Investigation of quantum algorithms for direct numerical simulation of the Navier-Stokes equations

By K. P. Griffin, S. S. Jain, T. J. Flint AND W. H. R. Chan

1. Motivation and objectives

The potential for exponential speedup of computations by leveraging quantum systems (Feynman 1982) opens up new avenues of possibilities for the direct simulation of complex systems spanning a wide range of scales, including nonlinear dynamics described by the Navier-Stokes equations. In this treatise, we investigate quantum algorithms for the numerical solution of the Navier-Stokes equations for incompressible flows at finite Reynolds numbers.

Previous efforts towards the solution of the Navier-Stokes equations on a quantum architecture have either re-framed the task at hand as a binary optimization problem on a quantum annealer (Ray *et al.* 2019) or employed quantum algorithms to solve a sub-step of the classical algorithm (Steijl & Barakos 2018; Steijl 2019). Other researchers have simulated related partial differential equations (PDEs) using variables stored on multi-qubit nodes through type-II quantum algorithms (Yepez 2001*a*; Berman *et al.* 2002). These various efforts are summarized in Appendix A.

In the present work, we consider storing each velocity variable as an amplitude of a single product basis state, so that n qubits are sufficient to store information for $N = 2^n$ velocity variables. Rather than using a quantum annealing algorithm, we discuss an algorithm for a universal quantum computer, thereby leveraging the full power of quantum computation (the issue of error correction will not be addressed in this work). Also, rather than solving a sub-step of the problem on a quantum computer, we discuss implementing the time evolution operator corresponding to the Navier-Stokes equations directly and in its entirety in the discussed quantum algorithm. During the collection of flow statistics, care should be taken to avoid measuring the full quantum state to preserve the exponential speedup possible on a quantum computer. This implementation may be carried out via either Hamiltonian simulation of a linearized system, similar to the approach adopted by Engel *et al.* (2019) for the solution of the Vlasov equation, or a nonlinear variational algorithm that optimizes a set of variational parameters to approximate the quantum implementation of the time evolution operator, similar to the approach adopted by Lubasch *et al.* (2019) for the solution of the Burgers equation. In the case of the latter, optimization of the variational parameters will require interfacing with a classical computer as well. In that work, the time evolution operator corresponding to the Navier-Stokes equations is implemented as a function of these variational parameters after the optimization is complete, and is not directly transcribed from the equations of motion.

Given a known time evolution operator, the desired velocity variables are obtained via repeated loading of identical initial conditions into the quantum computer, followed by repeated measurement of the quantum system after an identical set of computations,

and then averaging of the recorded measurements using a classical computer. In general, the cost of solving differential equations scales at least with the number of degrees of freedom. Here, measurement of the velocity field to a predetermined precision also scales with the number of degrees of freedom,with an appeal to Holevo's theorem (Holevo 1973). However, it may be that only integral quantities of the solution, such as the coefficients of lift and drag for a simulation of an airplane, are of interest, in which case the full solution field is unnecessary. Instead, a handful of quantities on the order of the number of qubits or less may suffice. For this reason, it may be possible—depending on the quantities of interest—to simulate problems exponentially faster than on a classical computer (Harrow *et al.* 2009).

1.1. *Objectives*

This paper discusses three main considerations for the solution of the Navier-Stokes equations on a universal-gate quantum computer. In Section 2, we discuss how the velocity field can be efficiently stored on and retrieved from a quantum computer, and what limitations this places on our solution procedure to achieve an exponential advantage over classical computers. In Section 3, we discuss the advantages and disadvantages associated with solving the governing equations in physical space or Fourier space. In Section 4, we discuss the treatment of nonlinear terms, which is a fundamental challenge in quantum computation. We close by drawing conclusions and offering some perspective on the solution of the Navier-Stokes equations on quantum computers of the present and near future. This interdisciplinary paper seeks to be accessible to readers from both fluid mechanics and quantum computing communities. To achieve this aim, we recapitulate some fundamental concepts in footnotes and appendices.

If the reader is not familiar with the basic principles of quantum computing, we suggest referring to Appendix B.

2. Representation and retrieval of the velocity field

In the discussed framework, the velocity field—the quantity of interest—is mapped to values between 0 and 1, and normalized such that all values sum to 1. These velocities are represented as squared amplitudes of a set of quantum product basis states (see Appendix B for a quantum primer). Each amplitude, or the corresponding probability, represents a velocity, and in that sense is a deterministic number. The amplitudes are time evolved to a final state using the Navier-Stokes equations for the instantaneous velocity field. The resulting amplitude vector containing the velocities is deterministic but cannot be directly measured. This is because when a quantum state is measured, it collapses to one of the product basis states, rather than the deterministic superposition of product basis states. However, the velocity field is encoded in each of the constituent product basis states. For this reason, we will need to repeat the simulation many times if we want to obtain meaningful flow information. By taking statistics of which product basis state is measured after each run, we can eventually reconstruct the amplitude vector, and thus the velocity field of interest. A quantum circuit may also, in principle, be designed to yield integral quantities of the flow field, such as its mean and higher moments. Multiple measurements will still be needed to reasonably estimate these statistics to a predetermined level of precision, but the number of measurements will conceivably be lower than in the case of determining the full flow field.

Note that these quantum simulations are not independent realizations of turbulence. In other words, their average is not an ensemble or Reynolds average of turbulence.

Rather, these simulations have exactly the same initial conditions. For a theoretical error-correcting or sufficiently fault-tolerant quantum computer, all the simulations should lead to the same amplitude vector before measurement. However, since measurement collapses the quantum state, many simulations are needed to reconstruct this vector, which contains an instantaneous velocity field and not an averaged velocity field.

2.1. *Estimating the upper bound on computational speedup*

We will define an efficient quantum algorithm as one that is exponentially faster than a classical algorithm. A requirement for an efficient algorithm is that it uses a number of qubits, n, that is logarithmic with the number of degrees of freedom in the classical problem, N. A further requirement is that the quantum gates needed to time advance the quantum problem can be approximated by a number of two qubit gates that is polynomial with n (this limits the gate depth and complexity). In this section, we consider the number of qubits required and the number of time steps simulated, which provides an upper bound on the computational speedup of any quantum algorithm. To determine the total cost of the quantum algorithm, we also consider the number of required simulation realizations in Section 2.2 and the gate complexity and gate depth in Section 4.

For a three-dimensional problem on a grid with $M \times M \times M$ points, the mesh size is M^3, or equivalently we can define $N \equiv M^3$. This problem may be solved using $n = 3 \log M = \log N$ qubits. For an incompressible high-Reynolds-number calculation, the convective Courant-Friedrichs-Lewy (CFL)† number typically limits the time step size permitted for the numerical scheme to remain accurate and for an explicit time integration scheme to remain stable. This implies, for a fixed velocity field, that as the mesh is refined, the size of the time step should change proportionally. In the aforementioned three-dimensional problem, the number of time steps scales like M, and the number of qubits is $\mathcal{O}(\log M)$. Meanwhile, the classical problem size is $\mathcal{O}(M^3)$ for each of $\mathcal{O}(M)$ steps. The lower bound on the cost of a quantum simulation is $\mathcal{O}(M \log M)$ (before considering the number of required simulations or the gate depth and complexity) and the lower bound on the cost of a classical simulation is $\mathcal{O}(M^4)$. This means that the quantum computer can at best achieve between a quadratic and a quartic reduction in cost (before considering the number of quantum simulations or gate complexity/depth).

In principle, the time step can be held constant as the grid is refined, but the calculation will become unstable unless an implicit time advancement scheme is used. Furthermore, using an implicit scheme to bypass the smallest timescales of the problem is only justified in special cases where the smallest timescales need not be resolved to accurately solve the problem. For example, this is justified when the spatial mesh is over-resolved. For example, when a cylindrical mesh is used, adequate resolution in the azimuthal dimension can lead to over-resolved cells near the centerline. The second case when the time step can be kept constant with refinement is when only the steady solution is of interest. The third application is in problems that have quasi-steady modes. For example, in a spring mass system with two springs of very different stiffnesses, the stiff spring relaxes very quickly, and this fast timescale need not be captured to solve for the relaxation of the slow spring accurately. In these three cases, the number of time steps does not scale with problem size, so the lower bound on cost is $\mathcal{O}(M^3)$, whereas for the quantum simulation it is $\mathcal{O}(\log M)$. In the best case, and exponential speedup is possible (before considering the number of quantum simulations or gate complexity/depth).

† The definition of the convective CFL$= U\Delta t/\Delta x$, where U is the maximum velocity of the system, Δt is the time step and Δx is the grid spacing.

2.2. *Retrieving a limited set of statistics*

The complexity of initializing a quantum state of size M^3 is $\mathcal{O}(M^6)$ or $\mathcal{O}(4^n)$‡. The complexity of measuring the full state on the classical computer depends on the quantities of interest, as mentioned in Section 1 and earlier in this section. If the full flow field is of interest, then the number of quantum realizations (and thus the number of initializations) is at least N, as pointed out by Harrow *et al.* (2009). To reduce the number of required realizations, we can reduce the number of measured quantities. For example, Harrow *et al.* (2009) only measures $\langle x| W |x\rangle$¶, where $|x\rangle$ is the full quantum state and W is a weight matrix, which takes a weighted average of the whole state and assumes that that is the quantity of interest. In a fluid mechanics application, W might represent a conditional average over a surface if surface fluxes are of interest. In this way, the number of required simulations is reduced by a factor of N. The exact number of simulations depends on the desired precision to which the weighted average is computed. The precision scales inversely with the square root of the number of samples.

If we are taking full advantage of the quantum computer to advance the problem in time without intermediate measurements, and eventually measure only a limited number of quantities, then the problem can in principle be carried out exponentially faster than the classical time advancement of this problem if the time step is held fixed and the gate complexity and depth is polynomial with the number of qubits (this will be further analyzed in Section 4). Observe from the scalings above that the initialization of the data, as well as measurement of the full flow field is of interest, is the bottleneck here given its $\mathcal{O}(M^3)$–$\mathcal{O}(M^6)$ cost. In addition, if we are interested in the intermediate full flow fields, then frequent measurement of the full quantum state and subsequent re-initialization are necessary. Since each measurement and initialization requires an $\mathcal{O}(M^3)$–$\mathcal{O}(M^6)$ operation, the exponential advantage of the quantum algorithm may be lost. This mapping of the system unknowns (velocities) directly to squared amplitudes (as was used by Engel *et al.* (2019) in solving the Vlasov equation) is different from previous research on solving the equations on quantum annealing machines (Chang *et al.* 2019; Ray *et al.* 2019; Srivastava & Sundararaghavan 2019; McClean *et al.* 2016). These prior works have represented the system unknowns (velocities) in binary where each qubit represents a single bit. Note that this approach would be significantly less powerful on a universal quantum computer since the number of velocities that can be stored scales linearly with the number of qubits rather than exponentially as in our approach.

More concretely, problems that involve saving full flow fields at many time steps may not be well suited for the discussed model of computation. For example, a checkpointing algorithm for the solution of adjoint equations would be challenging to efficiently realize on a quantum computer, since it involves a significant amount of reading, initializing, and writing intermediate full flow fields. Meanwhile, in other problems of interest, minimal reading and writing of data might be possible. For example, when an engineer evaluates the statistically stationary turbulent flow over a vehicle, they may be interested in only a few integral measures of the solution such as lift and drag.

‡ Using the algorithm (Shende *et al.* 2006) that is used in Qiskit, a Python-based framework for quantum computing, the number of CNOT gates required to initialize a quantum state scales as 4^n.

¶ Equivalently, this inner product can be notated as $\vec{x}^T W \vec{x}$.

3. Governing equations and their consequences

In this work, we consider the non-dimensional incompressible Navier-Stokes equations† in velocity-pressure form‡,

$$\frac{\partial u_i}{\partial t} = -\frac{\partial u_i u_j}{\partial x_j} - \frac{\partial P}{\partial x_i} + \frac{1}{Re}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right), \tag{3.1}$$

together with the continuity equation

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{3.2}$$

where repeated indices imply summation. In order to directly discretize and time advance the Navier-Stokes equations on a quantum computer in the spirit of a Hamiltonian simulation (Lloyd 1996), the operator that updates the velocity state must be unitary†. Equivalently, we require that the semi-discrete equations take the form

$$\frac{\partial u_i}{\partial t} = A u_i, \tag{3.3}$$

where A is an anti-Hermitian operator. See Appendix C for a proof of this equivalence. Using integration by parts, the time derivative is an anti-Hermitian operator if the boundary term (in the time dimension) is zero. This is true, for example, for a problem that is periodic in time. Similarly, we can show using integration by parts that the first derivative in space is anti-Hermitian if the boundary terms are zero. Physically, terms in the divergence form in the Navier-Stokes equations involve only redistribution of momentum in the case of periodic boundary conditions. However, the anti-Hermiticity of A is not always guaranteed in the case of the Navier-Stokes equations applied to general flows. For this reason, a few variants of the governing equations are considered in the ensuing subsections, with appropriate transformations or assumptions, to enable their solution on a quantum computer. Other approaches that do not require the Navier-Stokes time evolution operator to be directly transcribed into a unitary time evolution operator, such as variational algorithms, are discussed later in Section 4.

3.1. *Governing equations in physical space*

For incompressible flows, the continuity equation, Eq. (3.2), is a constraint on the velocity field. u_i cannot simply be marched forward in time according to Eq. (3.1) because the pressure field must also be solved for, so that time advancement maintains a divergence-free velocity field. To find this pressure, we form a Poisson equation by taking the divergence of Eq. (3.1) and using the continuity equation, Eq. (3.2), to obtain

$$-\frac{\partial^2 P}{\partial x_i \partial x_i} = \frac{\partial^2 u_i u_j}{\partial x_i \partial x_j}. \tag{3.4}$$

† The right-hand side contains the nonlinear, pressure, and viscous terms in that order. The equations are written in conservative (divergence) form, so that it is clear from integration by parts that for problems with periodic boundary conditions, these terms only redistribute momentum within the domain.

‡ Recent work concerning Hamiltonian systems with dissipation (Galley 2013) could potentially result in a set of equations more amenable to solution on a quantum computer.

† A quantum computer is composed of gates that perform unitary operations. To translate a problem to a quantum circuit, it must first be written as an algorithm composed of unitary operations.

Solving this elliptic equation is typically the most expensive step in classical algorithms for the time advancement of the Navier-Stokes equations. Although this system can be large in realistic applications, it is linear so there are efficient iterative methods for solving it on a classical computer. One hybrid quantum-classical approach to solving the Navier-Stokes equations involves solving only the Poisson equation on a quantum computer, while computing the nonlinear and viscous terms in the momentum equation on a classical computer (Cao *et al.* 2012). This avoids the issue of solving a nonlinear problem on a quantum computer, but does not promise an exponential speedup to the classical approaches. The speedup depends on the fraction of the original problem that is spent solving the Poisson equation. A related limitation of this approach is that as a hybrid method, it involves initializing and measuring the full state every time step. This is unfavorable since initializing the right-hand side and measuring the left-hand side of the Poisson equation scales with the problem size rather than the number of qubits.

As we are seeking an exponential speedup for solving the Navier-Stokes equations, we discuss directly time advancing the velocity field on a universal quantum computer, rather than using this hybrid quantum-acceleration model. However, the necessity of solving a Poisson equation that depends on the velocity field makes the solution considerably more complicated. Due to the no-cloning theorem (Park 1970) (see Appendix B), it will be difficult to use the velocity field to compute the right-hand side of the Poisson equation while still preserving the velocity field so that the unitary time advancement operator A can be applied to it. The velocity field may be copied if it is first measured by a classical computer and then re-initialized on two sets of qubits, but the initialization and measurement of the full state incurs a substantial cost. One solution to this problem is to use a compressed representation of the velocity field. Another solution is to avoid having to solve a Poisson equation for pressure by solving the Navier-Stokes equations in Fourier space. In Fourier space, we only need to solve one equation instead of two every time step, obviating the need to copy the velocity field for the solution of an auxiliary equation. This approach is discussed further in Section 3.2.

Another way to avoid solving the Poisson equation is to consider the compressible equations. However, this adds additional complexity (not unlike that of solving the Poisson equation), such that the transport of momentum now depends on the transport equation for density (the compressible continuity equation). That is, it is difficult to obtain u from ρu and ρ while preserving the states of ρu and ρ due to the no-cloning theorem.

3.1.1. *Special case: triply periodic domain in physical space with no external forcing*

Next, we consider the special case of solving the equations in physical space in a triply periodic domain in the absence of external forcing. This case is considered because a simple normalization of the variables is available to make the time advancement operator anti-Hermitian. Here, the volume-integrated momentum in the domain is a constant by Gauss's divergence theorem, since the time rate of change of the volume integral of the momentum equation is only given by boundary terms, which cancel in a periodic domain

$$\frac{1}{\Omega}\int_{\Omega}\frac{\partial u_i}{\partial t}\mathrm{d}\vec{x} = 0. \tag{3.5}$$

First, consider a potential normalization—for just the x-velocity for illustration purposes—that maps all values of u to u' such that the values are between 0 and 1, and they sum to 1. The idea is to add a global value to the velocity field, so that all the velocities

are positive, and then divide by their sum

$$u'(\vec{x}) = \frac{u(\vec{x}) + \max_{\vec{x}} [|u(\vec{x})|]}{\sum_{\vec{x}} \{u(\vec{x}) + \max_{\vec{x}} [|u(\vec{x})|]\}}. \tag{3.6}$$

This normalization has the undesirable property that although u is conserved†, u' is not necessarily conserved, since $\max_{\vec{x}} [|u(\vec{x})|]$ can vary in time. Instead, we propose a more useful normalization, $\tilde{u}$, which involves adding a constant reference velocity, U_{ref}, such that

$$u(\vec{x}) + U_{ref} > 0 \qquad \forall\, \vec{x}, t. \tag{3.7}$$

Now, we redefine the normalized velocity field to be $\tilde{u}$, where

$$\tilde{u}(\vec{x}) = \frac{u(\vec{x}) + U_{ref}}{\sum_{\vec{x}} [u(\vec{x}) + U_{ref}]}, \tag{3.8}$$

which sums to 1 and is conserved if u is conserved‡. By analyzing the kinetic energy equation, one can show that the kinetic energy is not growing in a triply periodic box without external forcing. Hence, the largest velocity achievable for all time can be determined from the initial condition. We set U_{ref} to equal this largest achievable velocity

$$U_{ref} = \sqrt{\sum_{\vec{x}} u_i(\vec{x}) u_i(\vec{x})}, \tag{3.9}$$

which can be physically interpreted as the velocity obtained if all of the kinetic energy were to be concentrated in a single cell in the domain, as this sets the upper bound on the velocity in any cell. This procedure may be generalized to the other velocity components. One important limitation of this normalization is that U_{ref} grows linearly with the problem size. This becomes problematic because in the limit of an infinite problem size, the normalized velocities obey a uniform distribution. To prevent this, a better estimate of U_{ref} is required when the problem size is large. For example, in the decaying turbulence example presented above, a reasonable upper bound for the energy in a given control volume scales with the infinity norm of the energy in the initial condition,

$$U_{ref} \sim \sqrt{\|u_i(\vec{x}) u_i(\vec{x})\|_\infty}. \tag{3.10}$$

For more general problems where the volume-integrated momentum is not constant and the kinetic energy is not necessarily decaying, more general normalizations of the state vector are proposed in other works. For example, Engel *et al.* (2019) require additional qubits to track the magnitude of energy, while Lubasch *et al.* (2019) introduce an additional variational parameter to account for the norm of the vector. Both of these papers will be discussed further in Section 4. However, the normalization approach developed in this section is preferable to those approaches since it incurs no additional computational cost. The reader should keep in mind, though, that this approach should only be applied when the volume-integrated momentum is constant and the volume-integrated energy is decaying.

3.2. *Governing equations in Fourier space*

For periodic domains, an alternative form of the equations is available which removes the need to solve a Poisson equation and greatly simplifies the solution procedure. One

† Discrete conservation of u implies that $\sum_{\vec{x}} u(\vec{x})$ is not a function of time.

‡ In an incompressible fluid, density is a constant, so u is directly related to momentum. Similarly, $u_i u_i/2 = (u^2 + v^2 + w^2)/2$ is directly related to the kinetic energy by the fluid density.

consequence of this transformation is that the nonlinear term in physical space becomes a convolution product in Fourier space.

By taking the Fourier transform of the governing equations, Eqs. (3.1)-(3.2), we obtain

$$\frac{\partial \widehat{u}_i}{\partial t} + \mathrm{i}k_j \widehat{u_j u_i} = -\mathrm{i}k_i \widehat{P} - \frac{1}{Re} k^2 \widehat{u}_i, \tag{3.11}$$

$$\mathrm{i}k_i \widehat{u}_i = 0, \tag{3.12}$$

where k_i represents the components of the wavenumber vector, and k is the corresponding magnitude. $\widehat{(\cdot)}$ denotes the Fourier transform operator. After taking the divergence of Eq. (3.11) and using Eq. (3.12), we obtain a Poisson equation for $\widehat{P}$, which is used to eliminate $\widehat{P}$ in Eq. (3.11) to obtain

$$\frac{\partial \widehat{u}_i}{\partial t} = \mathrm{i}k_j \widehat{u_l u_j} \left(-\delta_{li} + \frac{k_i k_l}{k^2} \right) - \frac{1}{Re} k^2 \widehat{u}_i. \tag{3.13}$$

The Fourier transform of the nonlinear term is written as a convolution in Fourier space

$$\widehat{u_l u_j}(\vec{k}) = \sum_{k_1'} \sum_{k_2'} \sum_{k_3'} \widehat{u}_l(\vec{k}') \widehat{u}_j(\vec{k} - \vec{k}'). \tag{3.14}$$

Note that the convenient normalization discussed for the physical space equations in a triply periodic domain is not applicable in Fourier space. That transformation would require $\widehat{u}_i$ to be conserved, which is not the case. To prove this, consider a box of decaying turbulence. The momentum in physical space is conserved, as just mentioned, so $\widehat{u}_i(\vec{k} = 0)$ is constant. However, the fluctuations about the mean are decaying, so $\widehat{u}_i(\vec{k} \neq 0)$ are decaying to zero. This is not a serious issue since more general normalizations are available, as discussed at the end of Section 3.1.1.

3.2.1. *Spectral methods for general domains*

For non-periodic problems, the use of Fourier decomposition will lead to the Gibbs phenomenon, so other spectral decompositions† are more appropriate. The convergence of the spectral decomposition is improved greatly if the eigenfunction basis is defined on the same domain as the domain of the problem. For example, for a finite dimension—say, a wall-bounded dimension, like the wall-normal dimension in a channel flow—a decomposition in terms of Chebyshev polynomials may be appropriate. For a semi-infinite dimension, such as the wall-normal dimension over a flat plate, Laguerre polynomials may be well suited. For infinite dimensions, Hermite polynomials can be used. Many other decompositions exist and will lead to different clusterings of solution points in physical space, so the optimal decomposition will be problem specific.

On a classical computer, there is a strong preference for using Fourier and Chebyshev decompositions, because fast algorithms are available which scale as $\mathcal{O}(N \log N)$, where N is the problem size. Let n be the number of qubits, which scales as $\log N$. The quantum Fourier transform has a complexity $\mathcal{O}(n^2)$, which is exponentially faster than even the fast Fourier transform. Fast quantum Fourier transforms exist with complexity $\mathcal{O}(n \log n)$, but they are not exact, and so may not be appropriate for scientific computing of chaotic systems such as turbulence. The absence of an exact fast quantum spectral transform means that we should not necessarily prefer Fourier and Chebyshev transforms for a quantum algorithm like we would for a classical algorithm.

† Interested readers can refer to Moin (2010) or Andrews & Phillips (2003) for an introduction to spectral methods.

3.2.2. *Transformation of variables to analytically treat the viscous term*

Rogallo (1977) first proposed that an integrating factor can be used to analytically integrate the viscous term in the Navier-Stokes equations. We suggest using this method when advancing the equations in Fourier space on a quantum computer.

Begin by considering the viscous term. Use the following change of variables

$$\widehat{u}_i = \widehat{v}_i e^{-k^2(t-t_0)/Re}. \tag{3.15}$$

After plugging this ansatz into Eq. (3.13), the viscous term vanishes

$$\frac{\partial \widehat{v}_i}{\partial t} e^{-k^2(t-t_0)/Re} = \mathrm{i}k_j \widehat{u_l u_j} \left(-\delta_{li} + \frac{k_i k_l}{k^2} \right). \tag{3.16}$$

We make the following substitution for the convolution term

$$\widehat{u_l u_j}(\vec{k}) = \sum_{k_1'} \sum_{k_2'} \sum_{k_3'} \widehat{v}_l(\vec{k}') e^{-\|\vec{k}'\|^2(t-t_0)/Re} \widehat{v}_j(\vec{k}-\vec{k}') e^{-\|\vec{k}-\vec{k}'\|^2(t-t_0)/Re} \tag{3.17}$$

$$= \sum_{k_1'} \sum_{k_2'} \sum_{k_3'} \widehat{v}_l(\vec{k}') \widehat{v}_j(\vec{k}-\vec{k}') e^{-(\|\vec{k}-\vec{k}'\|^2+\|\vec{k}'\|^2)(t-t_0)/Re}. \tag{3.18}$$

Notice that the term in parentheses in Eq. (3.16) is merely a constant tensor. This implies that the brunt of computing the right-hand side of this equation is in computing the convolution term, Eq. (3.18).

4. Nonlinearity

Nonlinear products are a fundamental challenge in quantum mechanics†. We discuss two approaches for treating the nonlinear term in the Navier-Stokes equations: a fully nonlinear variational algorithm and a linearized approach.

4.1. *A fully nonlinear approach*

Quantum mechanics is inherently linear, so nonlinear products are difficult to directly treat in a quantum setting (Lubasch *et al.* 2019). A naïve approach to computing a nonlinear product on a quantum computer would involve measuring the full velocity field (which has a size of N, which is 2^n). Then, on the classical computer, the nonlinear product can be performed, which on the classical computer is an $\mathcal{O}(N)$ operation. The quantum computer is then reinitialized with this product. Initialization scales as $\mathcal{O}(N^2)$ (Shende *et al.* 2006). These steps would need to be performed for every time step, so an exponential speedup over a classical computer is unlikely with this approach.

Instead, we suggest using the variational quantum algorithm of Lubasch *et al.* (2019) to perform the nonlinear product. This algorithm is a hybrid quantum-classical algorithm like the naïve algorithm in the previous paragraph, but it is potentially exponentially faster. The approach is to construct the velocity field from a layered network, as shown in Figure 1(b). This network maps the input to the velocity field using d layers of n_g two-qubit gates. Each of the i gates has a parameter λ_i associated with it (where i goes from 1 to n_g). The classical computer provides $\vec{\lambda}$ in a way that attempts to construct the velocity field. To do so, the classical computer iteratively updates $\vec{\lambda}$ by measuring an ancilla qubit‡, and evaluates whether the correct choice of $\vec{\lambda}$ was used by checking if the

† The no-cloning theorem prohibits the copying of an arbitrary quantum state, so we cannot simply multiply a qubit by itself (see Appendix B).

‡ A qubit that doesn't hold simulation variables, but rather is used for algorithmic purposes.

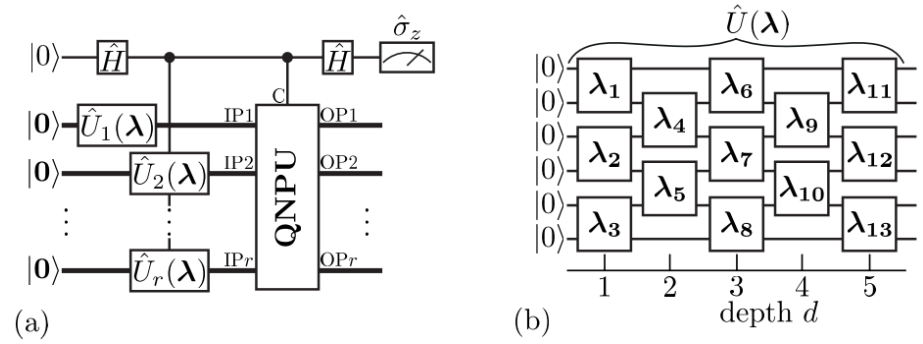

FIGURE 1. (a) Circuit for computing nonlinear product. (b) Layered network. Figure taken from Lubasch *et al.* (2019).

quantity estimated by the ancilla qubit measurements satisfies a desired constraint. The constraint proposed by Lubasch *et al.* (2019) advances the velocity field by a single time step under the forward Euler scheme, but may be generalized to other time advancement schemes. This procedure would then have to be repeated over the desired number of time steps until the final solution is obtained. The efficiency of this algorithm hinges on the magnitude of n_g, which is the length of $\vec{\lambda}$. In the limit that $n_g = N$, this algorithm is no more efficient than the aforementioned naïve algorithm, since N measurements are required at each iteration. However, Lubasch *et al.* (2019) shows that n_g scales like poly(n), since d, the depth of the network, scales like n and not N. For this reason, the cost of approximately initializing the quantum state is poly(n).

Using the variational parameters stored on the classical computer, copies of the quantum state can be sent to the input ports of the quantum nonlinear processing unit (QNPU) (see Figure 1(a)). Lubasch *et al.* (2019) show the upper bound for the depth of the quantum circuit used in the QNPU is also polynomial with the number of qubits. As a result the overall algorithm can be exponentially faster than a classical algorithm (Lubasch *et al.* 2019).

4.1.1. *Data compression*

The variational method of Lubasch *et al.* (2019), discussed in Section 4.1 for treating the nonlinearity in the problem, could potentially also be used for data compression. The layered network was required in their study because it permits the classical computer to store $\mathcal{O}(n)$ variational parameters that represent a solution field of size $\mathcal{O}(2^n)$. This can be interpreted as a lossy, exponential data compression. With the ever-increasing size of numerical simulations, and the potential for quantum computers to be used for even larger simulations in the future, the exponential data compression is valuable.

For any given time step where the full flow field is of interest, $\vec{\lambda}$ can be stored on a classical computer.

Recall that $\vec{\lambda}$ is exponentially smaller than the full flow field and can be readily stored. If the variational algorithm converges, then the stored $\vec{\lambda}$ may be fed into the layered network to reinitialize the full flow field on the quantum computer. Although initializing the flow field only requires poly(n) operations (Lubasch *et al.* 2019), writing the full flow field would require N operations†, so the stored $\vec{\lambda}$ does not provide us classical access

† Holevo's theorem states that n qubits can encode more than n bits of information, but only

to all the velocity variables. However, we can compute statistics or integral measures of the velocity field on the quantum computer, thereby reducing the dimensionality of the desired metrics within the quantum circuit, and can then efficiently output these statistics and measures (see Section 2.2 for details on reducing dimensionality).

For example, suppose we have a data file containing $\vec{\lambda}$. We can reinitialize the velocity field from this small data file. Suppose, further, that the velocity field describes a turbulent flow with small-scale fluctuations. Rather than writing the file and processing it on a classical computer, we could compute statistics, such as the turbulence intensities and Reynolds stress profiles in regions of interest. To access these statistics on a classical computer, the quantum state is measured repeatedly (see Appendix B on measuring these statistics). This process is fast compared to writing the whole flow field, since the dimensionality of these statistics is much smaller than the dimensionality of the entire flow field.

It is important to recognize that this means of exponential data compression is lossy. No estimates for this compression error are provided in Lubasch *et al.* (2019); however, the loss is small enough that they are able to solve Burgers' equation. The loss will be problem specific, as it depends on how optimal the layered-network basis is for representing the solution field (a turbulent flow field). Perhaps a layered network is an efficient way to store turbulence databases, as turbulence is inherently multi-scale in nature.

4.2. *Linearized approach*

There has been significant progress in the solution of linear PDEs on quantum computers. Harrow *et al.* (2009) demonstrated how to solve linear systems on a quantum computer†, which is the main computational requirement for numerically integrating PDEs. One effective approach to solving linear PDEs is given by Engel *et al.* (2019), who solve the Vlasov equation. However, the Navier-Stokes equations are nonlinear. We can linearize about the present state, and then time-advance the linearized problem, but this incurs a linearization error. Analysis of this error is presented in Section 4.2.1. This error is small if the time step is small or the integration time horizon is short. A feasible algorithm would then require frequent updating of the state around which the linearization is performed.

Linearized approaches to solving the Navier-Stokes equations have been used in conjunction with data assimilation for weather prediction (Korn 2009). However, when data are unavailable and high-fidelity predictions over long time horizons are required, the fully nonlinear approach in Section 4.1 should be preferred.

Moreover, even if we tolerate the linearization error, the process of efficiently linearizing a nonlinear equation is not trivial in a quantum setting. It likely requires us to measure the quantum state—so that the gates that implement the linear operator may be updated with the current velocity—and subsequently reinitialize the quantum state‡. Drawing from the ideas of Section 4.1.1, one might potentially avoid measuring the full quantum state by using a layered network to describe and regenerate the quantum state.

4.2.1. *Error incurred by time-advancing the linearized system*

The time evolution of a quantum state $|x\rangle$, with the variables encoded as amplitudes of the constituent product basis states, is unitary for a generic differential equation of

n bits of information are readily accessible (can be measured from a single realization of the quantum state).

† An accessible introduction to the Harrow-Hassidim-Lloyd algorithm is given by Dervovic *et al.* (2018).

‡ See the end of Section 3.1 for an explanation on why measurement is required.

the form

$$\frac{\mathrm{d}\,|x\rangle}{\mathrm{d}t} = A\,|x\rangle\,, \tag{4.1}$$

if A is an anti-Hermitian operator. See Appendix C for a proof of this statement. Now, consider the Taylor expansion of $|x(t)\rangle$ around t_0 as

$$|x(t)\rangle = |x(t_0)\rangle + \left.\frac{\mathrm{d}\,|x\rangle}{\mathrm{d}t}\right|_{t_0} \delta t + \mathcal{O}\{(\delta t)^2\}. \tag{4.2}$$

Use Eq. (4.1) and replace $d\,|x\rangle\,/dt$ by $A(t_0)$ as follows,

$$|x(t)\rangle = |x(t_0)\rangle + A(t_0)\,|x(t_0)\rangle\,\delta t + \mathcal{O}\{(\delta t)^2\} = \{I + A(t_0)\delta t\}|x(t_0)\rangle + \mathcal{O}\{(\delta t)^2\}. \tag{4.3}$$

Now, using Eq. (C 2), and replacing $I + A(t_0)\delta t$ by $U(t_0) + \mathcal{O}\{(\delta t)^2\}$, we obtain

$$|x(t)\rangle = U(t_0)\,|x(t_0)\rangle + \mathcal{O}\{(\delta t)^2\}. \tag{4.4}$$

By neglecting the second-order error term, we obtain an evolution equation for $|x\rangle$ as

$$|x(t)\rangle = U(t_0)\,|x(t_0)\rangle\,, \tag{4.5}$$

which is second-order accurate in time every time step and is also unitary. With this estimate for linearization in mind, we recommend the fully nonlinear approach for general problems.

5. Conclusions

We have discussed some key challenges and potential solutions for the direct numerical simulation of the Navier-Stokes equations on a universal quantum computer. One essential result is that an an efficient quantum algorithm requires that the user be only interested in a reduced set of statistics from the full problem, as a limited number of quantities of interest decrease the total number of quantum simulations required to recover an estimate of those quantities. We are fortunate that in the simulation of turbulent flows, rarely is the instantaneous velocity field of interest, so the cost of estimating the squared amplitudes of the product basis states is greatly reduced. Furthermore, rarely are these statistics sensitive to the exact initial conditions of the flow, which permits approximate initialization.

The solution of nonlinear equations on a quantum computer remains an outstanding problem. One promising approach by Lubasch *et al.* (2019) is discussed here. In that work, the nonlinear term in the Navier-Stokes equations is treated using a variational algorithm. Its success hinges on the use of a layered network to circumvent the high cost of state initialization on the quantum computer. To date, the limitations of this approach have not been well studied, particularly to what extent the information losses (in their layered network representation of the velocity field data) will affect the accuracy of time integration. For this reason this approach is endorsed with some reservation. That work has claimed to have solved the Burgers' equation, but did not report detailed results. This claim is encouraging since that equation is similar to the Navier-Stokes equations but without the pressure term. Either by solving a Poisson equation or by transforming to Fourier space, this algorithm could be extended to the Navier-Stokes equations. These approaches are discussed above. Our overall conclusion on treating nonlinearity is that this remains an unsolved issue and one that should be closely monitored by researchers interested in solving the Navier-Stokes equations on a quantum computer.

One exciting result from using a layered network of Lubasch *et al.* (2019) is the idea of leveraging it for exponential data compression. By storing only the variational parameters, we can regenerate the quantum state on the quantum computer and, although we can not access this exponentially larger data set, we can efficiently extract statistics from it provided that the dimension of the statistics is small compared to the size of the data set. This could have vast implications for scientific computing in the quantum age. Without this sort of exponential data compression, quantum calculations that are exponentially larger than classical contemporaries would be difficult to post-process or restart after a calculation is complete. One downside of this form of storage is that it is lossy, and this information loss needs to be quantified theoretically or at least empirically by applying it to data sets describing turbulent flows.

Next we summarize some secondary recommendations of this paper. We recommend representing velocities as squared amplitudes of product basis states, rather than a binary representation. We discussed using Rogallo's variable transformation to remove the viscous terms from the system of equations, burying their effect in the definition of a transformed variable. We propose a convenient normalization in physical space for triply periodic, unforced simulations. We discuss solving the linearized equations for data assimilation applications where a linearization error is permissible. We also discuss advancing the equations in Fourier space, which replaces the nonlinear term with a convolution term and removes the need to solve a Poisson equation in each time step.

Appendix A. Summary of previous attempts to solve fluid flow problems using a quantum computer

Ray *et al.* (2019) used an adiabatic annealing-based quantum computer, as opposed to a universal-gate quantum computer, to solve the laminar plane channel flow problem. This equation is a reduced form of the Navier-Stokes equations and is an ordinary differential equation for the streamwise velocity. The authors used a finite-difference method to discretize the system and convert the equation into a linear system. Subsequently, they converted the real variables into a binary format, and used a least-squares formulation to transform the linear system into an optimization problem that can be solved on an annealing-based quantum computer. However, the accuracy was not satisfactory, and did not seem to improve with the increase in the number of grid points or precision used in the computation, due to the noise in the system.

Srivastava & Sundararaghavan (2019) presented a solution method for a linear system of equations where the solution procedure involves transforming the linear system into a minimization problem and then mapping the energy functional onto a Ising Hamiltonian on a D-Wave quantum computer.

Xu *et al.* (2018) used probability density function methods for the simulation of turbulent binary scalar mixing process modeled using coalescence/dispersion closure (Pope 1982; Kosály & Givi 1987). They presented a quantum algorithm that is quadratically more efficient in terms of repetitions than Monte Carlo methods—i.e., the number of repetitions N_r required to achieve a precision of ϵ is on the order of $1/\epsilon$ as opposed to $1/\epsilon^2$ that is typical of Monte Carlo methods—and simulated their quantum algorithm on a classical computer by sampling from the same probability distribution as that of the measurement outcome of a quantum computer.

Steijl & Barakos (2018) and Steijl (2019) developed a hybrid quantum-classical computer approach to simulate Navier-Stokes equations. They used a hybrid particle-mesh

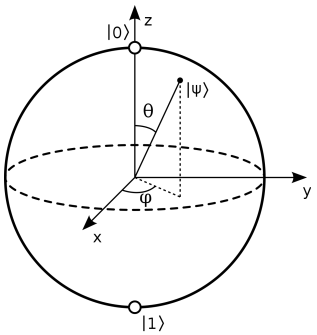

FIGURE 2. Bloch sphere representation of a qubit (wikipedia.org), where $|\psi\rangle$ represents a quantum state, θ is the colatitude, ϕ is the longitude, and $|0\rangle$ and $|1\rangle$ are the basis states.

vortex-in-cell method for the simulation of incompressible Navier-Stokes equations, where the equations are transformed into a transport equation for vorticty, and three Poisson problems for velocity. They modified the vortex-in-cell formulation to use the quantum Fourier transform (QFT) to solve Poisson equations as opposed to standard finite-difference method, to avoid amplification of the noise by the application of finite-difference operations on to the data.

They found that the QFT took up to 20 times longer than the classical fast Fourier transform, though the computational complexities were similar for both. They used up to 24 qubits to simulate colliding vortex rings on up to 256^3 mesh and also assessed the effect of noise on the accuracy of the simulation data.

Yepez (2002) proposed a factorized quantum lattice-gas algorithm for modeling nonlinear Burgers' equation for a type-II quantum computer. A type-II quantum computer is a large parallel array of quantum computers (nodes) connected through classical communication channels, which has the advantage that it can scale indefinitely, as opposed to a type-I conventional globally phase-coherent quantum computer (Yepez 2001*b*).

They showed that the microscopic quantum mechanical model that they proposed reduces to a transport equation—a quantum lattice-Boltzmann equation at the mesoscopic scale. To verify the model, a simulation of the formation of a shock was performed and compared against the analytical solution. The numerical simulation results showed excellent agreement with the analytical solution, thus verifying the algorithm. Furthermore, potential pathways for extending the algorithm to simulate the Navier-Stokes equations in three dimensions by using non-local collisions to apply inter-particle potential were presented along with the cost estimates of simulating a high-Reynolds-number turbulent flow.

Cao *et al.* (2012) used a hybrid quantum-classical approach where the classical computer updates the velocity field and only the linear Poisson equation is solved on the quantum computer. This can be thought of a quantum accelerator model where the most computationally demanding part of the classical calculation is off-loaded to the quantum co-processor. This approach does not promise an exponential speedup of the entire calculation since it still advances the full system on the classical computer.

Appendix B. Qubits, quantum states, and measurement in quantum systems

A qubit, or quantum bit, may be expressed as a superposition of two orthonormal basis states $|0\rangle$ and $|1\rangle$. A single qubit may store a quantum state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$,

where $\alpha, \beta \in \mathbb{C}$, $0 < |\alpha|^2, |\beta|^2 < 1$ and they sum to 1. This may be visualized as a unit vector on the Bloch sphere, as depicted in Figure 2. The north and south poles of the Bloch sphere represent the two standard basis vectors, $|0\rangle$ and $|1\rangle$, respectively. Upon measurement, the qubit will collapse to one of these two vectors. Once the qubit is measured, the original superposition of the two states cannot be recovered. This is called wavefunction collapse, and is an example of the observer effect, where a measurement of a phenomenon inevitably changes the phenomenon. After collapse, the state is like a classical bit, and takes a value of either 0 or 1.

Just before measurement, the probability of collapse to the $|0\rangle$ or $|1\rangle$ state is determined by the colatitude, θ. There is a probability $|\alpha|^2$ of measuring 0 and a probability $|\beta|^2 = 1 - |\alpha|^2$ of measuring 1. To reconstruct this probability distribution, we need to perform this measurement many times, since the wavefunction collapses after a single measurement and cannot be recovered. This reconstruction may be performed by repeatedly initializing the qubit to the state $|\psi\rangle$ and then measuring. Moreover, due to the no-cloning theorem (Park 1970), which implies that a quantum state can not be copied, we can not simply copy the qubit and measure the copies to infer the quantum state. Instead, the process that was used to prefer the qubit in the first place must be repeated for each desired measurement. For example, in the case of solving the Navier-Stokes equations, this might involve solving the equations for every measurement trial.

A quantum state vector can be formed from many qubits. The number of resulting product basis states grows exponentially with the number of qubits used. For example, consider the two-qubit quantum state, $|q_1 q_2\rangle$. Upon measurement, there are four possible product basis states, $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. The number of orthonormal basis states, N, grows exponentially with the number of qubits, n: $N = 2^n$. For each of the N states, there is a probability associated with measuring them. We will use these probabilities to store velocities (the quantity of interest when solving the Navier-Stokes equations). As in the single-qubit case, these probabilities must sum to 1 and must each be $\in [0, 1]$.

To understand the power of present-day quantum computers, consider that the largest quantum computers today have on the order of 70 qubits, so there are 10^{21} accessible product basis states. In principle, such a computer could encode as much information as a mesh with 10^{21} grid points. As a point of reference, the largest simulations today are on the order of 10^{12} grid points (Bermejo-Moreno *et al.* 2013). Most importantly, if we double the number of processors on a classical computer, we can solve a twice-larger problem in the same amount of time, assuming perfect weak scaling. Meanwhile, to double the problem size on a quantum computer, we only need to add a single qubit, because $N = 2^n$. For this reason, we expect the gap between quantum and classical computers to continue to grow exponentially in the future. However, scaling the size of quantum computers is not a simple task due to the inherent practical difficulties in maintaining the environment required for the operation of qubits without incurring errors (MicrosoftQuantumTeam 2018).

Appendix C. Proof that if A in Eq. (4.1) is anti-Hermitian, then the time evolution of $|x\rangle$ is unitary

Consider the solution of Eq. (4.1) for a linear problem, i.e., assuming that the operator A does not depend on $|x\rangle$)

$$|x(t)\rangle = e^{A(t-t_0)}|x(t_0)\rangle. \tag{C 1}$$

Let $U = e^{A\delta t}$, where $\delta t = t - t_0$. The Taylor expansion of U about I is

$$U = I + A\delta t + \frac{A^2(\delta t)^2}{2!} + \frac{A^3(\delta t)^3}{3!} + \dots. \tag{C 2}$$

Taking the conjugate transpose of (C 2) yields

$$U^\dagger = I + A^\dagger \delta t + \frac{(A^2)^\dagger(\delta t)^2}{2!} + \frac{(A^3)^\dagger(\delta t)^3}{3!} + \dots. \tag{C 3}$$

Since A is anti-Hermitian, $A^\dagger = -A$. Hence,

$$U^\dagger = I - A\delta t + \frac{A^2(\delta t)^2}{2!} - \frac{A^3(\delta t)^3}{3!} + \dots. \tag{C 4}$$

Note that $U^\dagger$ is equal to $e^{-A\delta t}$. This gives us

$$UU^\dagger = I. \tag{C 5}$$

Thus, U is a unitary operator, and the time evolution of a quantum state $|x\rangle$ is unitary, since

$$|x(t)\rangle = U|x(t_0)\rangle.$$

Acknowledgments

The authors are grateful to Yihui Quek and Dr. Kazuki Maeda, who provided thoughtful feedback in their revision of this manuscript. K. P. Griffin is funded by a National Defense Science and Engineering Graduate Fellowship. W. H. R. Chan is funded by a National Science Scholarship from the Agency of Science, Technology and Research in Singapore. S. S. Jain is funded by a Franklin P. and Caroline M. Johnson Graduate Fellowship.

REFERENCES

Andrews, L. C. & Phillips, R. L. 2003 *Mathematical Techniques for Engineers and Scientists*. Bellingham, WA: SPIE Press.

Berman, G. P., Ezhov, A. A., Kamenev, D. I. & Yepez, J. 2002 Simulation of the diffusion equation on a type-II quantum computer. *Phys. Rev. A* **66**, 012310.

Bermejo-Moreno, I., Bodart, J., Larsson, J., Barney, B. M., Nichols, J. W. & Jones, S. 2013 Solving the compressible Navier-Stokes equations on up to 1.97 million cores and 4.1 trillion grid points. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, p. 62. ACM.

Cao, Y., Daskin, A., Frankel, S. & Kais, S. 2012 Quantum circuit design for solving linear systems of equations. *Mol. Phys.* **110**, 1675–1680.

Chang, C. C., Gambhir, A., Humble, T. S. & Sota, S. 2019 Quantum annealing for systems of polynomial equations. *Sci. Rep.* **9**, 1–9.

Dervovic, D., Herbster, M., Mountney, P., Severini, S., Usher, N. & Wossnig, L. 2018 Quantum linear systems algorithms: a primer. *arXiv preprint 1802.08227*.

Engel, A., Smith, G. & Parker, S. E. 2019 A quantum algorithm for the Vlasov equation. *arXiv preprint 1907.09418*.

Feynman, R. P. 1982 Simulating physics with computers. *Int. J. Theor. Phys.* **21**, 467–488.

Galley, C. R. 2013 Classical mechanics of nonconservative systems. *Phys. Rev. Lett.* **110**, 174301.

Harrow, A. W., Hassidim, A. & Lloyd, S. 2009 Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **103**, 150502.

Holevo, A. S. 1973 Bounds for the quantity of information transmitted by a quantum communication channel. *Probl. Peredachi Inf.* **9**, 3–11.

Korn, P. 2009 Data assimilation for the Navier-Stokes-α equations. *Physica D* **238**, 1957–1974.

Kosály, G. & Givi, P. 1987 Modeling of turbulent molecular mixing. *Combust. Flame* **70**, 101–118.

Lloyd, S. 1996 Universal quantum simulators. *Science* **273**, 1073–1078.

Lubasch, M., Joo, J., Moinier, P., Kiffner, M. & Jaksch, D. 2019 Variational quantum algorithms for nonlinear problems. *arXiv preprint 1907.09032*.

McClean, J. R., Romero, J., Babbush, R. & Aspuru-Guzik, A. 2016 The theory of variational hybrid quantum-classical algorithms. *New J. Phys.* **18**, 023023.

MicrosoftQuantumTeam 2018 Achieving scalability in quantum computing. `https://cloudblogs.microsoft.com/quantum/2018/05/16/achieving-scalability-in-quantum-computing`.

Moin, P. 2010 *Fundamentals of Engineering Numerical Analysis*. Cambridge University Press.

Park, J. L. 1970 The concept of transition in quantum mechanics. *Foundations of Physics.* **1**, 23–33.

Pope, S. 1982 An improved turbulent mixing model. *Combust. Sci. Technol.* **28**, 131–145.

Ray, N., Banerjee, T., Nadiga, B. & Karra, S. 2019 Towards solving the Navier-Stokes equation on quantum computers. *arXiv preprint 1904.09033*.

Rogallo, R. S. 1977 An ILLIAC program for the numerical simulation of homogeneous incompressible turbulence. *NASA Tech. Memo.*

Shende, V. V., Bullock, S. S. & Markov, I. L. 2006 Synthesis of quantum-logic circuits. *IEEE T. Comput. Aid. D.* **25**, 1000–1010.

Srivastava, S. & Sundararaghavan, V. 2019 Box algorithm for the solution of differential equations on a quantum annealer. *Phys. Rev. A* **99**, 052355.

Steijl, R. 2019 Quantum algorithms for fluid simulations. *IntechOpen*.

Steijl, R. & Barakos, G. N. 2018 Parallel evaluation of quantum algorithms for computational fluid dynamics. *Comput. Fluids* **173**, 22–28.

Xu, G., Daley, A. J., Givi, P. & Somma, R. D. 2018 Turbulent mixing simulation via a quantum algorithm. *AIAA J.* **56**, 687–699.

Yepez, J. 2001*a* Quantum lattice-gas model for computational fluid dynamics. *Phys. Rev. E* **63**, 1–18.

Yepez, J. 2001*b* Type-II quantum computers. *Int. J. Mod. Phys.* **12**, 1273–1284.

Yepez, J. 2002 Quantum lattice-gas model for the burgers equation. *J. Stat. Phys.* **107**, 203–224.