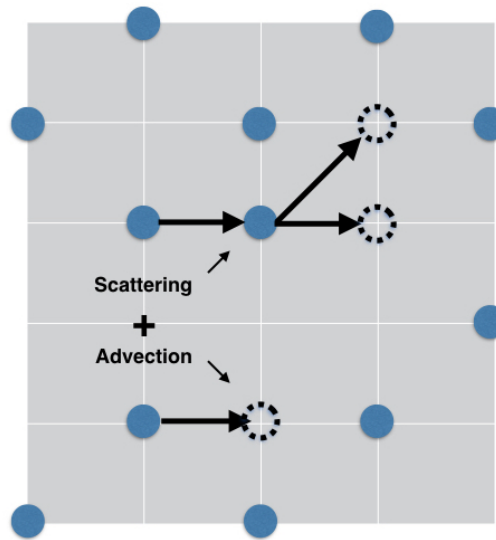


Figure 9: Schematic of an LBM simulation

5.1 LQC 1: Quantum Lattice Gas Automaton and Phase Coherent Quantum Networks

This method provides insights into *one* of the physical frameworks for generating a map from classical fluid dynamics to QC. As an aside, though this method has been proposed quite a while ago, only theoretical and

classical computer simulations of the QC method had been done (due to the absence of a real QC at the time), so the implementation on presently available QCs remains to be established. The key idea here is derived from Quantum Lattice Gas Automaton (QLGA) [22, 23], which is a quantum extension of the classical lattice gas system.

As a simple illustration let us consider a 1D lattice system. The classical lattice gas tags every particle with instantaneous positions and velocities $(\mathbf{x}, \mathbf{v})_i$, where the velocities \mathbf{v}_i at every lattice site points either to the left or the right. With this scheme, we generate an ensemble of 1D state configurations, which evolves according to a local evolution map. This mapping, like the LBM in Equation 7, is a combination of advection and scattering processes.

Now its quantum counterpart, the QLGA, prepares quantum superpositions of the classical states. For a single particle 1D lattice of length N, this results in each site having 2 pseudo occupation slots (q), corresponding to the left (l) and right (r) streaming particles with associated probabilities. This means that we now have a 2 qubit system $|\psi\rangle_i = \alpha|q_1q_2\rangle_i + \beta|q_1q_2\rangle_i$ and $q_1, q_2 \in \{l, r\}$ sitting on each site, hopping to adjacent sites with a basis set $\{|lr\rangle, |ll\rangle, |rr\rangle, |rl\rangle\}$. The scattering processes of these qubits is given by the scattering matrix \hat{S} that captures the interactions, while the advection is given by \hat{A} . If L and R are left and right scattering probability amplitudes (i.e., the probability that a particle travelling left as it enters a site continues leftwards, etc), the scattering matrices for the cases of one and two qubits (for some p) would be given by

$$S_1 = \begin{pmatrix} L & R \\ R & L \end{pmatrix} = \begin{pmatrix} \cos p & i \sin p \\ i \sin p & \cos p \end{pmatrix} \quad (8)$$

$$S_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & L & R & 0 \\ 0 & R & L & 0 \\ 0 & 0 & 0 & \theta \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos p & i \sin p & 0 \\ 0 & i \sin p & \cos p & 0 \\ 0 & 0 & 0 & \theta \end{pmatrix}, \quad (9)$$

where $|L|^2 + |R|^2 = 1 = |\theta|^2$ and θ represents the relevant multi-particle scattering events of the delta-function type. Thus the dynamics of the advection and scattering steps could be summarised quantitatively as follows, with the propagation of the left and right travelling wavefunctions:

$$\Psi(\mathbf{r}, t + 1; \rightarrow) = L\Psi(\mathbf{r} - 1, t; \rightarrow) + R\Psi(\mathbf{r} + 1, t; \leftarrow) \quad (10)$$

$$\Psi(\mathbf{r}, t + 1; \leftarrow) = L\Psi(\mathbf{r} + 1, t; \leftarrow) + R\Psi(\mathbf{r} - 1, t; \rightarrow). \quad (11)$$

Here, L and R scale as p, and the evolution of such a process is unitary and preserves the norm. The sum of these two wavefunctions satisfies the Schrödinger equation. Interestingly we can recover both the Dirac and the Schrödinger equation with limits $\delta\mathbf{r} \rightarrow 0$ and $\delta\mathbf{r}^2 \rightarrow 0$ as $\delta t \rightarrow 0$, which is done using the standard Chapman-Enskog asymptotic closure [22, 24, 25].

A fact worth mentioning is that the quantum effect of the scattering operator causes a local entanglement in a specific lattice zone radius, while the advection operator acts globally, causing superposition of configuration states as well as a global entanglement. The QLGA setup was originally used in this form to perform quantum many body system simulations [24–28], and was later modified into what is known as the Phase Coherent Quantum Lattice Network to compute mesoscopic and macroscopic fluid dynamics, and to study quantum mechanically the diffusion equation and Burgers flow [29–32].

These amended versions naturally evolved in response to the need for minimizing impediments such as errors due to large entanglement and noises that depend on the environment. The main difference in these extensions is that, instead of the qubits representing state superpositions, they directly replace the classical bits

that store site information in classical LBM. For instance, in a model with 4 qubits per site, with each lattice site having 4 nearest neighbors, one can encode 2^4 complex numbers per site. Thus, every site now acts as a small 4 qubit QC and many such QCs can be connected via a lattice network to form a coherent quantum network. In general, for an N -site lattice with N_q qubits per lattice (also the number of nearest neighbors), we have a lattice QC made of a total of $N_T = N * N_q$ qubits. We would thus have a 2^{N_T} -dimensional Hilbert space composed of 2^{N_q} dimensional submanifolds corresponding to local site-specific Hilbert spaces and, with \mathbf{C} representing the coefficient matrix and q_i 's being the qubits on every site, the total wavefunction would be

$$\Psi(\mathbf{r}_1, \dots, \mathbf{r}_N, t) = \sum_{\psi(\mathbf{r}_i, t)} \mathbf{C} \times (\psi(\mathbf{r}_1, t) \otimes \dots \otimes \psi(\mathbf{r}_N, t)) \quad (12)$$

$$= \sum_{\psi_{q_i}} \tilde{\mathbf{C}} \times (\psi_{q_1} \otimes \dots \otimes \psi_{q_{N_T}}). \quad (13)$$

The evolution operator for this lattice gas simulation is obtained by integrating the corresponding Schrödinger equation and, as explained above, this unitary evolution operator would now correspond to the product of the unitary advection (\hat{A}) and scattering (\hat{S}) matrices, giving us

$$\Psi(\mathbf{r}, t + \Delta t) = e^{i\hat{H}\Delta t/\hbar} \Psi(\mathbf{r}, t) = \hat{A}\hat{S}\Psi(\mathbf{r}, t). \quad (14)$$

We might now ask which of the mesoscopic fluid dynamical observables can be computed. For this, let us take a Bravais Lattice with lattice site positions given by \mathbf{R} , while the corresponding unit vectors \mathbf{r} and wavefunctions are given by the propagating Bloch vectors with a lattice-specific periodicity (i.e., we can reach any site from any other site by advancing through an integral multiple of the lattice periodicity, yielding a total of steps to be $k_i \leq N * (\text{period})$). Recasting a lesson from the classical lattice gas, we can now compute the occupancy probability, as well as mass and momentum densities [30] as follows.

(a) The occupancy probability for the quantum case is straightforward and is just the average of the number operator $\hat{n}_{\mathbf{R}} = \hat{c}_{\mathbf{R}}^\dagger \hat{c}_{\mathbf{R}}$. This can be measured on many practical QCs like NMR-QC (nuclear magnetic resonance) by Quantum State Metrology (which we shall describe in the sections to follow). From the basic postulate of quantum mechanics, the average is given by the trace of the operator taken with the lattice's density matrix, thus giving the occupation probability to be

$$P_{\mathbf{R}, t} = \text{Tr}[(|\Psi(\mathbf{R}, t)\rangle\langle\Psi(\mathbf{R}, t)|)\hat{n}_{\mathbf{R}}] = \text{Tr}[\rho(t)\hat{n}_{\mathbf{R}}]. \quad (15)$$

(b) If d is the lattice spacing, we can directly write down the mass and momentum densities (ξ and $\xi\mathbf{v}$) as

$$\xi(\mathbf{r}, t) = \lim_{d \rightarrow 0} \sum_{k_i, \mathbf{R}_1}^{\mathbf{R}_N} m \text{Tr}[\rho(t)\hat{n}_{\mathbf{R}_{k_i}}], \quad (16)$$

$$\xi(\mathbf{r}, t)\mathbf{v}(\mathbf{r}, t) = \lim_{d \rightarrow 0} \sum_{k_i, \mathbf{R}_1}^{\mathbf{R}_N} m v^2 \mathbf{r}_{(\mathbf{R}_{\text{mod}N_q})} \text{Tr}[\rho(t)\hat{n}_{\mathbf{R}_{k_i}}]. \quad (17)$$

Now, as the limit reaches the continuum (i.e., with higher lattice resolution), we could use quantum observables such as the number operator to estimate mesoscopic quantities. Finally, one can also write the mesoscopic transport equation as

$$P_{\mathbf{R}+d\mathbf{r}, t+\Delta t} = P_{\mathbf{R}, t} + \langle\Psi(\mathbf{R}, t)|\hat{S}^\dagger \hat{n}_{\mathbf{R}} \hat{S} - \hat{n}_{\mathbf{R}}|\Psi(\mathbf{R}, t)\rangle, \quad (18)$$

since

$$\langle \Psi(\mathbf{R}, t) | \hat{S}^\dagger \hat{\mathbf{n}}_{\mathbf{R}} \hat{A}^\dagger | \Psi(\mathbf{R}, t + \Delta t) \rangle = \langle \Psi(\mathbf{R}, t) | \hat{S}^\dagger \hat{\mathbf{n}}_{\mathbf{R}} \hat{A} | \Psi(\mathbf{R}, t) \rangle.$$

These equations and operators can be simulated by appropriately recasting them in terms of generalized U_3 gates (IBMQ) given by 3 parametric unitary gates, which are the Euler angles [30, 33]:

$$U_3 = \begin{pmatrix} \cos(\frac{\theta}{2}) & e^{-i\lambda} \sin(\frac{\theta}{2}) \\ e^{i\phi} \sin(\frac{\theta}{2}) & e^{i(\phi+\lambda)} \cos(\frac{\theta}{2}) \end{pmatrix}. \quad (19)$$

Here $0 \leq (\phi, \lambda) < 2\pi$ and $0 \leq \theta \leq \pi$. Thus, the takeaways of this method are the following: (1) it provides an understanding of the translation of classical LBM calculations to its quantum analog, thus enabling newer methods for building QC circuits to simulate fluid dynamics; (2) it gives an idea of how one could make use of quantum lattice properties and entanglement to map it to macroscopic properties of the flow; and (3) though a clear estimate on scaling behavior, compared to classical LBM, is an open avenue, it seems obvious that one can gain over classical LBM, in both space and time complexity of the problem, since we are using state space configurations in terms of quantum superpositions and are performing simultaneous evolutions of these states. Also we draw the reader's attention to a more recent quantum algorithm for a collisionless Boltzmann equation which offers insights into possible circuit implementations of such lattice methods[34]. We shall also examine a more recent variant of this method that is more amenable to implementation and expected to utilize exponential speedup due to better way of quantum superposition.

5.2 LQC 2: Dirac Equation and the Pseudo Spin Boson system

A variant procedure is the construction of a map from the classical LBM to the Dirac equation [20, 35, 36]. We shall not dwell on details of this method but provide only an outline of the fluid dynamical aspects. This method generates a map, using what is known as a coupled pseudo-spin bosonic system, which is amenable for implementation on a trapped ion QC or a superconducting QC. We first note that we can translate Equation (7) into a ‘‘Majorana type Dirac equation’’ of the form [37]

$$i \frac{\partial \Psi}{\partial t} + i \hat{\mathbf{A}}_{pb} \nabla_{pb} \Psi = \mathbf{M} \Psi, \quad (20)$$

where $\hat{\mathbf{A}}$ is the quantum analog of the advection matrix (a Clifford operator—since it is given by Pauli matrices), while \mathbf{M} is the quantum representation of the mass term, which is Hermitian. Since we know that Clifford operators (isomorphic in \mathbb{R}^3 with Pauli operators) do not simultaneously commute, we will need to have diagonal advection operators and symmetric and imaginary scattering matrices. The idea is essentially to map the mass density to a corresponding probability density of a wavefunction that is embedded in an appropriate ‘‘Fock space’’ of the given bosonic modes. For instance, in the 1D case, we would have a single bosonic wavefunction distribution as $|\Psi(x, t)\rangle_i = \int dx P(x) |x\rangle_i$, where $P(x)$ holds the information of the mass distribution. (For a higher dimensional Fock space with 2 bosonic modes in a 2D lattice, $|\Psi(\mathbf{x}, t)\rangle_i = \int dx_1 dx_2 P(x_1, x_2) |x_1\rangle |x_2\rangle$.) For producing different such distributions, we need to have an external parametrized knob that can control them. This task is accomplished by what is known as the ‘‘pseudo spins’’, which are coupled to the bosonic modes as $|\Psi\rangle = \sum_i P_i |s\rangle_i \otimes |\Psi(\mathbf{x}, t)\rangle$. The operators to diagonalize the advection matrix would, in the second quantized Dirac picture, be the following [36]:

$$e^{-i\hat{\mathbf{A}}\Delta t/\hbar} = \exp\left(-\frac{i\Delta t}{\hbar} \frac{\pi}{2\sqrt{2}} (\mathbf{Z}_1 \otimes I_2 + \mathbf{X}_1 \otimes \sigma_2^{pb})\right). \quad (21)$$

Here Z_i and X_i are the usual Pauli operators on i^{th} mode and σ^{pb} refers to the pseudo spin. On the other hand, the scattering operator \hat{S} , which is essentially a non-unitary type evolution step, is made “pseudo-unitary” by making its evolution dependent on an ancillary qubit, which acts as the control (refer to [36] for detailed methodology); finally, this \hat{S} is decomposed into a weighted sum of two unitary operators as $e^{-i\hat{S}\Delta t/\hbar} = e^{\hat{U}_1 + \alpha\hat{U}_2}$. Successive application of these operators on the initial lattice wavefunction can be done by a standard and useful trick of decomposing the evolution operators via a suitable Lie-Trotter-Suzuki decomposition, which allows one to translate such operations as a quantum circuit using generalised U_3 -type gates. Importantly, resources needed to do this operation is a polynomial in the degrees of freedom, while being sub-polynomial in error. Finally, we add suitable quantum metrology to extract the final state. The entire process has been illustrated for a simple advection-diffusion equation in [36]. Also, since this method, unlike previous ones, replaces classical bits directly by qubits, the pseudo-spin system that is used manifests quantum superposition, thus allowing one to exploit the exponential speed up of the superposition principle of QC; it would certainly be interesting to validate this expectation on present QCs.

5.3 LQC 3: Quantum to Classical Mapping

This method has often been used in early theoretical calculations by performing a mapping from a d -dimensional partition function for quantum systems to a $(d+1)$ -dimensional partition function for classical systems. In condensed matter systems, this is accomplished by mapping classical to quantum lattices by means of classical Monte-Carlo methods [38, 39]; in conformal field theories, this is achieved by understanding gravitational bulk-boundary correspondence [40, 41], etc. Though we will not describe details, we believe that this has the potential for hybridizing either LQC 1 or LQC2 along with quantum-to-classical mapping for solving fluid dynamical problems.

In this method, one basically computes a quantum partition function $\text{Tr}(e^{-H\beta})$ from the Feynman imaginary time path integral

$$Z = \sum_{r, r_i} \langle r | e^{-H\Delta\beta/N} | r_1 \rangle \langle r_1 | \dots | r_N \rangle \langle r_N | e^{-H\Delta\beta/N} | r \rangle, \quad (22)$$

which is exactly the classical $(d+1)$ -dimensional partition function, except that the extra dimension is replaced by space instead of time; it can be solved using the conventional transfer matrix type calculation. Now if one sets up a lattice-style fluid dynamics problem in 2D, it would be equivalent to solving a 1D quantum lattice problem with this map. (There also have been efforts to map d dimensions $\rightarrow d$ dimensions for purposes of understanding classical statistical mechanics in quantum mechanical terms [42].) This is now being actively applied to connect classically simulated annealing to quantum annealing methods to solve optimization problems. In fact, we shall discuss them separately to see how machines based on quantum annealing are being used to solve fluid dynamics problems.

6.0 CONTINUUM SIMULATIONS

Simulations done in a true continuum sense (i.e., no lattice models) ultimately boil down to preparing a mix of computational tools or algorithms that can emulate standard mathematical methods that solve the governing equations (PDEs and ODEs), quantum mechanically. To this end, we draw the reader’s attention to some currently implementable quantum algorithms and circuits that one could use in fluid dynamics.

When we refer to an implementable quantum algorithm, we mean quantum circuits that can be constructed from known coherent set of quantum logic gates and measurement processes. A complete computational process or simulation of a fluid dynamic problem involves three essential steps: (1) Initial data input or loading;

(2) Processing and generating new data; and (3) Reading the processed information to obtain results. Each of these steps, though obvious, involves nontrivial operations in QC. We will now take a closer look at them now.

6.1 Data Loading

The data inputs could be user-defined computational parameters or initial conditions of an ODE/PDE integrator, etc. Classically we input and store data, for instance in C++, by writing algorithmically `int a = 10;`, so that `a` holds the value 10. At the machine level, the data are “written” as a magnetic inscription of local magnetic polarities on a hard disk. We now ask how we could do the same exercise on a QC, i.e., store the value 10 in `a`. The storing of such a qubit at machine level comes in a variety of ways mentioned earlier. Here, we shall not explain how these physical realisations work, but dwell more on the algorithmic level, considering the machine level as an existent oracle. Though there are a variety of specific methods, in brief, we can load a classical bit of information onto a quantum computer in two ways:

(1) Amplitude loading: This method, also known as state preparation or the initialization method [13, 43], where one initializes a specific qubit state with the user-defined complex probability amplitudes of quantum superposition. That is, the input data are loaded in the form of complex probability amplitudes of wavefunctions. One of the algorithms followed for instance on practical QCs (like IBMQ) is outlined in Algorithm 1 [44], though one can always come up with custom circuits to construct a given state (we will detail another procedure, along with implementation, in Section ??). In this method, called the recursive quantum multiplexer algorithm, one starts with the required state and designs a circuit to transform the required state to all 0s; thus, the inverse circuit prepares our required state starting from all 0 states. As a demonstration, suppose we want to load four complex values such as (i) , $(2 + i)$, $\sqrt{2}i$, 1 (in this order). Since we can store the N values in $\log_2 N$ qubits, we need a 2 qubit system that looks like the following:

$$|\Psi\rangle = \frac{1}{3} (i|00\rangle + (2 + i)|01\rangle + \sqrt{2}i|10\rangle + 1|11\rangle). \tag{23}$$

The required set of unitary operations, obtained by following Algorithm 1 [44], is represented in the quantum circuit shown in Figure 10. Thus the probabilities of these states are essentially the magnitudes of these states (i.e. $\frac{1}{9}$, $\frac{5}{9}$, $\frac{2}{9}$ and $\frac{1}{9}$). Upon performing state tomography (see later), the final state when computed with Qiskit—the IBM Quantum Experience platform—is shown in Figure 11, which almost exactly matches the required state. However, most state initialization procedures

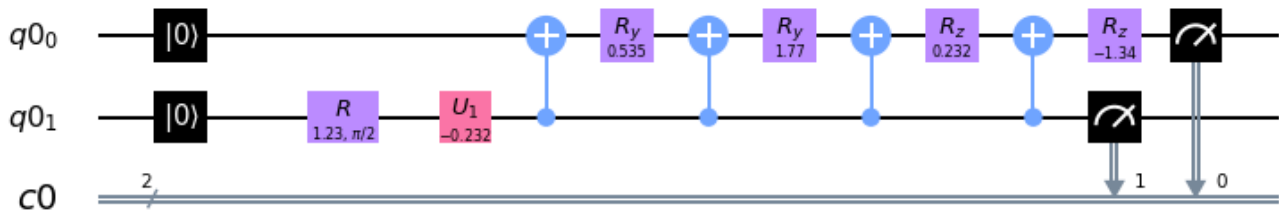


Figure 10: Circuit for amplitude loading 4 complex numbers

(2) State loading: Instead of using complex amplitudes, one may store the data directly as the state ket vectors. Suppose we want to load the decimal number 10, whose binary form is 1010. By state loading, we mean that there is some state that looks like $|\psi\rangle = |\phi\rangle \otimes (|1\rangle + |0\rangle + |1\rangle + |0\rangle)$. To do this, since we need 4 basis states,

Algorithm 1: Amplitude Loading: Quantum State Preparation

Input: $c_1, c_2 \dots c_n$
Output: $U_{load}, |\Psi\rangle_{final} = c_1|00\dots0\rangle + c_2|00\dots1\rangle + \dots + c_n|11\dots1\rangle$

```

1 Function AMPLITUDELOAD ( $c_1, c_2 \dots c_n$ ) :
2    $U = \mathbb{I}$ 
3    $|\Psi\rangle_{final} = |00\dots0\rangle$  (On Quantum Device)
4    $|\Psi\rangle_{temp} \leftarrow c_1|00\dots0\rangle + c_2|00\dots1\rangle + \dots + c_n|11\dots1\rangle$  (On Classical Device)
5   while ( $|\Psi\rangle_{temp} \neq |00\dots0\rangle$ ) do
6      $|\Psi\rangle_{temp} = \text{DISENTANGLE}(|\Psi\rangle_{temp})$  Least Significant Bit
7     Choose  $\alpha_i, \beta_i$  and global phase  $\phi$ , such that:
8     while ( $|q_i\rangle \leftarrow c_i e^{i\phi}|0\rangle$ ) do
9        $U_z = \mathbf{R}_z(\beta_i)|q_i\rangle$ 
10       $U_y = \mathbf{R}_y(\alpha_i)|q_i\rangle$ 
11       $(U_{zy})_i \leftarrow U_y \otimes U_z$ 
12       $U \leftarrow (U_{zy})_i \otimes U$ 
13   Now  $U = (U_{zy})_n \otimes \dots \otimes (U_{zy})_1$ 
14    $U_{load} = U^\dagger$ 
15    $|\Psi\rangle_{final} = U_{load}|\Psi\rangle_{temp} = c_1|00\dots0\rangle + c_2|00\dots1\rangle + \dots + c_n|11\dots1\rangle$ 
16   return  $U_{load}, |\Psi\rangle_{final}$ 

17 Function DISENTANGLE ( $|\Psi\rangle_{temp}$ ) :
18    $|\Psi\rangle_{local} = |\Psi\rangle_{temp} \leftarrow c_1|00\dots q_1\rangle + c_2|00\dots q_2\rangle + \dots + c_n|11\dots q_n\rangle$ 
19   return  $|\Psi\rangle_{local} \leftarrow c_1|00\dots\rangle \otimes |q_1\rangle + c_2|00\dots\rangle \otimes |q_2\rangle + \dots + c_n|11\dots\rangle \otimes |q_n\rangle$ 

```

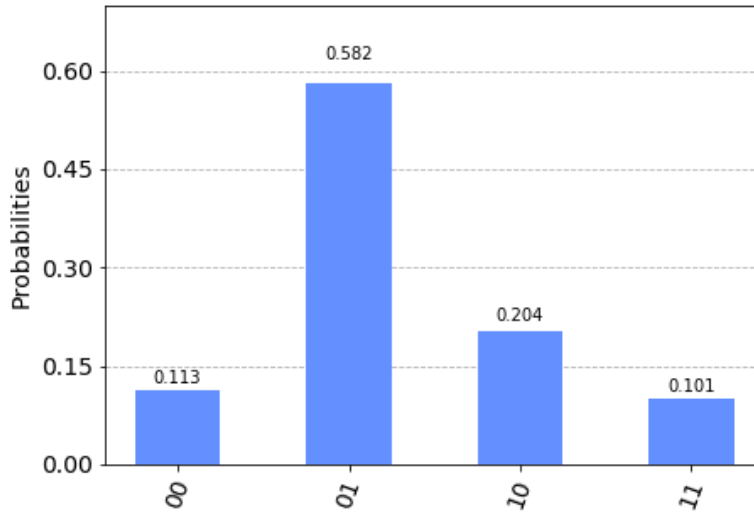


Figure 11: Required state prepared on IBMQ

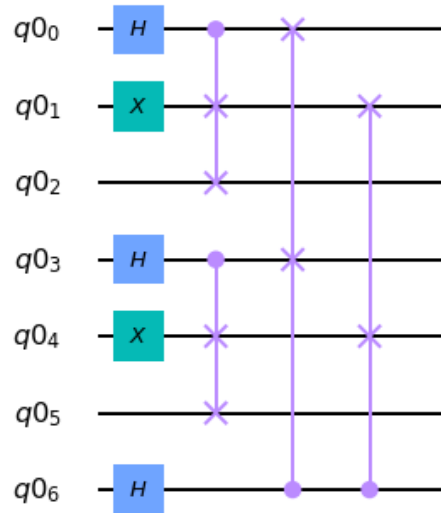


Figure 12: Circuit for *state loading* 4 classical bits

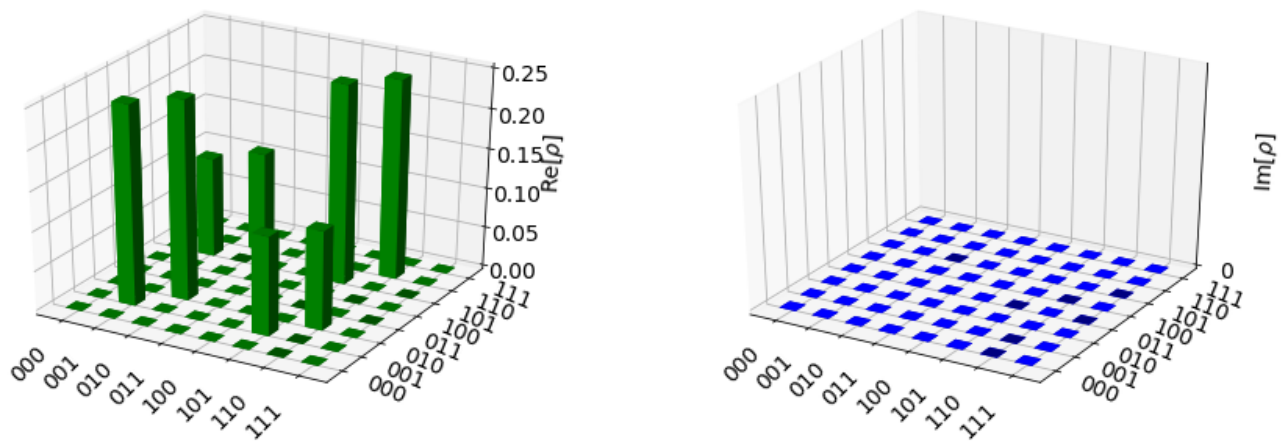


Figure 13: The density matrix of the data loaded 3 qubit state

one has to construct a state such as $|\psi\rangle = (|0\rangle|0\rangle \otimes |1\rangle + |0\rangle|1\rangle \otimes |0\rangle + |1\rangle|0\rangle \otimes |1\rangle + |1\rangle|1\rangle \otimes |0\rangle)$, where we see that the third qubit holds the value we need, 1010. To prepare such a state, we use a combination of Controlled-SWAP, X, H and Toffoli gates as shown in Figure 12. (A detailed explanation of such circuit designs is found at [45].)

Figure 13 shows the density matrix $\rho = \sum_i p_i |\psi\rangle_i \langle\psi|_i$ of the prepared state. Looking at the third bit in Figure 13, which reads 1,0,1,0 corresponding to the highest peaks, we have the demonstration that the algorithm has encoded the data into our qubits. Though one can see 6 qubits in the circuit, 4 of them are actually ancillary, i.e., they are dummy qubits needed only for the processing and can be discarded at the end of the circuit by disentangling them from our data. Therefore, as the number of bits (N) grows larger, it can be shown that the most optimal circuit would need $\log_2(N)$ qubits. This scaling in data loading is necessary if we have to build quantum processing circuits with exponential scaling.

6.2 Output Measurements

After loading and producing new data, the data are processed by a series of unitary operations required by the problem and the final state of the qubits is the output result we seek. The challenge is to efficiently estimate these final states, which are the complex probability amplitudes and the state vectors. This process is called quantum state estimation or quantum state metrology or quantum state tomography [13, 46–48].

This process is not straightforward because, in order to probe these qubits, one would have to perform a measurement, which physically means interacting with the wavefunctions. Such interactions, apart from inducing noise, also collapse the wavefunctions to one of the basis states resulting in smudging and loss of data and results. The most elementary way to evaluate a state is to perform a von Neumann projective measurement on the state along (say) the z-axis that forms the eigenvector of the computational basis states $|0\rangle$ and $|1\rangle$. But, quantum mechanics tells us that projective measurements project the state to a particular basis vector and may not always depict the complete information represented by a wavefunction. To get a total estimate of a given superposition state, one would have to do something better.

The next simplest method is the empirical probability estimation by ensemble averaging, i.e., conducting several identical experiments to generate a set of outputs and perform projective measurements every time, to collect the probability statistics of each state under superposition (whose accuracy obviously increases with the ensemble size). In order to make our discussion clear, let us briefly discuss the meaning of quantum measurements.

QM Postulate: *Measurement is an operator A that acts on a set of quantum states ψ_i to yield a physical observable eigenvalue “a” with the following properties:*

1. The probability of obtaining outcome “a” is

$$P(a) = \langle\psi_i|A_a^\dagger A_a|\psi_i\rangle. \tag{24}$$

2. The post measurement state would be

$$|\psi_f\rangle = \frac{A_a|\psi_i\rangle}{\sqrt{\langle\psi_i|A_a^\dagger A_a|\psi_i\rangle}}. \tag{25}$$

Now, these measurements give us only the probability of a given eigenvalue “a” from a given state $|\psi\rangle$. But this output is “weak” in the sense that it is only from one single state and the measurement process, and other successive measurements may not yield the same result even though we start out from identically prepared

initial states, because of noise and decoherence. Thus we might have to average over many such experiments and measurements. Since output of each experiment differs, it naturally creates an ensemble of states $\{|\psi_i\rangle\}$.

As an aside, we note that this fact could be used advantageously for turbulence simulations. Since even slightly different initial conditions lead to different dynamics, due to the inherent chaotic nature of the system, in general, the present ensemble automatically represents an ensemble of many turbulent evolutions. The standard way that quantum mechanics suggests for characterizing such an ensemble of states, whose exact form we have to probe, is to use the density operator formalism. If p_α is the probability of obtaining a state $\{|\psi_\alpha\rangle\}$, the density operator is given by $\rho = \sum_\alpha p_\alpha |\psi_\alpha\rangle\langle\psi_\alpha|$. We may restate the previous measurement postulate in terms of ρ as

1. The probability of obtaining outcome “a” is

$$P(a) = \text{Tr}(A_a^\dagger A_a \rho). \quad (26)$$

2. Its post measurement state would be

$$\rho_f = \frac{A_a \rho A_a^\dagger}{\sqrt{\text{Tr}(A_a^\dagger A_a \rho)}}. \quad (27)$$

The density operator has the property that $\text{Tr}(\rho) = 1$ i.e., probabilities (non-negative eigenvalues) sum up to unity. Since we are only interested in obtaining the statistics of each superposition state, we ask the question: What type of measurement procedure respects the positivity and completeness property of the density operator as well as yield the probability amplitudes of the states in the ensemble? The answer is termed POVM Measurements (Positive Operator Valued Measure). The POVM elements constitute a set of operators $\{P_{VM}^a\} \equiv \{A_a^\dagger A_a\}$ that are constrained to be positive, since $\langle\psi_i|P_{VM}^a|\psi_i\rangle = \langle\psi_i|A_a^\dagger A_a|\psi_i\rangle = P(a) \geq 0$, and complete, i.e., $\sum_a P_{VM}^a = \sum_a A_a^\dagger A_a = \mathbb{I}$. In fact, as one can easily observe, we can even get the measurement operator corresponding to a given POVM element by $\sqrt{P_{VM}^a} = A_a$.

We are generally not interested in the measurement itself but only in the statistics. Thus the POVM provides a clean way of doing this without worrying about the state itself. Another important advantage of positive definiteness of these operators is that, for a given set of non-orthogonal states, a POVM set of operators $\{P_{VM}^a, P_{VM}^b \text{ and } (P_{VM}^c = \mathbb{I} - (P_{VM}^a + P_{VM}^b))\}$ can be used to compute their statistics or distinguish the states. Thus, the output of sandwiching the operators between state vectors gives us the statistics. The operators are designed to create a unique one-to-one mapping from the operator space to a particular output state. So any positive output, resulting from the first two operators in our set, points uniquely to a specific state, while an output from the third operator implies that we cannot comment anything about the state. So, essentially such a measurement process never lets us go wrong in identifying states, but at the cost of being unable to comment about the output from one of its operators. With this machinery we are ready to look at quantum state tomography.

6.3 Quantum State Tomography

Let us begin with a density matrix representing an unknown quantum state that needs to be profiled. Say we have just one set of non-orthogonal qubit states. Experimentally it is impossible to construct the quantum state from just one copy of the states. But we can make several POVM measurements from multiple copies and compute the statistics. The set of operators $\{\mathbb{I}/2, X/2, Y/2, Z/2\}$ that form a set of orthonormal operators can be used to expand the density matrix as

$$\rho = \frac{1}{2} \left(\text{Tr}(\rho)\mathbb{I} + \text{Tr}(X\rho)X + \text{Tr}(Y\rho)Y + \text{Tr}(Z\rho)Z \right). \quad (28)$$

The expectation of these operators can be obtained by $\text{Tr}(X\rho)$. Now each of these expectation values can be estimated by repeated measurements. Once we have a large sample size with good estimates of each of these operator outputs, we can reconstruct the density operator of the unknown state. The standard deviation of the estimate is $1/\sqrt{N}$, where N is the ensemble size, as one would expect for a Gaussian random variable in the large N limit. This in essence is the picture of Quantum State Tomography. The same procedure can be extended to multiple qubit systems as well. This procedure is usually achieved in practice by a few popular techniques whose illustrations and explanations are given in [49]. More advanced quantum metrology techniques, which also make use of quantum phase estimation methods that could be used for fluid dynamics applications, are discussed in [65]. As an example of a simple ensemble averaged measurement, we demonstrate tomography of two entangled qubits. For this, we prepare a sample entangled Bell state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{29}$$

via Amplitude Loading (as shown in Figure 14) and try to estimate its probability amplitudes by running on the IBMQ. As one can clearly see, each qubit has the probability 1/2, which is exactly what we want to estimate experimentally. As one can see the results from Figure 15, the histogram peaks properly with almost equal probabilities (≈ 0.5) at $|00\rangle$ and $|11\rangle$. The small but finite probabilities of the other two states is due to quantum errors and decoherence in the system. With this, we conclude our brief discussion on output measurements.

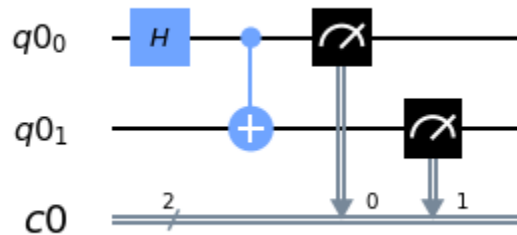


Figure 14: The quantum circuit for entanglement preparation and measurement

6.4 Data Processing

Having briefly discussed inputs and outputs of a quantum computational process, we shall now examine the quantum algorithms that one needs to use to generate, process and manipulate data. A comprehensive and updated collection and descriptions of most of the available quantum algorithms can be found in [13, 33, 49, 54, 66]. To keep our discussion contained, we choose only a few important ones that are possible candidates for fluid dynamics simulations in Table 2 (Table notes: Q/C-Quantum/Classical; VQE-Variational Quantum Eigen solver; QAOA-Quantum Approximate Optimisation Algorithm; ** t = number of qubits with n-bit phase approximation and ϵ error. Specific references to the algorithms in Table 2 can be found at [33, 49, 54, 66]) With these, the working procedure to go about solving a CFD problem, involving a specific set of PDEs, on a QC would involve the following steps (In Section 10.2, we shall illustrate with an example):

Step A - Data Loading: First, we need to initialise the variables of the PDE under consideration, say the velocity and pressure field, \mathbf{u} , \mathbf{p} to an appropriate numerical initial iterate value. This is done by the quantum state

Table 2: Quantum Algorithms

#	Algorithm	Description/Used in	Complexity/Speed-up
1	Superdense Coding [13]	Data compression & communication	Compression Ratio 2:1
2	Quantum Fourier Transform (QFT) [13]	DFT, Phase Estimation, Period Finding, Arithmetic, Discrete log & spectral methods	Q: $[\Theta(n^2), \Theta(n \log n)]$ C: $\Theta(n2^n)$ (n=#gates)
3	Quantum Phase Estimation [13]	Quantum phases, Order Finding, Shor's Algorithm, HHL, Amplitude Amplification & Quantum Counting [50]	$O(t^2)$ operations** $t = n + \left\lceil \log \left(2 + \frac{1}{2\epsilon} \right) \right\rceil$
4	Grover's Search [13, 51] & Amplitude Amplification [52]	Data search, Amplitude Estimation, Function minima, approx. & Quantum counting	Q: $O(\sqrt{N})$ C: $O(N)$ (N=#ops)
5	Quantum Simulation [13, 27]	Integrates Schrödinger equation, HHL, All Hamiltonian system simulations $(e^{-iHt/\hbar})$	<i>superpoly</i> <i>poly(n,t): n=dof, t= time</i>
6	Gradients [53, 54]	Computes gradients, convex optimisation volume estimation, minimising quadratic forms	<i>quadratic - superpoly</i>
7	Partition Function [54] & Sampling	Evaluate/approx partition functions Pott's, Ising Models & Gibbs sampling	<i>quadratic - superpoly</i>
8	Linear Systems & HHL Algorithm [54, 55]	Solves $\mathbf{AX}=\mathbf{b}$ for eigen values & vectors ODEs, PDEs, simultaneous eqns. Optimisation, Finite Element Methods etc	<i>superpoly - exponential</i>
9	ODE [54, 56]	Integrates $\dot{\mathbf{x}} = \alpha(t)(\mathbf{x}) + \beta(t)$ & similar forms	<i>superpoly - exponential</i>
10	Wave Equation [57, 58]	Integrates $\ddot{\phi} = c^2 \nabla^2 \phi$ & similar forms	<i>superpoly - exponential</i>
11	PDE / Poisson Equation [54, 59, 60]	Integrates $-\nabla^2 \phi(\mathbf{x}) = b(\mathbf{x})$ and PDEs of similar forms: $\mathbf{D}\phi(\mathbf{x}) = b(\mathbf{x})$	<i>superpoly - exponential</i>
12	Matrix Product States with VQE [61]	Solves non-linear PDEs with VQE and MPS	-
13	QFT Arithmetic [62]	QFT based: + , - , * , mean , weighted sum	<i>superpoly - exponential</i>
14	Function Evaluation [63]	(Ex) inverse, exponentiation.. etc for State Loaded data	varies
15	VQE and QAOA [33]	Computes optimisation type problems	varies
16	Quantum Annealing [64]	Computes optimization type problems	varies

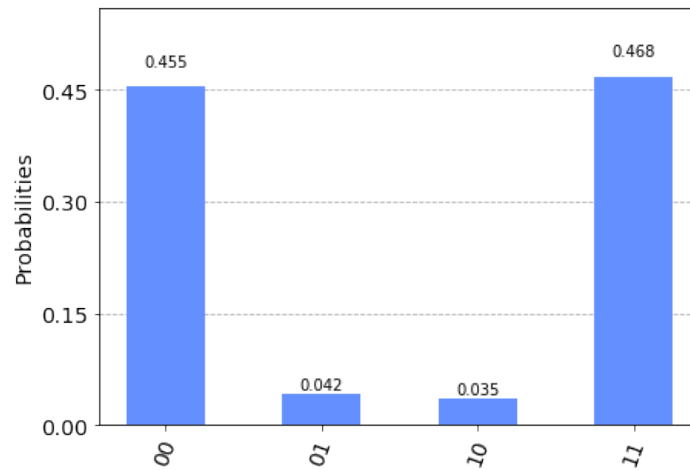


Figure 15: Probability amplitude estimates

preparation as described earlier.

STEP B - Data Acquisition: To integrate the governing equations, one could think of several numerical methods to do this for example:

1. Finite Difference Method: Like any FDM, we discretize the system first. This discretization procedure, along with boundary conditions, yields a matrix which performs the differential operation. We could use any variant of Algorithm #8 #13 to start solving this problem. In general, solving such an FDM setup boils down to solving a matrix inversion problem, which is done by Algorithm #10 (HHL) = Alg #2 + Alg #3 + Alg #4. The speedup and complexity depends on the specific combination of each of these algorithms.
2. Pseudospectral Methods: We can use Algorithm #3 (QFT) to first map both the LHS and RHS to the spectral space, first computing the derivatives in the spectral space and then using HHL to invert.
3. Amplitude amplification: Further, we can append Algorithm # 5 to perform amplitude amplification to amplify the probability of obtaining the right answer in every experimental run.

STEP C - Output Measurements: Once we obtain the eigenvalues and eigenvectors, we can perform a quantum state tomography to extract the results and store them in classical registers.

Hybrid Simulations: It is important to highlight, that for near term simulations considering hybrid classical-quantum methods might be the key. For instance, the currently available pseudospectral DNS codes face a major bottleneck with the FFTW steps that need to be mapped out for computing derivatives. If at all we could efficiently set up a hybrid algorithm, where only the FFTW steps are replaced by the QFT, we could tap into the exponential speed up offered by QFT. To motivate this direction, a QFT demonstration is given in the following section.

6.5 Quantum Fourier Transform

The Quantum Fourier Transform [13] is very similar to the Discrete Fourier Transform performed by the currently available FFTW routines. To keep the discussion concrete, consider the simple example of discretizing the domain and sampling the function at (say) four points. Now, the DFT of this function f at these four points may be written as

$$F[k] = \sum_{j=0}^{j=N-1} f[j] \exp\left(2\pi i \frac{jk}{N}\right). \tag{30}$$

Here, the amplitudes $f[j] = \{f[0], f[1], f[2], f[3]\}$ are being Fourier transformed to $F[k] = \{F[0], F[1], F[2], F[3]\}$. Very similarly, the Quantum Fourier Transform (QFT) is a unitary operation that transforms as

$$\sum_{i=0}^{i=N-1} \alpha_i |i\rangle \mapsto \sum_{i=0}^{i=N-1} \beta_i |i\rangle, \tag{31}$$

where

$$\beta_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{j=N-1} \alpha_j \exp\left(2\pi i \frac{jk}{N}\right) \tag{32}$$

Equivalently, if we set $\omega^{jk} = \exp\left(2\pi i \frac{jk}{N}\right)$

$$|\alpha\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{\beta=0}^{\beta=N-1} \omega^{jk} |\beta\rangle. \tag{33}$$

Note that the states in the computational basis $|0\rangle, |1\rangle, |2\rangle, |3\rangle$ can be expressed in the binary form as well,

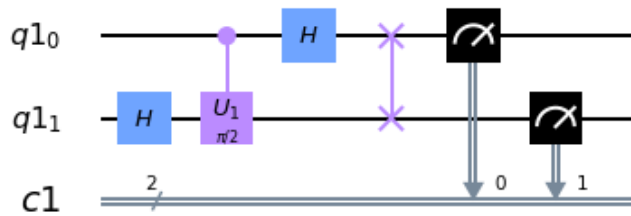


Figure 16: The quantum circuit for 2-qubit QFT

which is $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. With this step, we obtain the transformations of the sampled amplitudes to be

$$\begin{aligned} \beta_0 &= \frac{1}{2}[\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3] \\ \beta_1 &= \frac{1}{2}[\alpha_0 + i\alpha_1 - \alpha_2 - i\alpha_3] \\ \beta_2 &= \frac{1}{2}[\alpha_0 - \alpha_1 + \alpha_2 - \alpha_3] \\ \beta_3 &= \frac{1}{2}[\alpha_0 - i\alpha_1 - \alpha_2 + i\alpha_3], \end{aligned}$$

caused by the unitary transformation operator U_{QFT} . If $\omega = e^{i\pi/2}$ the operator is given by

$$U_{QFT} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & 1 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega \end{pmatrix}. \quad (34)$$

The quantum circuit that implements such a 2-qubit QFT is shown in Figure 16 below.

To illustrate its action, let us prepare the state already shown in Figure 11 using Amplitude Loading and run the QFT on it. The expected outputs of the Fourier transform for these amplitudes, computed analytically, is $\beta_0 = 0.574, \beta_1 = 0.037, \beta_2 = 0.306, \beta_3 = 0.138$, while the results obtained from the QFT are shown in Figure 17. The QFT results are $\beta_0 = 0.569, \beta_1 = 0.037, \beta_2 = 0.311, \beta_3 = 0.083$. The precision can improve

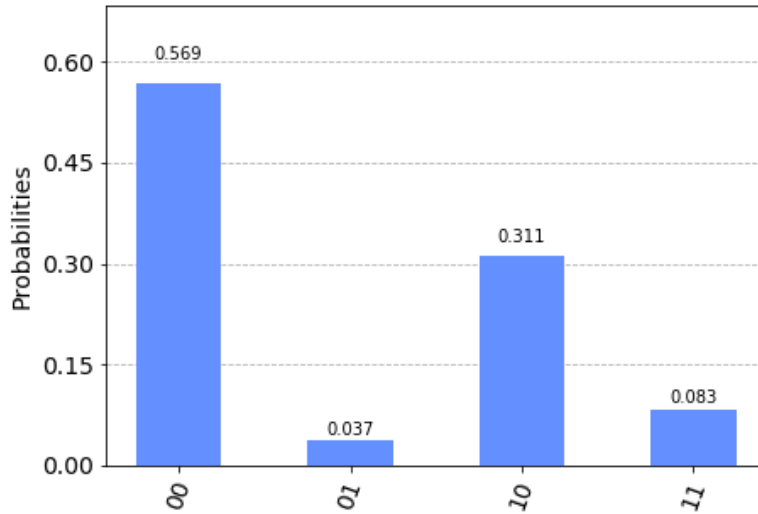


Figure 17: The QFT output

with better Quantum State Tomography, while the computation of QFT is exponentially faster than DFT.

6.6 Quantum Turbulence

It would obviously be instructive to look at quantum fluid dynamics as well. A quantum fluid, like any other quantum system, is described by its corresponding many-body interacting Hamiltonian and its evolution is governed by a corresponding Schrödinger equation (a quick insight can be obtained from [67, 68]). The evolution described by this equation represents an analytically consistent way of obtaining a proper microscopic evolution. Since tracking $\approx 10^{23}$ particles is computationally impractical, given that macroscopic observables are what we wish to understand, there is a huge motivation for developing macroscopic equations of motion. The complete many-body quantum simulation may well be possible with a powerful quantum computer, but one may also be able to develop more efficient versions of numerically integrating different model equations. Both methods provide insight on the right descriptions of quantum turbulence and vortex reconnection. The following QC tasks are possible candidates for studying quantum turbulence.

6.6.1 Quantum simulation

The most microscopic description is that of the Hamiltonian picture. For instance, consider a Bose-Einstein condensate Hamiltonian and its corresponding Schrödinger equation:

$$\mathcal{H}_{BEC} = -\frac{\hbar^2}{2m}\nabla^2 + V_{ext}(\mathbf{r}) + U_{int}(\mathbf{r} - \mathbf{r}'), \quad (35)$$

$$\frac{\partial}{\partial t}|\Psi_{BEC}\rangle = \frac{1}{i\hbar}\left[-\frac{\hbar^2}{2m}\nabla^2 + V_{ext}(\mathbf{r}) + U_{int}(\mathbf{r} - \mathbf{r}')\right]|\Psi_{BEC}\rangle \quad (36)$$

$$|\Psi_{BEC}\rangle = e^{-i\mathcal{H}_{BEC}t/\hbar}|\Psi_{BEC0}\rangle. \quad (37)$$

Now this calls for a many-body simulation. With the growing surge of QC methods of many-body algorithms, we could perform a quantum simulation (Algorithm #7) [13] to evolve this equation. Among the methods being developed, the Trotter decomposition method (or the Lie-Trotter-Suzuki decomposition) is worth mentioning. If P and Q are Hermitian operators (that need not commute), we have, for any t

$$\lim_{N \rightarrow \infty} (e^{iPt/N} e^{iQt/N})^N = e^{i(P+Q)t}. \quad (38)$$

Now the same result can be used to derive higher order corrective equations such as:

$$e^{i(P+Q)\delta t} = (e^{iP\delta t} e^{iQ\delta t}) + \mathcal{O}(\delta t^2) = (e^{iP\delta t/2} e^{iQ\delta t} e^{iP\delta t/2}) + \mathcal{O}(\delta t^3). \quad (39)$$

Now this is a very useful result, since we can take our \mathcal{H}_{BEC} and split it into $\mathcal{H}_{BEC} = \sum_{i=1}^n \mathcal{H}_i$ Hamiltonian operators acting on smaller sub-systems spanning a local Hilbert space. If $[\mathcal{H}_i, \mathcal{H}_j] = 0 \forall i, j$ and $\hbar = 1$, we have

$$e^{-i\mathcal{H}_{BEC}t} = e^{-i\mathcal{H}_1t} e^{-i\mathcal{H}_2t} \dots e^{-i\mathcal{H}_nt}. \quad (40)$$

But if commutation is not imposed, a similar correction is obtained:

$$e^{-i\mathcal{H}_{BEC}t} = e^{-i\mathcal{H}_1t/2} e^{-i\mathcal{H}_2t} e^{-i\mathcal{H}_1t/2} + \mathcal{O}(\delta t^3). \quad (41)$$

Following this procedure, each time-step operator can be decomposed into basic unitary logic gates and a corresponding evolution circuit can be constructed. The following model equations would be amenable to numerical integration using QC algorithms.

6.6.2 The two-fluid model

Simulating the Landau's equations would be useful for those looking at quantum fluids at low velocities and with no quantum vortices, since this model works best for irrotational and incompressible flows. The idea would be to build on the previously discussed algorithms for dealing with ODEs and PDEs procedures to integrate the following equations:

$$\frac{\partial \mathbf{u}_1}{\partial t} + \mathbf{u}_1 \cdot \nabla \mathbf{u}_1 = -\nabla \left(\frac{p_1}{\rho_1} \right), \quad (42)$$

$$\frac{\partial \mathbf{u}_2}{\partial t} + \mathbf{u}_2 \cdot \nabla \mathbf{u}_2 = -\nabla \left(\frac{p_2}{\rho_2} \right) + \frac{\nu}{\rho_2} \nabla^2 \mathbf{u}_2. \quad (43)$$

Here the subscripts 1 and 2 correspond to superfluid and normal fluid, respectively. Let us now look at methods that includes quantum vortices as well.

6.6.3 Gross-Pitaevskii model

Along with a few approximations and assumptions, we can use the standard trick of Madelung Transformation to establish a relationship between the BEC wavefunction and fluid macroscopic properties such as density and velocity. This is the Gross-Pitaevskii equation

$$\frac{\partial}{\partial t} |\Psi_{cond}\rangle = \frac{1}{i\hbar} \left[-\frac{\hbar^2}{2m} \nabla^2 + V_{ext}(\mathbf{r}) + U_{int} |\langle \Psi_{cond} | \Psi_{cond} \rangle| \right] |\Psi_{cond}\rangle. \quad (44)$$

The built-in assumptions are: (a) Though the actual BEC wavefunction is a sum of the actual condensate wavefunction and the perturbative term, at $T \sim 0$, we say $\Psi_{BEC} \approx \Psi_{cond}$. (b) Length scales are of the order of the vortex cores. (c) Only contact interactions are allowed $U_{int} = U\delta(\mathbf{r} - \mathbf{r}')$. This model is the nearest microscopic description, yet has many limitations. A detailed outlook could be obtained from [67, 68].

6.6.4 Vortex filament model

The next level would be to move to scales greater than the vortex core sizes. We visualise the fluid as a ensemble of arcs of quantum vortices and track these vortex arcs \mathbf{l} , which is the vortex filament model. The evolution of these arcs is given by

$$\frac{d\mathbf{l}}{dt} = \mathbf{u}_{sa} + \mathbf{u}_f, \quad (45)$$

where \mathbf{u}_{sa} is the self-advecting velocity of the vortex and \mathbf{u}_f is the mutual friction between the normal fluid arc surface. The computationally heavy step to be done by the QC is the evaluation of the Biot-Savart integral

$$\mathbf{u}_i = \frac{\Gamma}{4\pi} \oint \frac{(\mathbf{l} - \mathbf{l}')}{|\mathbf{l} - \mathbf{l}'|^3} \times d\mathbf{l}', \quad (46)$$

to compute \mathbf{u}_{sa} , Γ being the circulation of the vortex filaments.

6.6.5 HVBK model

This model, obtained from a slight amendment of Landau's equation, gives the best description for the largest scales, much larger than the core size. The additional terms are the mutual friction force \mathbf{f}_{mf} and the arc tension force \mathbf{f}_T . This model too has limitations owing to its assumptions on the vortex arc orientations. The equations are (let us call $\mathbf{F} = \mathbf{f}_{mf} + \mathbf{f}_T$):

$$\frac{\partial \mathbf{u}_1}{\partial t} + \mathbf{u}_1 \cdot \nabla \mathbf{u}_1 = -\nabla \left(\frac{p_1}{\rho_1} \right) - \mathbf{F}, \quad (47)$$

$$\frac{\partial \mathbf{u}_2}{\partial t} + \mathbf{u}_2 \cdot \nabla \mathbf{u}_2 = -\nabla \left(\frac{p_2}{\rho_2} \right) + \frac{\nu}{\rho_2} \nabla^2 \mathbf{u}_2 + \frac{\rho_1}{\rho_2} \mathbf{F}. \quad (48)$$

7.0 VARIATIONAL SOLVERS AND QUANTUM ANNEALERS

The last method to be outlined is based on variational optimization. Suppose we want to solve the conventional CFD problem of simulating a Stokes flow by using a discretization solver such as Gauss-Seidel or Jacobi. The problem reduces to solving for eigenvalues of the form $A\mathbf{x} = B$ using the HHL algorithm. It can also be solved as an optimization problem. That is, we define a cost function such as the difference between the LHS and RHS of the eigenvalue problem, and iterate and modify \mathbf{x} so as to minimise the cost function to 0. Classical

methods include algorithms such as gradient descent, steepest descent, conjugate gradient method, etc. Such optimization procedures could be used for QC as well.

A. Variational Quantum Eigen (VQE) Solver. The idea stems from the principle of quantum mechanics for solving the eigenvalue problems variationally. It is usually done as a hybrid of quantum and classical computing. So far, VQE has been applied for different condensed matter and quantum chemistry problems, but it can be extended to other problems as well. On a hybrid machine, the steps are noted below. Detailed descriptions can be found in [33, 49, 69].

1. Consider a matrix P with one of its eigenvectors $|\psi_p\rangle$. Then we know that the $|\psi_p\rangle$ is invariant in the sense of $P|\psi_p\rangle = p|\psi_p\rangle$, where p is the corresponding eigenvalue. Let us regard P as a Hamiltonian, which is a positive definite Hermitian matrix, with positive and real eigenvalues. Thus, the expectation value of the Hamiltonian is $\langle\psi|\mathcal{H}|\psi\rangle \geq 0$. The smallest eigenvalue $p_{min} \leq \langle\psi|\mathcal{H}|\psi\rangle$ corresponds to the ground-state energy of the system (≥ 0), which can be estimated by Algorithm 2.
2. Thus while a QPU computes expectation values, the CPU runs an optimisation algorithm; together they can be used to estimate the eigenvalue and ground state configurations.

More recently some classical algorithms have been developed that are inspired by the fundamental principles of using quantum mechanics or computing to study fluid flows, which are further also translatable into an equivalent quantum circuit. For instance [61], uses variational methods coupled with Matrix Product States (MPS) to extract some features of the Navier-Stokes solutions.

B. Quantum Approximate Optimization Algorithms (QAOA): Generally, combinatorial optimization methods may not be tractable with polynomial resources. Other than developing problem specific methods, approximate algorithms such as QAOA can be handy. The goal is to take a discrete variable as an input, which could be strings of binaries such as $\mathbf{x} = x_1 \dots x_n$, where $x_i \in \{0, 1\}$ defines a cost function $E(\mathbf{x})$ that needs to be maximized. The cost function is essentially a map from $E(\mathbf{x}) : \{0, 1\}^n \mapsto \mathbb{R}$. The QAOA [33, 49, 70] thus forms the set of algorithms which does exactly this, and guarantees that the approximation ratio α satisfies

$$\alpha = \frac{E(\mathbf{x})}{E_{max}} \geq \alpha_{opt}. \quad (49)$$

C. Quantum Annealing: This method is now widely used to run on what are essentially known as Quantum Annealer Machines (which in essence are not quantum computers) such as those produced by companies such as DWave. The physical principle here is to use the quantum analogue of simulated annealing that one uses to solve optimization problems in classical physics, but the phenomena of quantum tunnelling sets the quantum version apart from the classical one. This phenomena is exploited for scanning fast through different minima of a given energy landscape of a cost function. In the classical Monte Carlo, one would have to thermally excite the system to jump the energy barrier to the next minimum, while in the quantum case, even with tall energy barriers and with a certain thin barrier width, one can "tunnel" to the adjacent minimum as shown schematically in Figure 18. bThe DWave system does exactly this. With a combination of quantum tunnelling, quantum entanglement and a transverse field bias, it can perform quantum annealing efficiently [64, 71] and find optimal solutions via minimization. This has already been put to use to study the Navier-Stokes channel flow [72]. In this work, one first converts the NS equation into a discretized version and sets up the problem as an $A\mathbf{x} = B$ eigenvalue problem as usual. Later, this is numerically investigated by converting the problem into

Algorithm 2: Variational Quantum Eigenvalue Solver

Input: \mathcal{H} , $|\psi(\mathbf{k})\rangle$, \mathbf{k} (parameter), ϵ (tolerance)
Output: $|\psi(\mathbf{k})\rangle_{opt}, E_{opt}$

```

1 Function VQE ( $\mathcal{H}$ ,  $|\psi(\mathbf{k})\rangle$ ,  $\mathbf{k}$ ,  $\epsilon$ ) :
2   INITIALISE ( $|\psi(\mathbf{k})\rangle$ )
3   (Can also use  $U_3(\mathbf{k})$  gates (Eqn. 19) for parametrisation)
4   while ( $p_{min} \leq \epsilon$  &&  $P|\psi(\mathbf{k})\rangle == p_{min}|\psi(\mathbf{k})\rangle \approx p|\psi_p\rangle$ ) do
5      $E_{opt} = \langle \psi | \mathcal{H} | \psi \rangle$  (ground state energy estimate)
6      $|\psi(\mathbf{k})\rangle_{opt} = |\psi(\mathbf{k})\rangle = \text{CLASSICAL\_OPTIMISER}(|\psi(\mathbf{k})\rangle)$ 
7   return  $|\psi(\mathbf{k})\rangle_{opt}, E_{opt}$ 
8 Function CLASSICAL\_OPTIMISER ( $|\psi(\mathbf{k})\rangle$ ) :
9   Optimise  $|\psi(\mathbf{k})\rangle$  based on parameter  $\mathbf{k}$  on a CPU
10  return  $|\psi(\mathbf{k})\rangle_{opt}$ 

```

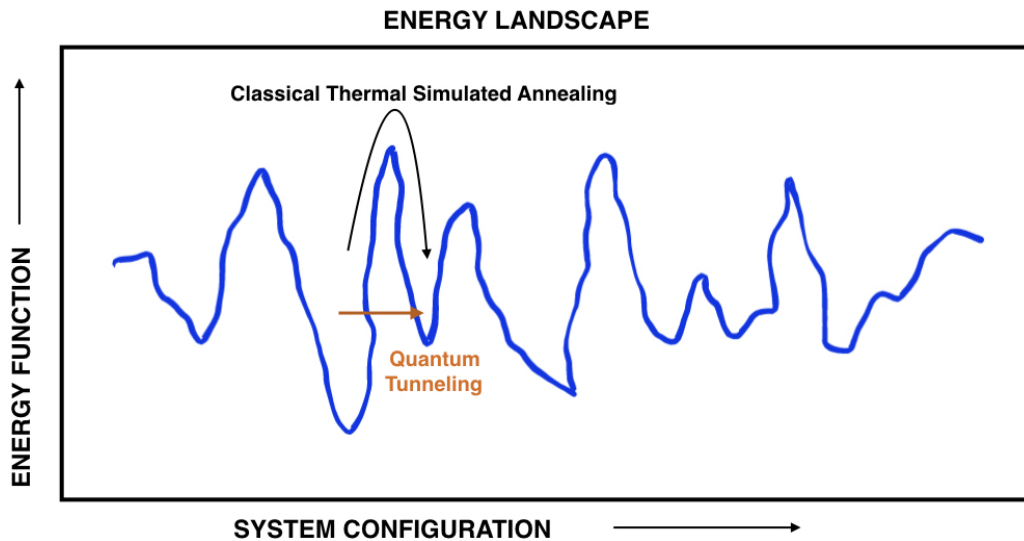


Figure 18: Quantum Annealing

an optimization type setup called the Quantum Unconstrained Binary Optimisation (QUBO) supported by the DWave machine.

In general, many variational and optimization type methods are already being used for commercial applications such as traffic flow management and finance management with very big quantum annealers such as DWave which is offering 5000 qubits. Though the complexity estimates of these methods varies and is not yet clearly established, it still offers a lucrative quantum protocol that can solve optimization problems with a decent speedup.

8.0 QUANTUM PROGRAMMING AND MACHINES

Coming to the implementation and the actual execution of these ideas and algorithms, what we need are (a) efficient quantum computer simulators to test the correctness of quantum algorithms, and (b) real quantum computing devices to execute and assess the quantum advantage of practical QCs. There are now a large and growing number of efforts that have already built, and are trying to build, better quantum computers. Each of these QCs is being implemented using different quantum physical realizations and quantum materials that would be robust against external noise and decoherence, called Quantum Processing Units (QPU) or Quantum Processors. Given a QPU, the process of programming a set of quantum algorithms and converting them into forms understandable by a quantum machine, using instructions of suitable programming languages, is called quantum programming.

The different programming languages, though seemingly similar, vary in terms of instruction sets and the actual physics governing the operation of the QC. Different quantum programming methods and packages are being developed by different QC companies. Concentrating on computational fluid dynamics, different workstations ranging from simple PCs to massive supercomputers have been used. Similarly, the currently available quantum devices and programming packages are summarized in Figure 19. Each of these available quantum programming kits have their own strengths and weaknesses; for instance, certain devices are better set for optimization type problems compared to others.

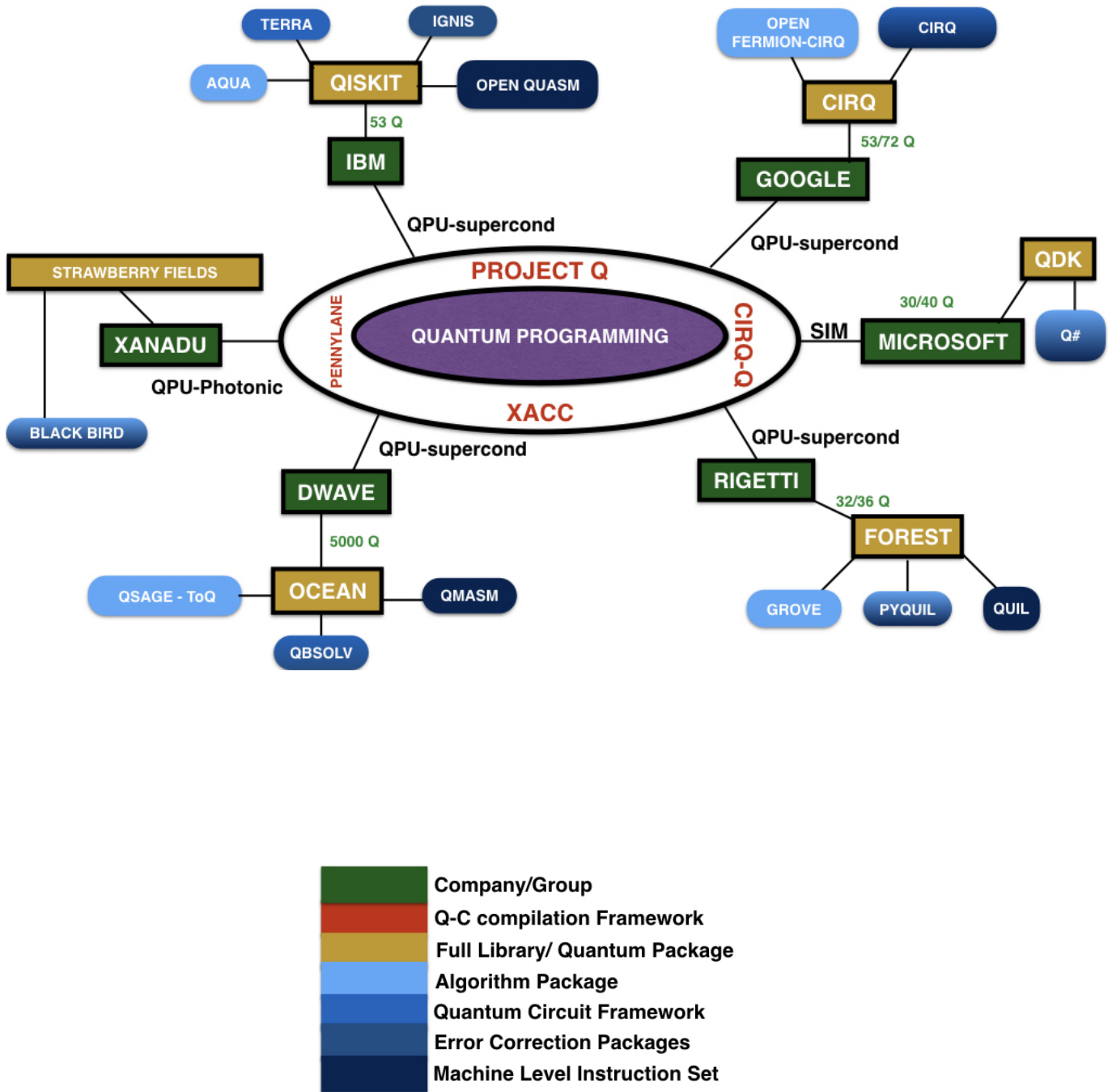


Figure 19: Different QCs and Architectures

Part III

Introducing *QuOn* - A Quantum Simulator and Quantum Linear System Algorithms

9.0 QUON - A HIGH PERFORMANCE QUANTUM SIMULATOR

Parallel to experimental implementations on real quantum devices, the key to advancing QCFD is the availability of quantum simulators that let researchers create, tweak and test new algorithms. Here we introduce a new quantum-simulation toolkit, aimed specifically for fluid dynamics applications, that can be used either independently or as part of other code bases. This is a toolkit for a fluid dynamicist to build and explore the new arena of QCFD [3]. We shall call this hybrid simulator (with its logo is given below). It brings together classical CFD and QC as **QuOn**!¹



9.1 The basic structure

Figure 19 captures several of the commercially available quantum simulation packages produced by a host of companies. Quite apart from them, Quipper [74] is another Haskell based quantum compilation package. However, all these tools are mostly general purpose quantum softwares. On the other hand, there are softwares such as ANSYS and OpenFOAM for performing classical CFD simulations. Thus, the desire to have a single dedicated quantum simulator, aimed particularly at solving CFD problems on quantum machines, motivated the genesis of QuOn, which is a high performance quantum simulator based on a C++ core. With QuOn, one can at present:

1. **Create custom quantum state/wavefunction(s)** with a chosen number of qubits, along with several functions to probe different attributes of the quantum state at any instant during quantum simulation.
2. **Compose quantum circuits or projects** using a large library of quantum gates and assemble them with custom circuit design to implement quantum algorithms.
3. **Integrate classical CFD solvers** such as finite difference, finite element methods written in any other language (e.g., MATLAB, Python) and run them in hybrid mode with quantum subroutines.

¹As an aside: “Quons” are a kind of mathematical quantum particles that violate fermionic and bosonic statistics, but their algebra is somewhat in between the two, called as “quon-algebra” [73].

4. **Simulate and test** these quantum algorithms by measuring the fidelity of quantum solutions with respect to classical/analytical results.
5. **Visualize quantum circuits and states** graphically using our simplistic visualization kit.
6. **Acquire and store quantum information** into classical formats via appropriate state tomography techniques.
7. **Faster and easier** parallelization than Python based tools.
8. **Offers more control** over fine details of the algorithm.
9. **Portable** (soon) to other supercomputing platforms and translatable to Qiskit instructions.

9.2 Features

Here, we take a look at its features in a bit more detail:

1. **Qubits:** Simulation capabilities range up to 20 qubits.
2. **Quantum gates library:** Hosts a vast library of 40+ quantum gates within the **gates.cpp** subroutine.
3. **Circuit composer:** The composer class within **circuit.cpp** lets a user create custom circuits invoking different gates and also call in-built subroutines for:
 - (a) **Gate decomposition:** To decompose any given multi-control gates into simpler gate blocks using methods such as Euler decomposition **eulerdecomp.cpp** and multi-CNOT decomposition.
 - (b) **Unitary decomposition:** To decompose a general n-by-n unitary matrix into 2-level unitaries which can further be reduced to single qubit and CNOT gates using the **unitary_decomp.cpp**.
 - (c) **Visualize:** To draw out the composed circuits for visualizing and verifying the constructions using the **visualiz.cpp**.
4. **Quantum State:** The user can initialize arbitrary quantum states and extract different features:
 - (a) **State preparation:** As already mentioned in Section 6.1, this could be done in several ways. Here, let us look at another method of amplitude loading called “Cosine-Sine Decomposition” (CSD) to prepare a quantum state.

Cosine-Sine Decomposition (CSD): Suppose we wish to prepare a state $|b\rangle = \frac{1}{\sqrt{66}}(2|0\rangle + 6|0\rangle + 1|0\rangle + 5|0\rangle)$, which could be the right hand side of an $A\mathbf{x} = \mathbf{b}$ problem. With CSD, we can create a generic state $|b\rangle \in \mathbb{R}^{2^n}$ which is normalized to $\|b\| = 1$ by applying, successively, a set of rotation gates whose parameters are pre-calculated into a binary tree shown in Figure 20. The binary tree is calculated bottom-up, while its eventual implementation as a quantum circuit is done by the top-down consideration of the tree. Let examine this in more detail:

- i. The set of values that needs to be loaded populates level n along with their respective sign information.
- ii. To begin constructing the tree, starting at level n, we square and sum the values pair-wise at each successive level as shown in Figure 20.

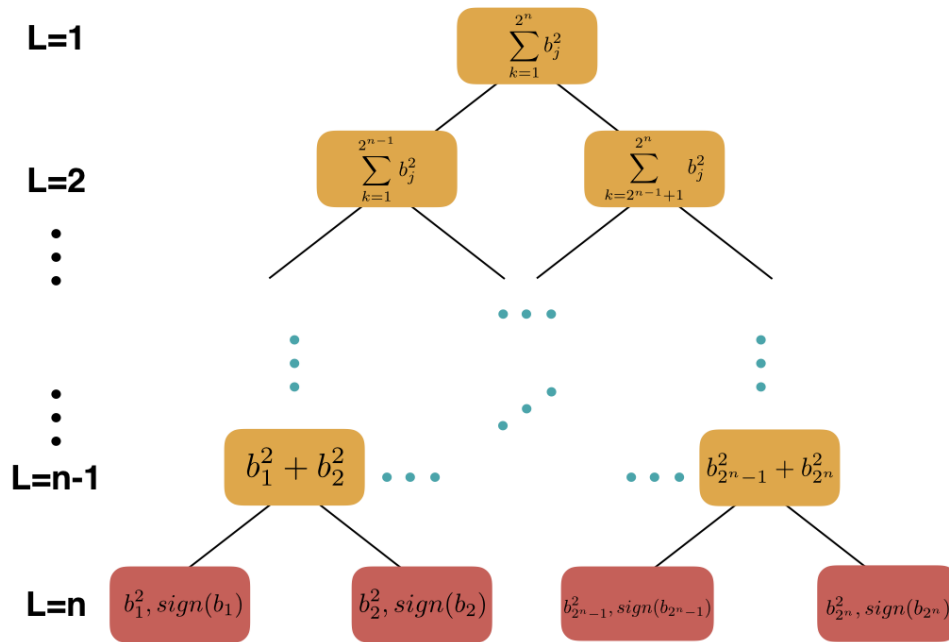


Figure 20: Cosine-Sine Decomposition Binary Tree

iii. This is done until the vertex of the tree is reached which essentially needs to be = 1.0, since the final quantum state is required to be normalized. For the example state $|b\rangle$, the tree is constructed as shown in Figure 21, where the normalization constant here is $\sqrt{66}$.

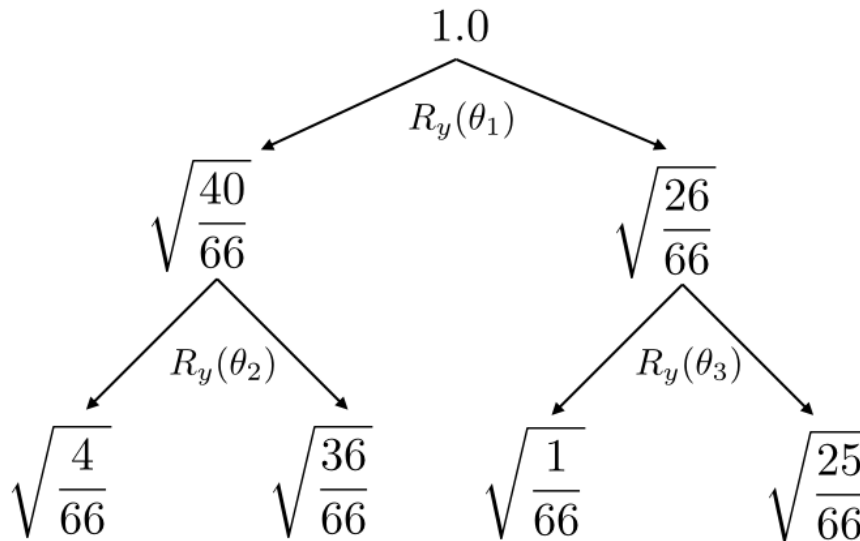


Figure 21: Cosine-Sine Decomposition for the example state

Once the tree has been prepared, to implement it as a quantum circuit we follow these steps:

- i. Start at level 1, with the required number of qubits (here $n = 2$ to store $2^n = 4$ values) all of them set to $|0\rangle$ to begin with,

$$|b\rangle = |0\rangle \otimes |0\rangle. \quad (50)$$

- ii. To reach level 2, we apply a rotation gate $R_y(\theta_1)$ where $\theta_1 = 2\arccos\left(\sqrt{\frac{\text{Left Node}}{\text{Right Node}}}\right)$, to the first qubit, where the action of the rotation gate follows - $R_y(2\theta)|0\rangle = \cos(\theta)|0\rangle + \sin(\theta)|1\rangle$. Therefore,

$$|b\rangle = \left(\sqrt{\frac{40}{66}}|0\rangle + \sqrt{\frac{26}{66}}|1\rangle\right) \otimes |0\rangle. \quad (51)$$

```
int main(){
    //general control set
    int *ctr;
    int C[500]={0};
    ctr = C;
    int n = 2;

    //Read the data to be loaded
    FILE *rhs = fopen("rhs.dat","r");
    for(long p=0;p<pow(2,nu)*2;p=p+1){
        fscanf(rhs,"%lf%lf",&u[p],&u[p+1]);
        printf("%lf %lf \n",u[p],u[p+1]);
        p=p+1;
    }
    fclose(rhs);

    // Re-write it in a format understood by QuOn
    FILE *b;
    b = fopen("b.dat","w");
    for(long v=0; v<(pow(2,nu)-(1-h))*2;v=v+1){
        fprintf(b,"%f+%fi\n",u[v],u[v+1]);
        v=v+1;
    }
    fclose(b);
    //-----
    // First compute the sign outputs for data loading
    circuit sign_check(n);
    circuit *sgncheck;
    sgncheck = &sign_check;

    state psi_sign(n);
    state *psi_sign1;
    psi_sign1 = &psi_sign;

    double * amp_sign = (double*)calloc(pow(2,n)*2,sizeof(double));
    amp_sign[0]=1.0;
    psi_sign.init(amp_sign);
    //set signs for the values
    int * SIGN = (int*)calloc(int(pow(2,n)),sizeof(int));
    double init_norm = CSD_load(n, u);
    coeff_final=coeff_final*init_norm;
    sign_check.LOAD(1,n, SIGN);
    sign_check.run(psi_sign1);
}
```

Figure 22: CSD QuOn- Implementation

- iii. We then follow the same procedure for every level, at every node and apply appropriate $R_y(\theta)$ gates to each qubit that has not been set, by *conditioning the gate on the specific string of qubits that are already set in the previous level*. The sequence of operations would be as follows,

$$|0\rangle \otimes |0\rangle \rightarrow \left(\sqrt{\frac{40}{66}}|0\rangle + \sqrt{\frac{26}{66}}|1\rangle \right) \otimes |0\rangle, \quad (52)$$

$$\left(\sqrt{\frac{40}{66}}|0\rangle + \sqrt{\frac{26}{66}}|1\rangle \right) \otimes |0\rangle \rightarrow \sqrt{\frac{4}{66}}|00\rangle + \sqrt{\frac{36}{66}}|01\rangle + \sqrt{\frac{26}{66}}|10\rangle, \quad (53)$$

$$\sqrt{\frac{4}{66}}|00\rangle + \sqrt{\frac{36}{66}}|01\rangle + \sqrt{\frac{26}{66}}|10\rangle \rightarrow \boxed{\sqrt{\frac{4}{66}}|00\rangle + \sqrt{\frac{36}{66}}|01\rangle + \sqrt{\frac{1}{66}}|10\rangle + \sqrt{\frac{25}{66}}|01\rangle}. \quad (54)$$

- iv. Finally, based on the signs of the values to be stored, one can apply Z gates conditioned on its corresponding basis state to flip the signs appropriately.
- v. The complexity of this algorithm is $O(2^n)$, which is linear in the size of the vector being loaded. An explicit CSD, data loading code on QuOn is shown in figures 22 and 23.

```

// Create a state psi1

state psi1(n);
state *psi1_P;
psi1_P = &psi1;

// Allocate and initialise the amplitudes of psi1
double * A1 = (double*)calloc(pow(2,n)*2,sizeof(double));

// Set values for all n qubits
A1[0]=1.0;
psi1.init(A1);

//Create CSD circuit
circuit CSD(n);
circuit *CSD1;
CSD1 = &CSD;

//Load |b> and set signs for the values

printf("The initial norm factor: %f \n", init_norm);
CSD.LOAD(n-nu,n-1, SIGN);

CSD.drawit();
CSD.run(psi1_P);
psi1.printit();
CSD.num_gates();

//psi1.density();
psi1.write(3);
return 0;

```

Figure 23: CSD QuOn- Implementation

The corresponding circuit that is generated by QuOn, upon running the above code, is shown in Figure 24. However, the state thus produced by any quantum circuit as this, would need to be measured to (a) store it in classical formats, and (b) check the correctness of our quantum circuit. This draws us to the next feature – quantum state tomography.

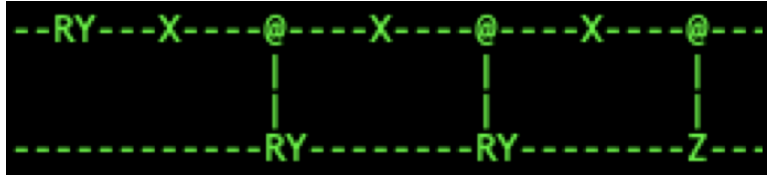


Figure 24: CSD QuOn- Circuit Visualization

(b) **State Tomography:** For almost all practical purposes of CFD simulations, one is interested in extracting the full velocity field, which is stored as amplitudes of the superposition of quantum basis states. This is done via quantum state tomography, whose general idea was described in Section 6.3. In this aspect, QuOn is equipped with:

- i. **State Vector Simulation:** Outputs only the ideal quantum state of the system, without any probabilistic measurements. Though this is not in principle possible on a real device, this feature helps users to design new algorithms by checking its correctness at any stage in the circuit.
- ii. **Amplitude Estimation:** To estimate the state's quantum amplitudes, which are here assumed to be all real numbers (which it is, for all CFD problems), QuOn implements the algorithm due to [75]. Given a state $|b\rangle = \sum_i^{2^n} b_i|i\rangle$, with $\|b\| = 1$, up to a precision δ , estimates a \bar{b} such that $\|b - \bar{b}\| \leq 7\delta$, with a probability $\geq (1 - \frac{1}{20.83n})$ as follows:
 - A. Let $\hat{m} = \frac{36n2^n}{\delta^2}$. First, prepare and measure $|b\rangle$, \hat{m} times in the computational basis. We obtain a probability distribution for each of the basis states $|i\rangle$, with frequencies f_i and probabilities $p_i = \frac{f_i}{\hat{m}}$ respectively. Thus the estimated amplitudes will be $\sqrt{p_i}$ due to Born's rule. However, we have in the process lost the sign information (see C)!
 - B. Measurement: QuOn uses a Monte Carlo type measurement statistics generator, similar to the concept of shots in Qiskit, to sample the states' amplitudes as described above, with a capability of up to 20000 shots.
 - C. Sign Estimation: To estimate and recover the sign information, we append an auxiliary qubit and then prepare a system such that

$$|\phi\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle \otimes (\text{actual state}) + |1\rangle \otimes (\text{estimated state without sign}) \right), \quad (55)$$

$$|\phi\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle \otimes \sum_i^{2^n} b_i|i\rangle + |1\rangle \otimes \sum_i^{2^n} \sqrt{p_i}|i\rangle \right). \quad (56)$$

Now applying a Hadamard gate on the first, auxiliary qubit,

$$|\phi\rangle = \frac{1}{2} \left(\sum_i^{2^n} (b_i + \sqrt{p_i})|0\rangle \otimes |i\rangle + \sum_i^{2^n} (b_i - \sqrt{p_i})|1\rangle \otimes |i\rangle \right). \quad (57)$$

Next, perform another set of \hat{m} measurements on this state. If F_0 and F_1 are the frequencies of $|0\rangle \otimes |i\rangle$ and $|1\rangle \otimes |i\rangle$, respectively, and if $F_0 > 0.4p_i\hat{m}$, then the sign for all basis states, $\text{sgn}_i = 1$; else, the sign for all states $\text{sgn}_i = -1$. Thus the final reconstructed state is given

by

$$|\bar{b}\rangle = \sum_i^{2^n} \text{sgn}_i \sqrt{p_i} \quad (58)$$

Implementing the above with QuOn on our example state, with 20000 shots, we accurately obtain an estimated state reconstruct, as shown in figures 25 and 26, which, respectively, show the final state of the system and the histogram of the probability distribution for the quantum amplitudes.

```
|\psi\rangle = (0.2462 0.00001)|0\rangle + (0.7385 -0.00001)|1\rangle + (0.1231 0.00001)|2\rangle + (0.6155 -0.00001)|3\rangle
```

Figure 25: CSD QuOn- Measured State

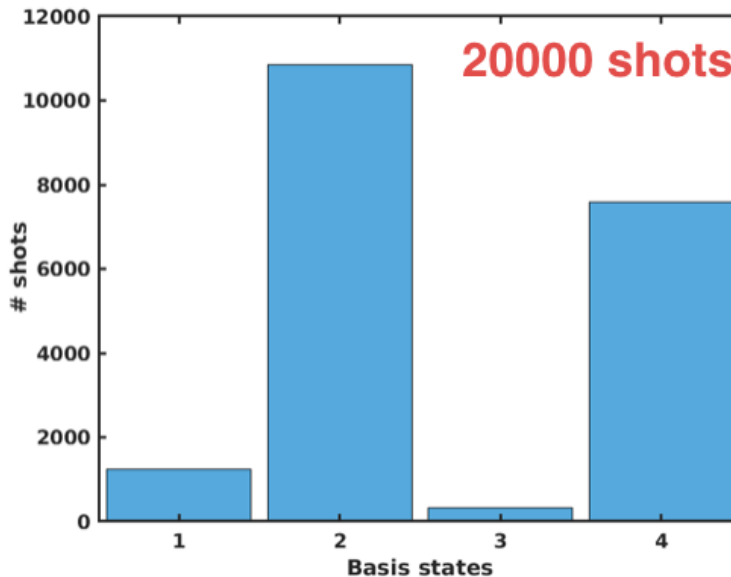


Figure 26: CSD QuOn- Shots histogram

- iii. **State Characteristics:** QuOn also lets the user query, at any point, other standard properties of a quantum state such as the norm and density matrices that characterize a given state.
5. **Algorithms for CFD:** So far, the two important implementations being developed on QuOn to perform CFD simulations are the Quantum Linear System Algorithm(QLSA - described in the next section) and Variational Quantum Eigenvalue Solver(VQE). These algorithms along with standard CFD subroutines such as finite difference, finite element methods etc, renders QuOn as a promising tool for hybrid QCFD simulations. The package along with its documentation will be made public and open-source soon on GitHub. It will come with detailed instructions on features, installation and demonstrations for those interested in exploring this new arena of QCFD further. With this brief outline of QuOn, let us now

examine some preliminary results that were obtained using QuOn along with Qiskit for an example CFD problem.

10.0 QUANTUM LINEAR SYSTEM ALGORITHMS AND AN EXAMPLE

The initial motivation to implement QC was to be able to simulate Hamiltonian dynamics [76] of systems governed by equations of the form $\frac{d\vec{x}}{dt} = A\vec{x}$. More recently, a broader class of problems, which are a set of linear equations of the form $A\vec{x} = \vec{b}$, have been studied; this was initiated with the HHL algorithm [55], leading to a new, broader class of algorithms called Quantum Linear System Algorithms [77]. In CFD, it is rather common in most cases to numerically set up a flow problem, using a suitable finite element/difference/volume method (FEM/FDM/FVM) that could eventually be reduced into solving an $A\vec{x} = \vec{b}$ problem. With the HHL algorithm, assuming that a sparse matrix A and a quantum state \vec{b} are already given to us, we can estimate a solution to \vec{x} , within an error ϵ , with only an $O(\log N)$ complexity, as compared to $O(N)$ of its best classical counterpart, thus promising an exponential speed up. However it is important to note the several theoretical caveats that one would need to address before reaching such a computational advantage [78]. We should note that efforts are continually being made to address these caveats [79, 80]; the QLSA approach itself has evolved greatly with time. For example, some efforts [81–83] have aimed at reducing the error complexity from $poly(1/\epsilon)$ to $poly(\log(1/\epsilon))$ of the underlying Hamiltonian simulation scheme. Consequently, they lead to improved QLSA solvers, catering to linear and nonlinear ODEs and PDEs using both finite element [59, 84–86] and spectral schemes [77], with better performance compared to its ancestor, the HHL algorithm [55]. Here we shall look at the HHL algorithm in more detail and invoke its operation to solve an example CFD problem.

10.1 The HHL Algorithm

Definition: Given a Hermitian and invertible matrix $A \in \mathbb{C}^{2^n \times 2^n}$, a vector $b \in \mathbb{C}^{2^n}$ and a prescribed precision of $\epsilon > 0$, HHL computes a solution x such that $|||x\rangle - |A^{-1}b\rangle|| \leq \epsilon$. For an s -sparse matrix of size N , the classical run time scales as $\mathcal{O}(Ns\kappa \log(1/\epsilon))$, where κ is the condition number and ϵ is the desired accuracy, while the HHL offers a scaling of $\mathcal{O}(\log(N)s^2\kappa^2/\epsilon)$. (It is, however, important to note that it assumes that we are given a Hermitian matrix along with oracles to prepare the matrix A and vector b “efficiently”.)

10.1.1 A simplified outline

1. The matrix equation is set up with qubit states as

$$A|x\rangle = |b\rangle. \quad (59)$$

2. Then spectrally decompose the matrix A in its eigen basis,

$$A = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle\langle u_j|, \quad (60)$$

where u_j is the eigenvector and λ_j is the j th eigenvalue.

3. The inverse of the matrix would be

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle\langle u_j|. \quad (61)$$

4. If we now express the right hand side of equation 59 in the eigen basis of A as $|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle$, we can rewrite the action of A^{-1} on $|b\rangle$ as

$$|x\rangle = A^{-1}|b\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j |u_j\rangle, \quad (62)$$

which is the solution of the matrix equation.

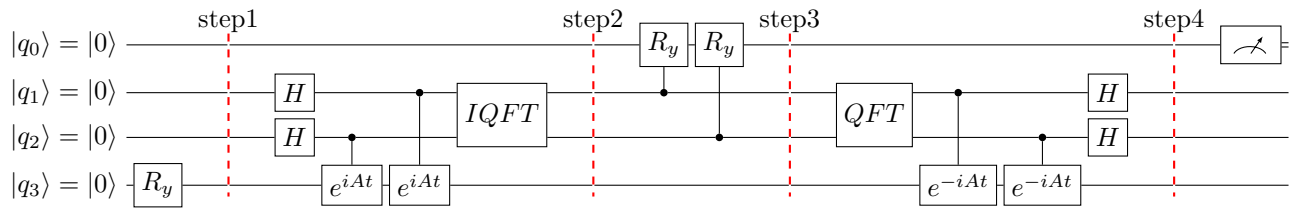


Figure 27: 4 qubit HHL quantum circuit

In what follows, we shall look at the detailed algorithmic steps to implement this procedure, which forms the working principle behind the construction of the quantum circuit of the HHL algorithm, shown in Figure 27.

10.1.2 The algorithm in a bit more detail

For this algorithm, we need a total of three quantum registers: the first register with n_l qubits to store binary representation of the eigen values of A . Also needed is a second register with n_b qubits to store the solution, where $N = 2^{n_b}$. Finally, we need a third register, which contains ancillary qubits that are used as scratch-pad bits and are set to $|0\rangle$ initially.

1. STEP 1: Use CSD to load data/RHS $|b\rangle \in \mathbb{C}^N$ onto the qubit register 2 (q_3),

$$|0\rangle_{n_b} \rightarrow |b\rangle_{n_b}. \quad (63)$$

2. STEP 2: Now we apply the Quantum Phase Estimation (QPE) algorithm to Register 2 (q_{1-2}). In short, for a given linear operator U , if $e^{i\pi\theta}$ is an eigenvalue, QPE basically estimates the phase angle θ in a binary format $|\theta\rangle_{n_l}$. Thus by using the following linear operator we obtain the transformation

$$U = e^{iAt} := \sum_{j=0}^{N-1} e^{i\lambda_j t} |u_j\rangle \langle u_j|. \quad (64)$$

Now expressing it using the eigen basis representation of A , we can write the state of the system as

$$\sum_{j=0}^{N-1} |b_j\rangle_{n_b} |\lambda_j\rangle_{n_l} |u_j\rangle_{n_b}, \quad (65)$$

where $|\lambda_j\rangle_{n_l}$ is the n_l bit representation of λ_j .

3. STEP 3: We concatenate an ancillary qubit (q_0) and then apply a controlled rotation to it conditioned on the eigenvalues λ_j . With C as a constant of our choice (such that $C \leq \lambda_{\min}$) we get

$$\sum_{j=0}^{N-1} |b_j\rangle_{n_b} |\lambda_j\rangle_{n_l} |u_j\rangle_{n_b} \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right). \quad (66)$$

4. STEP 4: We apply an inverse operation QPE[†] on Register 2 to uncompute its values

$$\sum_{j=0}^{N-1} |b_j\rangle_{n_b} |0\rangle_{n_l} |u_j\rangle_{n_b} \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right). \quad (67)$$

5. STEP 5: Finally measuring the ancillary qubit, if the outcome is $|1\rangle$, it denotes a success of the algorithm and we are left with the state

$$\sqrt{\frac{1}{\sum_{j=0}^{N-1} |b_j|^2 / |\lambda_j|^2}} \sum_{j=0}^{N-1} \frac{Cb_j}{\lambda_j} |0\rangle_{n_l} |u_j\rangle_{n_b}, \quad (68)$$

where $\frac{Cb_j}{\lambda_j}$ forms the solution to the matrix equation. However, if we fail (outcome = $|0\rangle$), we repeat starting from Step 1 until success is achieved.

10.2 An Example: 2D Hele-Shaw Flow

As an example, we consider the 2D steady, incompressible Hele-Shaw flow system. For this we use non-dimensional flow parameters by scaling them with their natural scales: Length: $\mathbf{x}^* = \mathbf{x}^*/\bar{L}$; Time: $t^* = t^*/\bar{T}$; Velocity: $\mathbf{u}^* = \mathbf{u}^*/\bar{U}$; Body force: $\mathbf{f}^* = \mathbf{f}^*/\bar{F}$; and Pressure: $p^* = p^*/\bar{P}$. Here, $\bar{L} = D$ (diameter of the pipe), $\bar{P} = \mu\bar{U}/\bar{L}$, $\bar{U} = U_{max}$ (centre line velocity), $\bar{T} = \bar{L}/\bar{U}$ and $\bar{f} = \mu/\bar{L}\bar{T}$, where μ is the viscosity. We set $L = 1$, $P_{in} = 200$, $P_{out} = 100$, $\mu = 1$, and $\rho = 1$. Using these factors and assuming body forces $\mathbf{f} = 0$, we can rewrite the Navier-Stokes equations in its non-dimensional form as

$$Re \left(\frac{\partial \mathbf{u}^*}{\partial t} + (\mathbf{u}^* \cdot \nabla) \mathbf{u}^* \right) = -\nabla p^* + \Delta \mathbf{u}^*, \quad (69)$$

where $Re = (\bar{U}\bar{L})/\nu$ is the Reynolds number. We now transition to the Stokes limit by letting $Re \rightarrow 0$ which, in practice, means $\bar{D} \rightarrow 0$ as in micro-channel flows. Thus, dropping the $*$ notation we obtain the governing equations for the Hele-Shaw flow:

$$\Delta \mathbf{u} - \nabla p = 0, \quad (70)$$

$$\nabla \cdot \mathbf{u} = 0. \quad (71)$$

The set up and boundary conditions are shown for an example of 4×5 grid in Figure 28. The specific combination of Dirichlet and Neumann boundary conditions chosen here represents the case of a developed flow regime with a uniform pressure drop across the domain. To convert this problem into an $\mathbf{Ax} = \mathbf{b}$ form, we first decouple the velocity and pressure terms by taking a divergence of equation 70 as

$$\nabla \cdot (\Delta \mathbf{u} - \nabla p) = 0. \quad (72)$$

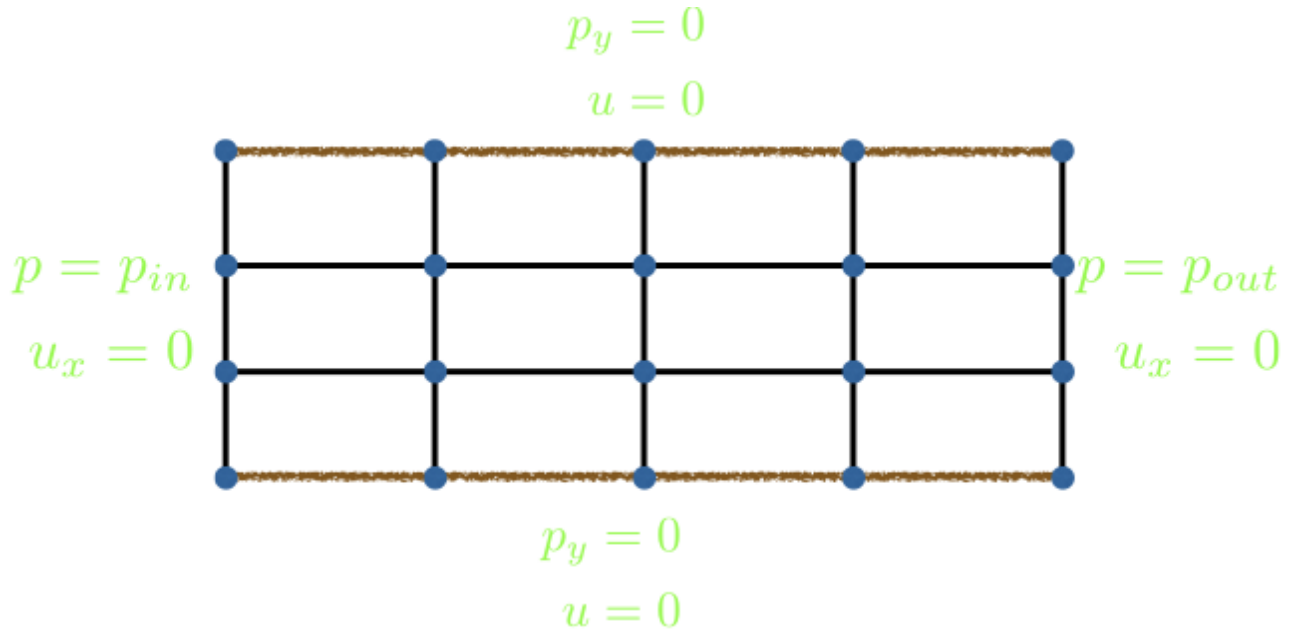


Figure 28: Computational setup and boundary conditions

After a few rearrangements and employing equation 71, we obtain the decoupled equations

$$p_{xx} + p_{yy} = 0, \tag{73}$$

$$u_{xx} + u_{yy} = p_x, \tag{74}$$

$$v_{xx} + v_{yy} = p_y. \tag{75}$$

Equations 73-75 are three Poisson equations that needs to be solved using HHL. To do this, we first discretize the system as seen in figure 28 by employing a central difference scheme (figure 29) for the Laplacian operator as (for $h = L/N$):

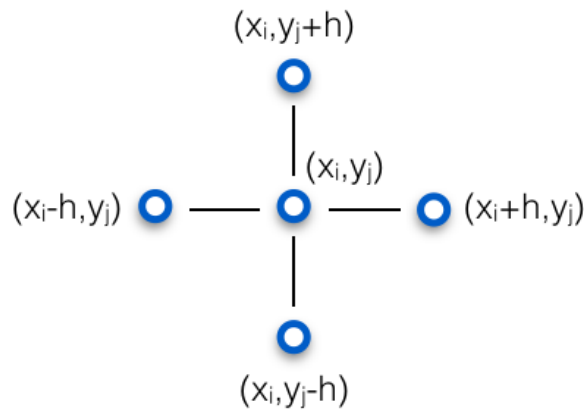


Figure 29: Central difference stencil

$$\Delta u = \frac{u(x_i + h, y_j) + u(x_i, y_j + h) - 4u(x_i, y_j) + u(x_i - h, y_j) + u(x_i, y_j - h)}{h^2} + \text{h.o.t} \quad (76)$$

The ordering of grid points here is *lexicographic*, i.e., we scan through the points first left to right and then row-by-row moving bottom-up. This discretization of the system leads us to Laplacian matrices that are sparse and tri-diagonal, with at most two other far diagonals having non-zero entries (this structure depends on the order of discretization and boundary conditions). The resulting matrices generally have the form

$$A = \begin{bmatrix} -4 & 0 & \dots & 2 & \dots & 0 \\ 0 & -4 & \dots & & \ddots & \vdots \\ \vdots & & \ddots & & & \\ 2 & \dots & & -4 & \dots & 2 \\ \vdots & \ddots & & & \ddots & \\ 0 & 0 & 2 & \dots & & -4 \end{bmatrix} \quad (77)$$

Finally, we set up the matrix equations corresponding to equations 72-74. Invoking the HHL algorithm on QuOn and Qiskit, and inverting the matrix equations and consequently measuring them yields the velocity field.

10.2.1 Numerical procedure

1. Load the values of the $|b\rangle$ vector corresponding to the RHS of equations 72-74 of the matrix equation. This can be done by amplitude loading methods such as CSD as described in Section 9.2. This has a complexity $\sim O(N)$.
2. Construct the Laplacian matrix operators for equations 72-74. With this ready, we are in a position to invoke the HHL algorithm to invert the matrix equations. For instance, this can be done on Qiskit as seen in figures 30-33, which shows the Qiskit python script as well as the equivalent circuit generated. (The description of the code construction on QuOn is skipped here for brevity. However, when the QuOn package is released, users will have access to the source code.) This has complexity of $\mathcal{O}(\log(N)s^2\kappa^2/\epsilon)$.
3. Finally we use quantum state tomography procedure to extract the velocity solutions, with a complexity of $\sim O(\frac{36n2^n}{\delta^2})$.

10.2.2 Results

The results for pressure and velocity solutions thus obtained from both Qiskit and QuOn simulations with 12 qubits are shown in Figures 34-37.

1. For the problem under consideration, a developed Poiseuille flow, an analytic solution is known. The pressure solution is given by a linear pressure drop and the velocity being parabolic according to the equations

$$P = \frac{(p_{out} - p_{in})}{L}x + p_{in} \quad (78)$$

$$U = \frac{0.5}{\mu} \frac{(p_{out} - p_{in})}{L}y(D - y). \quad (79)$$

```
In [1]: from qiskit.aqua import run_algorithm
        from qiskit.aqua.input import LinearSystemInput
        from qiskit.quantum_info import state_fidelity
        from qiskit.aqua.algorithms.classical import ExactLSsolver
        import numpy as np
```

```
In [2]: params = {
        'problem': {
            'name': 'linear_system'
        },
        'algorithm': {
            'name': 'HHL'
        },
        'eigs': {
            'expansion_mode': 'suzuki',
            'expansion_order': 2,
            'name': 'EigsQPE',
            'num_ancillae': 3,
            'num_time_slices': 50
        },
        'reciprocal': {
            'name': 'Lookup'
        },
        'backend': {
            'provider': 'qiskit.BasicAer',
            'name': 'statevector_simulator'
        }
    }
```

```
In [3]: def fidelity(hhl, ref):
        solution_hhl_normed = hhl / np.linalg.norm(hhl)
        solution_ref_normed = ref / np.linalg.norm(ref)
        fidelity = state_fidelity(solution_hhl_normed, solution_ref_normed)
        print("fidelity %f" % fidelity)
```

Figure 30: HHL 1

```
In [3]: def fidelity(hhl, ref):
        solution_hhl_normed = hhl / np.linalg.norm(hhl)
        solution_ref_normed = ref / np.linalg.norm(ref)
        fidelity = state_fidelity(solution_hhl_normed, solution_ref_normed)
        print("fidelity %f" % fidelity)
```

```
In [4]: matrix = [[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                  [0,4,-1,0,0,-2,0,0,0,0,0,0,0,0,0,0],
                  [0,-1,4,0,0,0,-2,0,0,0,0,0,0,0,0,0],
                  [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0],
                  [0,-1,0,0,0,4,-1,0,0,-1,0,0,0,0,0,0],
                  [0,0,-1,0,0,-1,4,0,0,0,-1,0,0,0,0,0],
                  [0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],
                  [0,0,0,0,0,-1,0,0,0,4,-1,0,0,-1,0,0],
                  [0,0,0,0,0,0,-1,0,0,-1,4,0,0,0,-1,0],
                  [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0],
                  [0,0,0,0,0,0,0,0,0,0,-2,0,0,4,-1,0],
                  [0,0,0,0,0,0,0,0,0,0,-2,0,0,-1,4,0],
                  [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
                  ]

vector = [200, 200, 0,0, 200, 200 ,0,0 ,200, 200, 0,0,200,200,0,0 ]
params['input'] = {
    'name': 'LinearSystemInput',
    'matrix': matrix,
    'vector': vector
}
```

Figure 31: HHL 2

```

result = run_algorithm(params)
print("solution ", np.round(result['solution'], 5))

result_ref = ExactLSSolver(matrix, vector).run()
print("classical solution ", np.round(result_ref['solution'], 5))

print("probability %f" % result['probability_result'])
fidelity(result['solution'], result_ref['solution'])

/home/ssb638/anaconda3/lib/python3.7/site-packages/qiskit/aqua/qiskit_aqua.py:119: DeprecationWarning: Declarative API will be removed next Aqua release. Please construct classes and call appropriate methods.
  warnings.warn(aqua_globals.CONFIG_DEPRECATION_MSG, DeprecationWarning)
Input matrix is not hermitian. It will be expanded to a hermitian matrix automatically.

solution [160.5327 +0.j 96.44688-0.j 27.85237+0.j 0.      +0.j 160.5327 +0.j
 150.32492-0.j 41.97797+0.j 0.      +0.j 160.5327 +0.j 150.32492-0.j
 41.97797+0.j 0.      +0.j 160.5327 -0.j 96.44688-0.j 27.85237+0.j
 0.      +0.j]
classical solution [200.      133.33333 66.66667 0.      200.      133.33333 66.66667
 0.      200.      133.33333 66.66667 0.      200.      133.33333
 66.66667 0.      ]
probability 0.112224
fidelity 0.968017

```

Figure 32: HHL 3

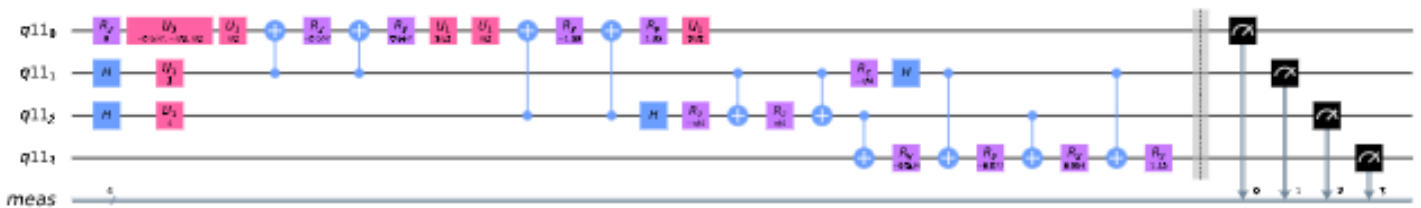


Figure 33: HHL circuit implementation on IBMQ

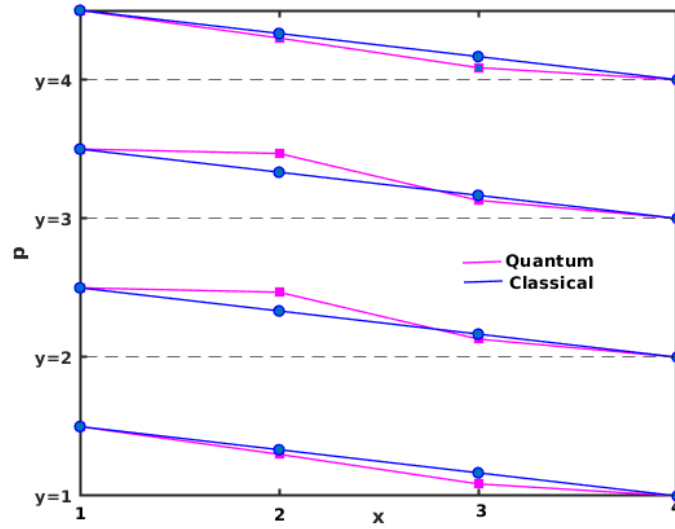


Figure 34: Pressure solution on Qiskit - Fidelity 96.8%

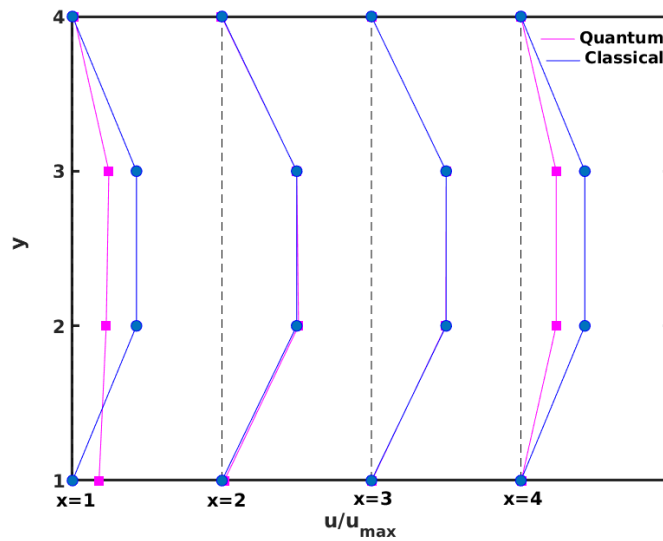


Figure 35: Velocity solution on Qiskit - Fidelity 90.51%

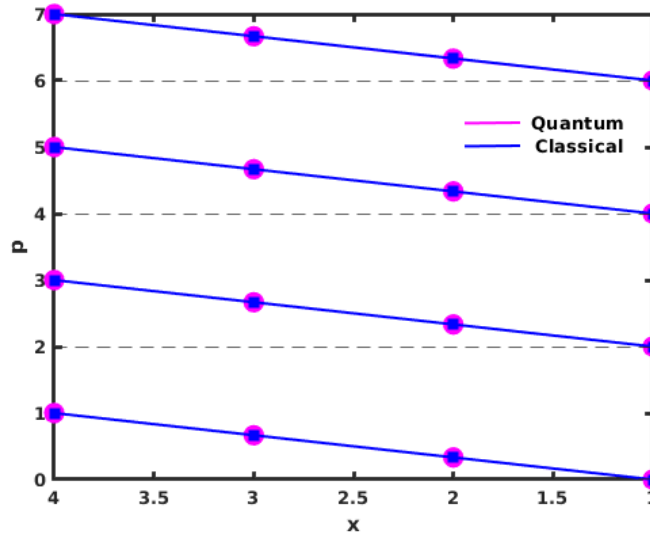


Figure 36: Pressure solution on QuOn - Fidelity 99.99%

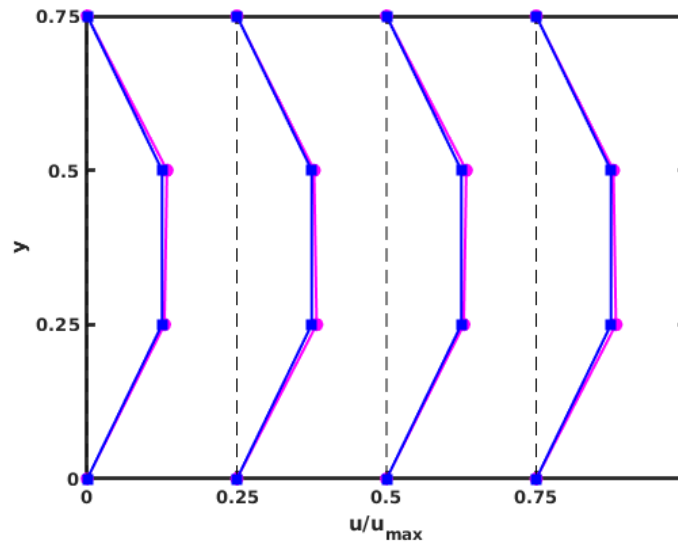


Figure 37: Velocity solution on QuOn - Fidelity 99.97%

2. We observe that even with such limited grid size, the linear pressure profile and parabolic velocity profile results qualitatively meet expectations.
3. The code on Qiskit is slightly more erroneous given that it fails to satisfy boundary conditions for the velocity solutions, as shown in figure 35. Qiskit demonstrates a fidelity (compared to solutions from classical solvers) of 96.8% and 90.51% for pressure and velocity solutions, respectively.
4. On the other hand, the QuOn solutions are more accurate and satisfy boundary conditions. QuOn demonstrates a fidelity of 99.99% and 99.97% for pressure and velocity solutions, respectively.

With a short summary of the lectures, we shall discuss these results somewhat further, outlining the primary obstacles that need to be addressed to make progress with QCFD, in the following section.

11.0 DISCUSSION AND CONCLUDING REMARKS

We have so far discussed and illustrated a selected collection of quantum methods and tools for QCFD simulations. Most of these tools present at least a quadratic speedup, sometimes superpolynomial or exponential, compared to their classical counterparts. This, in itself, is an incentive for the QCFD study. The existence of commercially available QCs such as IBMQ provides an additional thrust to this effort. Let us highlight a few noteworthy observations derived from the example problem that we solved.

1. **Complexity:** The fundamental HHL algorithm modified and implemented in an end-to-end fashion for the Poiseuille flow problem, has an overall complexity of $\sim \mathbb{O}(N^2 \log(N) s^2 \kappa^2 / \epsilon)$. This complexity is comparable with a Gauss-Jordan iteration which has a complexity of $\sim \mathbb{O}(N^3)$. Incorporating more recent versions of QLSA [77] for even a fully non-linear governing equations, if implemented as quantum circuits could provide an end-to-end complexity of $\sim \mathbb{O}(N^2 \text{polylog}(N s \kappa / \epsilon))$, which forms the basis of our ongoing efforts. There are, however, some caveats to which one needs to pay attention in order to achieve even polynomial or exponential speed up.

2. Caveats

- (a) **Non-Hermitian matrices:** The HHL algorithm is prescribed for Hermitian matrices only. However, almost all matrices generated for CFD applications are non-Hermitian. This problem could be circumvented by expanding the matrix into a hermitian matrix, as

$$\tilde{A} = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix} \quad (80)$$

This new hermitian matrix could be solved, by padding the RHS with 0's as

$$\tilde{A} \mathbf{x} = [\mathbf{b}, \mathbf{0}]^T \quad (81)$$

which gives a solution $[\mathbf{0}, \mathbf{x}]^T$. The first half of the vector could be truncated to obtain the solution.

- (b) **State preparation:** One crucial problem that plagues almost all quantum algorithms is the cost of preparing the initial quantum state. In most cases the depth of circuit grows exponentially with the size of the vector which increases exponentially with the number of qubits needed to store it. This is one reason the exponential speedup offered by HHL remains unrealized, as seen in the example problem which has an overall complexity of $\sim \mathbb{O}(N^2 \log(N) s^2 \kappa^2 / \epsilon)$. Any effort that can bring down the complexity of state preparation to sub-linear or logarithmic in vector size is critical for applications.

- (c) **State tomography:** Another critical obstacle that compromises the speedup is measuring the final state of the quantum system. Almost all CFD simulations would need us to extract the entire velocity field, which has again a complexity of *at least* $\sim \mathcal{O}(N)$. Added to this, the measurement scheme itself induces a certain amount of error. In most parts, the development in this direction is dictated by improvements on the hardware end of QC. One possible option could be, instead of a tomography of the entire field, to measure a function of the velocity field such kinetic energy, dissipation, etc., which is a one-shot measurement. If M is a Hermitian matrix corresponding to a physical observable, the expectation value m could be measured from the final quantum state as

$$\langle m \rangle = \langle u | M | u \rangle \quad (82)$$

- (d) **Iterative schemes:** Given that tomography is rather expensive for current applications, iterative schemes and time marching problems will add an extra complexity to the overall algorithm.
- (e) **Matrix pre-conditioning:** Since the QLSA algorithms have at least a linear dependence on the condition number $\mathcal{O}(\kappa)$, it is also advisable to first pre-condition the matrix to reduce the depth of quantum circuit generated.
- (f) **Eigenvalue range:** The fundamental HHL algorithm is designed to solve matrix systems that has eigen values $\in [0, 1]$. It is obvious that this is not the case for any arbitrary problem. However, as shown in [85] and further implemented in QuOn, it seems to be a more optimal choice if the eigenvalues are transformed to lie $\in [-1, -\frac{1}{\kappa}] \cup [\frac{1}{\kappa}, 1]$. It must be stressed that QuOn's implementation solves for all eigenvalues.
3. **Nonlinearity:** The next level of solving QCFD problems is to address nonlinear equations. There have been several recent efforts [61, 86] in this direction. Further possible directions could stem from the following ideas:
- (a) **Multiple copies:** To compute nonlinear terms of quadratic or even higher orders, one could make multiple copies of the velocity field [87] of it—though we are not allowed make copies of a quantum state within the algorithm due to No Cloning Theorem—and store it as a quantum state of two qubits - $[u_1, u_2] \otimes [u_1, u_2] = [u_1^2, u_1 u_2, u_2 u_1, u_2^2]$. This state encodes all possible second order terms. Subsequently, algorithms would need to be designed to evolve the velocity field with the requisite data encoding. However, this method is also known to suffer from certain scaling issues that needs to be taken into account.
- (b) **Linearization and linear embedding:** Where it is possible, linearize the problem using suitable transformations (e.g., Cole-Hopf transformation in the case of Burgers equation) and then remap the transformations back to the original form. Another direction would be to use linear embedding techniques such Carleman or Koopman linearization, where one embeds the nonlinearity in a higher dimensional space to linearize the problem as in [86]. However, it has also been shown in the same work that such methods are amenable for only weakly nonlinear problems.
4. **Error correction:** We wish to bring to the reader's attention that quantum error correction and decoherence reduction methods form a key field of study, whole scope is to reduce noise-related errors and make QC codes more robust and accurate. Though this aspect was not discussed here, it forms another critical obstacle in being able to do meaningful simulations. The QCFD algorithms should not only aim at speedups but also should be hardware-aware and take into consideration the decoherence issues while designing them.

11.1 Summary

1. In Parts 1-2, we provided a brief introduction and motivated the need to transition from CFD to QCFD, reviewed basic concepts of quantum computing and its principles.
2. We then discussed and listed a host of possible quantum algorithms that could be potential candidates for QCFD simulations in both lattice and continuum methods.
3. In Part 3, we introduced a new high-performance quantum simulator QuOn and described its features.
4. Finally, we introduced a QLSA algorithm and discussed its implementation and results on both QuOn and Qiskit.
5. The Key take away is that quantum algorithms are still in their nascent stage. There exist several fundamental obstacles and hence a dire need for clever algorithms that incorporate nonlinearities and error corrections in order to progress QCFD on near-term quantum devices. We listed some potential methods.

In all, for fluid dynamicists, the onset of QC provides an exciting opportunity to study the subject in a completely new way. Progress calls for familiarity with this new paradigm of computing, building and putting together newer and existing quantum algorithms for QCFD solvers. Since we are still in the early stage, it would be wise to perform hybrid computations, where some functions are done on a QPU with the others on a CPU or GPU. Our intention here has been to motivate the pursuit of new directions of computational fluid mechanics that have the potential of a huge impact.

ACKNOWLEDGMENTS

We acknowledge the use of the IBM Q and IBM Q Experience platform for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team.

REFERENCES

- [1] Feynman, R. P., “Simulating physics with computers,” *Int. J. Theor. Phys*, Vol. 21, No. 6/7, 1999.
- [2] Preskill, J., “Quantum Computing in the NISQ era and beyond,” *Quantum*, Vol. 2, 2018, pp. 79.
- [3] Bharadwaj, S. S. and Sreenivasan, K. R., “Quantum Computation of Fluid Dynamics,” *Indian Acad. Sci. Conference Series*, Vol. 101, No. 1, 2020.
- [4] Ishihara, T., Kaneda, Y., Yokokawa, M., Itakura, K., and Uno, A., “Small-scale statistics in high-resolution direct numerical simulation of turbulence: Reynolds number dependence of one-point velocity gradient statistics,” *Journal of Fluid Mechanics*, Vol. 592, 2007, pp. 335–366.
- [5] Iyer, K. P., Scheel, J. D., Schumacher, J., and Sreenivasan, K. R., “Classical 1/3 scaling of convection holds up to $Ra=1015$,” *Proceedings of the National Academy of Sciences*, Vol. 117, No. 14, 2020, pp. 7594–7598.
- [6] Yeung, P. and Ravikumar, K., “Advancing understanding of turbulence through extreme-scale computation: Intermittency and simulations at large problem sizes,” *Physical Review Fluids*, Vol. 5, No. 11, 2020, pp. 110517.

- [7] Orszag, S. A. and Patterson Jr, G., “Numerical simulation of three-dimensional homogeneous isotropic turbulence,” *Physical Review Letters*, Vol. 28, No. 2, 1972, pp. 76.
- [8] Kida, S. and Murakami, Y., “Kolmogorov similarity in freely decaying turbulence,” *The Physics of fluids*, Vol. 30, No. 7, 1987, pp. 2030–2039.
- [9] Yeung, P. and Zhou, Y., “Universality of the Kolmogorov constant in numerical simulations of turbulence,” *Physical Review E*, Vol. 56, No. 2, 1997, pp. 1746.
- [10] Kaneda, Y., Ishihara, T., Yokokawa, M., Itakura, K., and Uno, A., “Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box,” *Physics of Fluids*, Vol. 15, No. 2, 2003, pp. L21–L24.
- [11] Yeung, P., Zhai, X., and Sreenivasan, K. R., “Extreme events in computational turbulence,” *Proceedings of the National Academy of Sciences*, Vol. 112, No. 41, 2015, pp. 12633–12638.
- [12] Ravikumar, K., Appelhans, D., and Yeung, P., “GPU acceleration of extreme scale pseudo-spectral simulations of turbulence using asynchronism,” *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–22.
- [13] Nielsen, M. A. and Chuang, I., “Quantum computation and quantum information,” 2002.
- [14] Rogallo, R. S., *Numerical experiments in homogeneous turbulence*, Vol. 81315, National Aeronautics and Space Administration, 1981.
- [15] Pope, S. B., “Turbulent flows,” 2001.
- [16] Yeung, P., Donzis, D., and Sreenivasan, K., “High-Reynolds-number simulation of turbulent mixing,” *Physics of Fluids*, Vol. 17, No. 8, 2005, pp. 081703.
- [17] Smagorinsky, J., Galperin, B., and Orszag, S., “Large eddy simulation of complex engineering and geophysical flows,” *Evolution of physical oceanography*, 1993, pp. 3–36.
- [18] Meneveau, C. and Katz, J., “Scale-invariance and turbulence models for large-eddy simulation,” *Annual Review of Fluid Mechanics*, Vol. 32, No. 1, 2000, pp. 1–32.
- [19] Davidson, P. A., *Turbulence: an introduction for scientists and engineers*, Oxford university press, 2015.
- [20] Succi, S., Benzi, R., and Higuera, F., “The lattice Boltzmann equation: a new tool for computational fluid-dynamics,” *Physica D: Nonlinear Phenomena*, Vol. 47, No. 1-2, 1991, pp. 219–230.
- [21] Succi, S., *The lattice Boltzmann equation: for fluid dynamics and beyond*, Oxford university press, 2001.
- [22] Meyer, D. A., “From quantum cellular automata to quantum lattice gases,” *Journal of Statistical Physics*, Vol. 85, No. 5-6, 1996, pp. 551–574.
- [23] Meyer, D. A., “Quantum computing classical physics,” *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, Vol. 360, No. 1792, 2002, pp. 395–405.
- [24] Boghosian, B. M. and Taylor, W., “Quantum lattice-gas models for the many-body Schrödinger equation,” *International Journal of Modern Physics C*, Vol. 8, No. 04, 1997, pp. 705–716.

- [25] Boghosian, B. M. and Taylor IV, W., “Quantum lattice-gas model for the many-particle Schrödinger equation in d dimensions,” *Physical Review E*, Vol. 57, No. 1, 1998, pp. 54.
- [26] Boghosian, B. M. and Taylor IV, W., “Simulating quantum mechanics on a quantum computer,” *Physica D: Nonlinear Phenomena*, Vol. 120, No. 1-2, 1998, pp. 30–42.
- [27] Lloyd, S., “Universal quantum simulators,” *Science*, 1996, pp. 1073–1078.
- [28] Abrams, D. S. and Lloyd, S., “Simulation of many-body Fermi systems on a universal quantum computer,” *Physical Review Letters*, Vol. 79, No. 13, 1997, pp. 2586.
- [29] Yepez, J., “Lattice-gas quantum computation,” *International Journal of Modern Physics C*, Vol. 9, No. 08, 1998, pp. 1587–1596.
- [30] Yepez, J., “Quantum lattice-gas model for computational fluid dynamics,” *Physical Review E*, Vol. 63, No. 4, 2001, pp. 046702.
- [31] Yepez, J., “Quantum lattice-gas model for the diffusion equation,” *International Journal of Modern Physics C*, Vol. 12, No. 09, 2001, pp. 1285–1303.
- [32] Yepez, J., “Quantum lattice-gas model for the Burgers equation,” *Journal of Statistical Physics*, Vol. 107, No. 1-2, 2002, pp. 203–224.
- [33] Q, I., “<https://qiskit.org/textbook/preface.html>,” .
- [34] Todorova, B. N. and Steijl, R., “Quantum algorithm for the collisionless Boltzmann equation,” *Journal of Computational Physics*, Vol. 409, 2020, pp. 109347.
- [35] Benzi, R., Succi, S., and Vergassola, M., “The lattice Boltzmann equation: theory and applications,” *Physics Reports*, Vol. 222, No. 3, 1992, pp. 145–197.
- [36] Mezzacapo, A., Sanz, M., Lamata, L., Egusquiza, I., Succi, S., and Solano, E., “Quantum simulator for transport phenomena in fluid flows,” *Scientific reports*, Vol. 5, 2015, pp. 13153.
- [37] Fillion-Gourdeau, F., Herrmann, H., Mendoza, M., Palpacelli, S., and Succi, S., “Formal analogy between the Dirac equation in its Majorana form and the discrete-velocity version of the Boltzmann kinetic equation,” *Physical review letters*, Vol. 111, No. 16, 2013, pp. 160602.
- [38] Sondhi, S. L., Girvin, S., Carini, J., and Shahar, D., “Continuous quantum phase transitions,” *Reviews of modern physics*, Vol. 69, No. 1, 1997, pp. 315.
- [39] Hsieh, T. H., “From d -dimensional Quantum to $d+ 1$ -dimensional Classical Systems,” *Student review*,(2), Vol. 1, 2016.
- [40] Aharony, O., Gubser, S. S., Maldacena, J., Ooguri, H., and Oz, Y., “Large N field theories, string theory and gravity,” *Physics Reports*, Vol. 323, No. 3-4, 2000, pp. 183–386.
- [41] Polyakov, A. M., “Gauge fields and strings,” *Contemp. Concepts Phys.*, Vol. 3, 1987, pp. 1–301.
- [42] Somma, R. D., Batista, C. D., and Ortiz, G., “Quantum approach to classical statistical mechanics,” *Physical review letters*, Vol. 99, No. 3, 2007, pp. 030603.

- [43] Plesch, M. and Brukner, Č., “Quantum-state preparation with universal gate decompositions,” *Physical Review A*, Vol. 83, No. 3, 2011, pp. 032302.
- [44] Shende, V. V., Bullock, S. S., and Markov, I. L., “Synthesis of quantum-logic circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No. 6, 2006, pp. 1000–1010.
- [45] Cortese, J. A. and Braje, T. M., “Loading classical data into a quantum computer,” *arXiv preprint arXiv:1803.01958*, 2018.
- [46] Vogel, K. and Risken, H., “Determination of quasiprobability distributions in terms of probability distributions for the rotated quadrature phase,” *Physical Review A*, Vol. 40, No. 5, 1989, pp. 2847.
- [47] Leonhardt, U., *Measuring the quantum state of light*, Vol. 22, Cambridge university press, 1997.
- [48] Nawrocki, W., “Introduction to quantum metrology,” *Quantum standards and instrumentation*. Springer, Heidelberg, 2015.
- [49] Coles, P. J., Eidenbenz, S., Pakin, S., Adedoyin, A., Ambrosiano, J., Anisimov, P., Casper, W., Chennupati, G., Coffrin, C., Djidjev, H., et al., “Quantum algorithm implementations for beginners,” *arXiv preprint arXiv:1804.03719*, 2018.
- [50] Brassard, G., Høyer, P., and Tapp, A., “Quantum counting,” *International Colloquium on Automata, Languages, and Programming*, Springer, 1998, pp. 820–831.
- [51] Grover, L. K., “Quantum mechanics helps in searching for a needle in a haystack,” *Physical review letters*, Vol. 79, No. 2, 1997, pp. 325.
- [52] Brassard, G., Hoyer, P., Mosca, M., and Tapp, A., “Quantum amplitude amplification and estimation,” *Contemporary Mathematics*, Vol. 305, 2002, pp. 53–74.
- [53] Jordan, S. P., “Fast quantum algorithm for numerical gradient estimation,” *Physical review letters*, Vol. 95, No. 5, 2005, pp. 050501.
- [54] Jordan, S., “<https://quantumalgorithmzoo.org/>,” .
- [55] Harrow, A. W., Hassidim, A., and Lloyd, S., “Quantum algorithm for linear systems of equations,” *Physical review letters*, Vol. 103, No. 15, 2009, pp. 150502.
- [56] Berry, D. W., “High-order quantum algorithm for solving linear differential equations,” *Journal of Physics A: Mathematical and Theoretical*, Vol. 47, No. 10, 2014, pp. 105301.
- [57] Costa, P. C., Jordan, S., and Ostrander, A., “Quantum algorithm for simulating the wave equation,” *Physical Review A*, Vol. 99, No. 1, 2019, pp. 012323.
- [58] Suau, A., Staffelbach, G., and Calandra, H., “Practical Quantum Computing: solving the wave equation using a quantum approach,” *ACM Transactions on Quantum Computing*, Vol. 2, No. 1, 2021, pp. 1–35.
- [59] Cao, Y., Papageorgiou, A., Petras, I., Traub, J., and Kais, S., “Quantum algorithm and circuit design solving the Poisson equation,” *New Journal of Physics*, Vol. 15, No. 1, 2013, pp. 013021.
- [60] Arrazola, J. M., Kalajdzievski, T., Weedbrook, C., and Lloyd, S., “Quantum algorithm for nonhomogeneous linear partial differential equations,” *Physical Review A*, Vol. 100, No. 3, 2019, pp. 032306.

- [61] Gourianov, N., Lubasch, M., Dolgov, S., van den Berg, Q. Y., Babae, H., Givi, P., Kiffner, M., and Jaksch, D., “A quantum-inspired approach to exploit turbulence structures,” *Nature Computational Science*, 2022, pp. 1–8.
- [62] Ruiz-Perez, L. and Garcia-Escartin, J. C., “Quantum arithmetic with the quantum Fourier transform,” *Quantum Information Processing*, Vol. 16, No. 6, 2017, pp. 152.
- [63] Hadfield, S., “Quantum algorithms for scientific computing and approximate optimization,” *arXiv preprint arXiv:1805.03265*, 2018.
- [64] DWave, “https://docs.dwavesys.com/docs/latest/c_gs_2.html,” .
- [65] Xu, G., Daley, A. J., Givi, P., and Somma, R. D., “Turbulent mixing simulation via a quantum algorithm,” *AIAA Journal*, 2018, pp. 687–699.
- [66] Montanaro, A., “Quantum algorithms: an overview,” *npj Quantum Information*, Vol. 2, No. 1, 2016, pp. 1–8.
- [67] Barenghi, C. F., Skrbek, L., and Sreenivasan, K. R., “Introduction to quantum turbulence,” *Proceedings of the National Academy of Sciences*, Vol. 111, No. Supplement 1, 2014, pp. 4647–4652.
- [68] Barenghi, C. F., L’vov, V. S., and Roche, P.-E., “Experimental, numerical, and analytical velocity spectra in turbulent quantum fluid,” *Proceedings of the National Academy of Sciences*, Vol. 111, No. Supplement 1, 2014, pp. 4683–4690.
- [69] Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., Aspuru-Guzik, A., and O’Brien, J. L., “A variational eigenvalue solver on a photonic quantum processor,” *Nature communications*, Vol. 5, 2014, pp. 4213.
- [70] Farhi, E., Goldstone, J., and Gutmann, S., “A quantum approximate optimization algorithm,” *arXiv preprint arXiv:1411.4028*, 2014.
- [71] Battaglia, D. A., Santoro, G. E., and Tosatti, E., “Optimization by quantum annealing: Lessons from hard satisfiability problems,” *Physical Review E*, Vol. 71, No. 6, 2005, pp. 066707.
- [72] Ray, N., Banerjee, T., Nadiga, B., and Karra, S., “Towards Solving the Navier-Stokes Equation on Quantum Computers,” *arXiv preprint arXiv:1904.09033*, 2019.
- [73] Greenberg, O., “Particles with small violations of Fermi or Bose statistics,” *Physical Review D*, Vol. 43, No. 12, 1991, pp. 4111.
- [74] Green, A. S., Lumsdaine, P. L., Ross, N. J., Selinger, P., and Valiron, B., “Quipper: a scalable quantum programming language,” *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*, 2013, pp. 333–342.
- [75] Kerenidis, I. and Prakash, A., “A quantum interior point method for LPs and SDPs,” *ACM Transactions on Quantum Computing*, Vol. 1, No. 1, 2020, pp. 1–32.
- [76] Feynman, R. P., “Simulating physics with computers,” *Feynman and computation*, CRC Press, 2018, pp. 133–153.

- [77] Childs, A. M., Liu, J.-P., and Ostrander, A., “High-precision quantum algorithms for partial differential equations,” *Quantum*, Vol. 5, 2021, pp. 574.
- [78] Aaronson, S., “Read the fine print,” *Nature Physics*, Vol. 11, No. 4, 2015, pp. 291–293.
- [79] Vazquez, A. C. and Woerner, S., “Efficient state preparation for quantum amplitude estimation,” *Physical Review Applied*, Vol. 15, No. 3, 2021, pp. 034027.
- [80] Giovannetti, V., Lloyd, S., and Maccone, L., “Architectures for a quantum random access memory,” *Physical Review A*, Vol. 78, No. 5, 2008, pp. 052310.
- [81] Georgescu, I. M., Ashhab, S., and Nori, F., “Quantum simulation,” *Reviews of Modern Physics*, Vol. 86, No. 1, 2014, pp. 153.
- [82] Berry, D. W., Childs, A. M., Cleve, R., Kothari, R., and Somma, R. D., “Exponential improvement in precision for simulating sparse Hamiltonians,” *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 2014, pp. 283–292.
- [83] Berry, D. W., Childs, A. M., and Kothari, R., “Hamiltonian simulation with nearly optimal dependence on all parameters,” *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, IEEE, 2015, pp. 792–809.
- [84] Ambainis, A., “Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations,” *arXiv preprint arXiv:1010.4458*, 2010.
- [85] Childs, A. M., Kothari, R., and Somma, R. D., “Quantum algorithm for systems of linear equations with exponentially improved dependence on precision,” *SIAM Journal on Computing*, Vol. 46, No. 6, 2017, pp. 1920–1950.
- [86] Liu, J.-P., Kolden, H. Ø., Krovi, H. K., Loureiro, N. F., Trivisa, K., and Childs, A. M., “Efficient quantum algorithm for dissipative nonlinear differential equations,” *Proceedings of the National Academy of Sciences*, Vol. 118, No. 35, 2021.
- [87] Leyton, S. K. and Osborne, T. J., “A quantum algorithm to solve nonlinear differential equations,” *arXiv preprint arXiv:0812.4423*, 2008.