npj | Quantum Information

www.nature.com/npjqi

ARTICLE OPEN

Finding flows of a Navier–Stokes fluid through quantum computing

Frank Gaitan[1]✉

There is great interest in using quantum computers to efficiently simulate a quantum system's dynamics as existing classical computers cannot do this. Little attention, however, has been given to quantum simulation of a classical nonlinear continuum system such as a viscous fluid even though this too is hard for classical computers. Such fluids obey the Navier–Stokes nonlinear partial differential equations, whose solution is essential to the aerospace industry, weather forecasting, plasma magneto-hydrodynamics, and astrophysics. Here we present a quantum algorithm for solving the Navier–Stokes equations. We test the algorithm by using it to find the steady-state inviscid, compressible flow through a convergent-divergent nozzle when a shockwave is (is not) present. We find excellent agreement between numerical simulation results and the exact solution, including shockwave capture when present. Finally, we compare the algorithm's computational cost to deterministic and random classical algorithms and show that a significant speed-up is possible. Our work points to a large new application area for quantum computing with substantial economic impact, including the trillion-dollar aerospace industry, weather-forecasting, and engineered-plasma technologies.

npj Quantum Information (2020)6:61 ; https://doi.org/10.1038/s41534-020-00291-0

INTRODUCTION

In one of the earliest papers motivating the development of quantum computers Feynman[1] pointed out that such computers are better suited to simulate the dynamics of a quantum system than existing classical digital computers. This is because classical computers require computational resources that scale exponentially with the size of the simulated quantum system, while quantum computers do not. It is widely expected that quantum simulation of quantum systems will be a major application area for quantum computing.

Quantum systems are not unique in being difficult to simulate. Many important classical systems exist whose dynamics strongly couple many degrees of freedom over multiple length-scales, making their simulation with classical computers hard. An example is a viscous fluid whose flows satisfy the Navier–Stokes nonlinear partial differential equations[2–5] (PDEs). Solving these PDEs is the primary task for such diverse problems as aerospace flight vehicle design, weather-forecasting, probing the dynamics of turbulence, and determining the magneto-hydrodynamics of plasmas in space and in earth-bound technologies. In spite of its importance, little work has been done on finding quantum algorithms for the Navier–Stokes equations. The earliest work we know of[6–9] is based on a quantum lattice-gas model. We are not aware of any work that examines whether this algorithm might provide a computational advantage over classical algorithms. Recently, new works[10–17] have appeared which also consider quantum algorithms for the Navier–Stokes equations. It is remarkable that so little work has been done on this problem even though the US aerospace industry in 2018 was a nearly one trillion (US) dollar industry[18], while weather-forecasting directly impacts the near-term plans of businesses, militaries, and disaster-relief agencies worldwide.

Here we introduce a quantum algorithm for solving the Navier–Stokes equations. As will be clear from our presentation, a similar treatment is applicable to arbitrary PDEs. Our algorithm thus extends to a general quantum algorithm for solving PDEs. To test our quantum algorithm, we use it to find the steady-state inviscid, compressible flow through a convergent-divergent nozzle. Such nozzles are used in rocket engines and in jet engines for supersonic flight and display a rich range of flows, from smooth acceleration from subsonic to supersonic speeds, to the appearance of a normal shockwave in the divergent part of the nozzle, which causes supersonic flow to decelerate to subsonic speeds as it crosses the narrow shock region. We find excellent agreement between numerical simulation of our algorithm and the exact flow solution, including shockwave capture when present. We then compare the quantum algorithm's computational cost to deterministic and random classical algorithms, and discuss flow regimes where there is a computational advantage. Lastly, we point to the large new application area our quantum PDE algorithm opens up for quantum computing which includes many economically important problems. The quantum algorithm we present is independent of the previously cited papers, and of a quantum algorithm for elliptic PDEs[19].

RESULTS

The Navier–Stokes equations are nonlinear PDEs which express the conservation of mass, linear momentum, and energy of a viscous fluid. They incorporate dissipative effects such as friction between adjacent fluid elements in relative motion (viscosity) and heat flow due to a non-uniform temperature distribution (thermal conductivity). Other dissipative effects can be included but we will not consider them here. We also suppress the effects of gravity, electric and magnetic fields, and volumetric heating sources. The local state of a fluid is specified by the flow velocity $\boldsymbol{v}(\boldsymbol{x},t)$ and two thermodynamic state variables such as mass density $\rho(\boldsymbol{x},t)$ and temperature $T(\boldsymbol{x},t)$. For problems in aerodynamics it is often a good approximation to treat a fluid as a calorically perfect gas[3,4], which is an ideal gas with constant specific heats (see

[1]Laboratory for Physical Sciences, 8050 Greenmead Dr., College Park, MD 20740, USA. ✉email: fgaitan@lps.umd.edu

Supplementary Information Section SI-1). The pressure $p(\mathbf{x},t)$ is then determined through the equation of state $p = \rho RT$, where R is the specific gas constant. With an eye to our later test application, and to limit a proliferation of terms, we write the Navier–Stokes equations[2–5] in one spatial dimension:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho v) = 0. \tag{1A}$$

$$\frac{\partial}{\partial t}(\rho v) + \frac{\partial}{\partial x}\left(\rho v^2 + p - \tau\right) = 0. \tag{1B}$$

$$\frac{\partial}{\partial t}\left[\rho\left(e + \frac{v^2}{2}\right)\right] + \frac{\partial}{\partial x}\left[\rho v\left(e + \frac{v^2}{2}\right) + pv - \tau v + \kappa\frac{\partial T}{\partial x}\right] = 0. \tag{1C}$$

Here v is the local flow velocity; $\tau = (2\mu + \lambda)\partial v/\partial x$ is the Newtonian viscous stress with first (second) viscosity coefficients μ (λ); κ is the thermal conductivity; and $e(\rho, T)$ is the local internal energy. For a particular flow problem, the local internal energy $e(\rho, T)$ and the pressure $p(\rho, T)$ are known functions of mass density and temperature and so can be eliminated from the Navier–Stokes equations. This leaves the three Navier–Stokes equations to be solved for the three unknowns $v(x, t)$, $\rho(x, t)$, and $T(x, t)$. Boundary and initial conditions will be discussed below.

In Supplementary Information Section SI-2 we put the Navier-Stokes (NS) equations in one spatial dimension into conservation form

$$\frac{\partial U}{\partial t} + \frac{\partial F[U, \partial U/\partial x]}{\partial x} = J[U], \tag{2}$$

where $U = (U_1, U_2, U_3)^{\mathrm{T}}$, $F = (F_1, F_2, F_3)^{\mathrm{T}}$, and $J = (J_1, J_2, J_3)^{\mathrm{T}}$ are three-component column vectors for the flow variables, fluxes, and sources, respectively, and T indicates the transpose operation. Formulas for U, F, and J appear in Eqs. (6), (10), and (11) of the Supplementary Information.

Quantum algorithm for Navier–Stokes equations

As with all computational fluid dynamics calculations it is necessary to discretize the NS dynamics[20–22]. We do this by replacing the spatial continuum parameterized by x with a spatial grid of m points $\{x_1, \ldots, x_j, \ldots, x_m\}$, and for simplicity, assume $\Delta x = x_{j+1} - x_j$ is independent of j. This reduces the uncountable number of degrees of freedom $\{\ldots, U(x,t), \ldots\}$ to a finite number $\{U(1,t), \ldots, U(j,t), \ldots, U(m,t)\}$, where $U(j,t) \equiv U(x_j, t)$. As explained in Supplementary Information Section SI-3, by introducing a finite difference approximation for first and second spatial derivatives, we can approximate the spatial derivatives in Eq. (2) by algebraic expressions. Moving these terms to the RHS and combining them with $J[U]$ reduces the NS PDEs to a set of nonlinear ordinary differential equations (ODEs)

$$\frac{dU}{dt} = f(U). \tag{3}$$

Examination of the formulas in Supplementary Information Section SI-2 shows that only $F[U, \partial U/\partial x]$ contains terms that depend on the viscosity coefficients, with these terms being present (absent) for viscous (inviscid) flow. Thus, the presence or absence of viscosity only affects the specific algebraic expression for $F[U, \partial U/\partial x]$, and the same is true for $f(U)$. We will return to this point below. We stress that our procedure for discretizing space is just one possible choice. Alternative discretization procedures are possible and each would lead to a different expression for the driver function $f(U)$. The alternative driver function for a given discretization would be substituted for our driver function in the quantum algorithm described below. This raises the interesting question of how different spatial discretization choices impact the stability of that quantum algorithm. As noted in Supplementary Information Section SI-3, although we have focused on the NS PDEs in one spatial dimension, a similar discretization procedure can be used to reduce an arbitrary set of PDEs to a corresponding set of ODEs.

The significance of the PDE → ODE reduction is that an (almost) optimal quantum algorithm exists for finding an approximate solution to a set of nonlinear ODEs[23]. Thus, by combining spatial discretization and the quantum ODE algorithm we obtain a quantum algorithm for solving any set of PDEs. Note that although ref. [23] appeared in 2006, it had escaped notice for 14 years that its quantum ODE algorithm could be straightforwardly promoted to a quantum PDE algorithm. As we show in Supplementary Information Section SI-4, the quantum ODE algorithm in turn uses the quantum amplitude estimation algorithm[24] (QAEA) to obtain an approximate ODE solution. A summary of the QAEA can be found in Supplementary Information Section SI-6. Our quantum PDE algorithm is thus the second extension in a sequence of quantum algorithms that yield progressively more powerful applications of quantum amplitude estimation: quantum amplitude estimation[24] → quantum ODE solver[23] → quantum PDE solver (this work).

We summarize the application of the quantum ODE algorithm to Eq. (3). A detailed presentation appears in Supplementary Information Section SI-4. Our task is to compute a bounded function $l(j,t)$ which approximates the exact solution $U(j,t)$ for times $0 \le t \le T$ subject to initial condition $l(j,0) = U(j,0) = U_0(j)$. Reference[23] assumes the driver function $f(U)$ has r continuous, bounded partial derivatives, with the components of the rth derivatives satisfying the Hölder condition

$$|\partial^r f^s(U) - \partial^r f^s(U')| \le H||U - U'||^\rho. \tag{4}$$

In Eq. (4): $1 \le s \le 3$ labels the components of $f(U)$; $\partial^r = \partial_1{}^a \partial_2{}^b \partial_3{}^c$ with $a + b + c = r$; $H > 0$; $||v||$ is the maximum norm which is the largest of $\{|v_1|, |v_2|, |v_3|\}$; and $0 < \rho \le 1$. The smoothness of the driver $f(U)$ is parameterized by $q = r + \rho > 0$. Functions satisfying these conditions are known as Hölder class functions[25,26] and are elements of the Hölder space $\mathcal{F}^{r,\rho}$.

To begin, the interval $0 \le t \le T$ is partitioned into n subintervals $I_i = [t_i, t_{i+1}]$ of duration $h = T/n$ using $n + 1$ times $\{t_0 = 0, \ldots, t_n = T\}$. Each I_i is further subdivided into $N_k = n^{k-1}$ sub-subintervals $I_i^{\mathrm{l}} = [t_i^{\mathrm{l}}, t_i^{\mathrm{l}+1}]$ of duration $\overline{h} = h/N_k = T/n^k$ using $N_k + 1$ times $\{t_i{}^0 = t_i, \ldots, t_i{}^{N_k} = t_{i+1}\}$. At each time t_i and grid-point j parameters $y_i(j)$ are introduced which approximate the exact solution $U(j,t_i) \approx y_i(j)$, and provide the initial condition for the approximate solution $l_i(j,t)$ in I_i. In each sub-subinterval $I_i^{\mathrm{l}} = [t_i^{\mathrm{l}}, t_i^{\mathrm{l}+1}]$, $l_i(j,t)$ is represented locally by a truncated Taylor series $l_{i,\mathrm{l}}(j,t)$ about t_i^{l} (see Eq. (17) in the Supplementary Information). By construction, $l_i(j,t)$ is continuous throughout I_i and $l_i(j,t_i) = y_i(j)$. Thus, once the $y_i(j)$ are known, $l_i(j,t)$ is uniquely determined. The $\{y_i(j)\}$ are determined iteratively using the relation (see Supplementary Information Section SI-4)

$$y_{i+1}(j) = y_i(j) + N_k \sum_{l=0}^{N_k - 1} \frac{\overline{h}}{N_k} \int_0^1 du\, f\left(l_{i,\mathrm{l}}(j,u)\right), \tag{5}$$

for $0 \le i \le n - 1$. The QAEA is used to approximately evaluate the second term on the RHS of Eq. (5). Note that this is the only part of the algorithm that requires a quantum computer. A summary of how the QAEA is used to evaluate this term is given in Supplementary Information Section SI-6C. As explained above, for the Navier–Stokes equations, whether a flow is viscous or not only impacts the algebraic expression for $f(U)$. For either type of flow, the QAEA takes the relevant form of $f(U)$ and returns an estimate of the second term on the RHS of Eq. (5). The quantum algorithm thus works for both viscous and inviscid flows. Once the $\{y_i(j)\}$ are known, they determine the $l_i(j,t)$ for $0 \le i \le n - 1$, and so the desired approximate solution $l(j,t) = l_i(j,t)$ for $t \in I_i$ and $0 \le i \le n - 1$. For a Hölder class driver function $f(U) \in \mathcal{F}^{r,\rho}$ the error ϵ in $l(j,t)$ satisfies[23] (with $0 \le t \le T$; $1 \le j \le m$; and $n \ge 5$) $\epsilon = \mathcal{O}(1/n^{a_k})$ with probability $1 - \delta$. Here $a_k = k(q + 1) - 1$ and $q = r + \rho$.

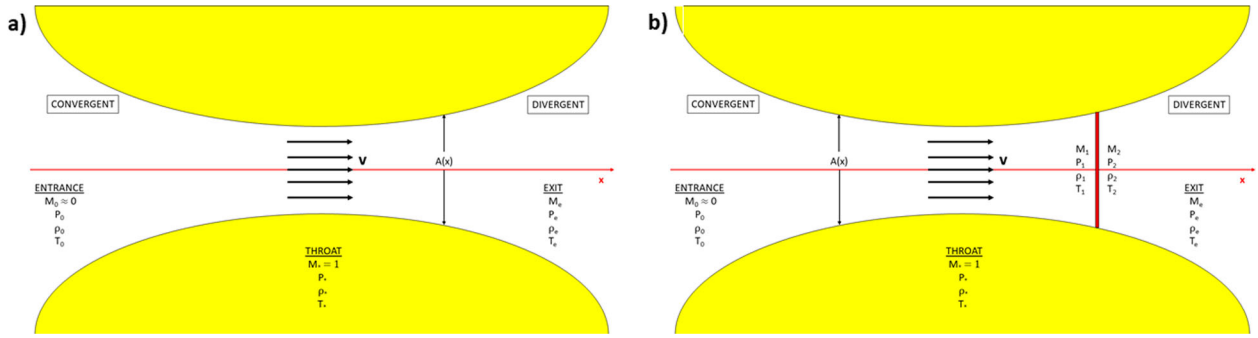

Fig. 1 Flow through a convergent-divergent (de Laval) nozzle. The nozzle profile is specified by its cross-sectional area $A(x)$. Steady-state Mach number $M(x)$, pressure $P(x)$, mass density $\rho(x)$, and temperature $T(x)$ vary with nozzle position x as described in Supplementary Information SI-5B. **a** Absent a shockwave the flow enters at low Mach number $M_0 \approx 0$, is accelerated in the convergent section to sonic speed at the throat ($M_* = 1$), then continues to accelerate to $M_e > 1$ at the nozzle exit. **b** Here the flow contains a normal shockwave (red rectangle) in the divergent section of the nozzle. This can occur for a wide range of exit pressure values P_e. The flow enters at low Mach number $M_0 \approx 0$, is accelerated in the convergent section to sonic speed at the throat, then continues to accelerate until just before reaching the shockwave ($M_1 > 1$). The flow then decelerates to subsonic speed ($M_2 < 1$) in crossing the shockwave and continues to decelerate until reaching the nozzle exit.

In Supplementary Information Section SI-5E we explain how the time partition parameters n and k are assigned values.

Inviscid compressible flow through de Laval nozzle

We were guided by two criteria in choosing a first application for our quantum algorithm: (i) the flow problem should be physically interesting, while (ii) not swamping our first effort with too many technical complications. Although a problem giving rise to turbulent/non-smooth flow, say boundary-layer flow over a flat plate, would satisfy criterion (i), it would violate criterion (ii) as the interesting effect is the transition from laminar to turbulent flow, which produces an adverse pressure gradient and boundary-layer separation. The de Laval nozzle flow problem considered below better balanced our criteria, even though the flow is laminar/smooth, and so not expected to show a quantum speed-up (see the section on "Computational complexity" below). We stress that the purpose of the following simulation is to show that our quantum algorithm does in fact find the non-trivial exact solution of this compressible nozzle flow problem. Although our first application is to an inviscid flow problem, we saw in the previous section that our quantum Navier–Stokes algorithm works for both viscous and inviscid flows. We will examine simulation of viscous flows in future work.

We thus test the quantum Navier–Stokes algorithm by applying it to inviscid ($\lambda = \mu = \kappa = 0$) compressible flow through a convergent-divergent (de Laval) nozzle (see Fig. 1). Such nozzles are used in rocket engines and in jet engines for supersonic flight. We suppose the fluid is air which, for flow temperatures below 1000 K, can be treated as a calorically perfect gas. The nozzle has length L and cross-sectional area $A(x)$ which smoothly decreases (convergent section) in going from the nozzle entrance to the throat, and then increases (divergent section) in continuing on to the nozzle exit. When $A(x)$ is slowly varying the flow is quasi-1 dimensional (see Supplementary Information Section SI-5A). The flow enters the nozzle from a reservoir at small velocity, accelerates in the convergent section to sonic speed at the throat (Mach number $M = v/a = 1$), and in the absence of a shockwave (Fig. 1a), the flow is supersonic throughout the divergent section, reaching maximum speed at the nozzle exit. For a wide range of exit pressure, a normal shockwave forms in the divergent section of the nozzle (Fig. 1b). The flow is supersonic in the divergent section ahead of the shockwave, decelerates to subsonic speed as it crosses the shock region, and continues to decelerate until reaching the nozzle exit. The location of the shockwave is determined by the exit pressure p_e and nozzle exit area A_e. The exact solution for steady-state nozzle flow with and without a shockwave is given in Supplementary Information Section SI-5B.

We used numerical simulation to examine how well the quantum Navier–Stokes algorithm succeeds in finding the steady-state flow through a de Laval nozzle. In Supplementary Information Section SI-5A we describe the derivation of the Navier–Stokes equations in the quasi-1D approximation. As expected, they take the conservation form which allows expressions for U, F, and J to be determined (see Eqs. (24) in the Supplementary Information). We then show how these lead to the driver function $f(U)$ for the Navier–Stokes ODEs (see Eqs. (27) in the Supplementary Information). It proves convenient to introduce dimensionless flow variables which are defined in Supplementary Information section SI-5A. The boundary and initial conditions for the simulation are given, respectively, in Supplementary Information sections SI-5C, D. The simulation evolves the initial condition until the steady-state flow is reached and this is compared to the exact flow solution.

The simulation inputs are: the (dimensionless) nozzle length $L = 3$; the number of spatial grid-points $N_{\text{grid}} = 31(61)$ used when a shockwave is absent (present); the (dimensionless) nozzle area $A(x) = 1 + 2.2(x - 1.5)^2$ for $0 \le x \le 3$; the ratio of specific heats for air $\gamma = 1.4$; the number of time derivatives appearing in the Taylor polynomials $r = 2$; the error $\epsilon_1 = 0.001$ in the QAEA estimate for the average of $f(U)$ over a subinterval I_i; the probability $\delta = 0.001$ that the approximate flow solution exceeds the $\mathcal{O}(1/n^{ak})$ error; the number of Courant–Friedrichs–Lewy (CFL) time-steps (see Supplementary Information Section SI-5E) $N_{\text{tot}} = 1400$ used in all simulations; and the (dimensionless) exit pressure $p_e = 0.6784$ used to produce a normal shock-wave in the divergent section of the nozzle at $x = 2.1$. All simulations used the time partition parameters $n = 16$ and $k = 3$. How these values are assigned is explained in Supplementary Information Section SI-5E. The simulation results and the exact solution for the steady-state flow appear in Fig. 2. The flow without (with) a normal shock-wave appear in Fig. 2a–c (2d–f). The agreement is seen to be excellent, including the location of the shockwave which is clearly visible at $x = 2.1$ in Fig. 2d–f as expected from the exact solution.

Computational complexity analysis

Here, we discuss the computational complexity of finding an approximate solution to a set of PDEs. For both quantum and classical algorithms it is necessary to discretize the problem and we focus on algorithms that reduce the PDEs to the same set of ODEs. The discretization costs are then identical and so the

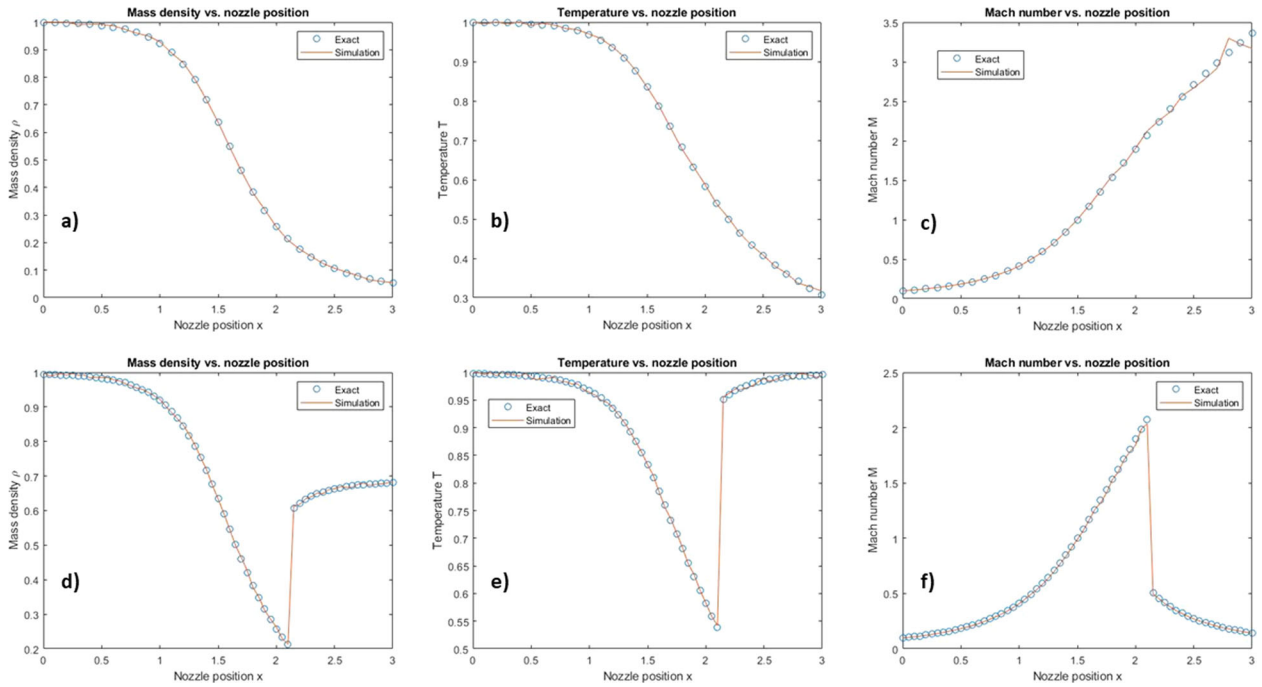

Fig. 2 Comparison of quantum Navier-Stokes simulation results and exact solution for steady-state, inviscid, compressible flow through a de Laval nozzle. The quantum Navier–Stokes simulation results for the steady-state mass density $\rho(x)$, temperature $T(x)$, and Mach number $M(x)$ are presented along with the exact results found in Supplementary Information SI-5B. **a-c (d-f)** show the mass density, temperature, and Mach number, respectively, for flow without (with) a shockwave in the divergent section of the nozzle. The circles are the exact solution values and the solid lines are the simulation results. The agreement is excellent and the shockwave is clearly visible in **d**–**f** at the expected location $x = 2.1$.

comparison of quantum versus classical PDE solvers is determined by the relative complexity of their respective ODE solvers.

To solve ODEs it is necessary to evaluate the driver function $f(U)$. It is assumed that an oracle for $f(U)$ is available. For classical algorithms the oracle can be a black-box/function that evaluates $f(U)$ (and possibly its partial derivatives), while for quantum algorithms the oracle is a query operator whose action on quantum states encodes $f(U)$. The action of the quantum oracle used in the QAEA is given by Eqs. (50) of the Supplementary Information. In both cases, an ODE algorithm's complexity will be measured by the number of oracle calls/queries used to produce an approximate ODE solution. Since each oracle call takes some amount of time, the algorithm complexity is a rough proxy for the time to determine an approximate solution.

The following analysis of ODE algorithms follows Kacewicz[23,27–29]. If an ODE algorithm A returns an approximate solution $l(j,t)$, the algorithm's error for a given $f(U)$ was defined in Supplementary Information Section SI-4 as

$$e(A,f) = \sup_{0\le t\le T;1\le j\le m} \|U(j,t) - l(j,t)\|, \tag{6}$$

where $U(j,t)$ is the exact solution and $\|\dots\|$ is the maximum norm. For a classical deterministic algorithm, its error $e^{\mathrm{det}}(A,\mathcal{F}^{r,\rho})$ over Hölder class functions is then

$$e^{\mathrm{det}}(A,\mathcal{F}^{r,\rho}) = \sup_{f\in\mathcal{F}^{r,\rho}}[e(A,f)]. \tag{7}$$

For a classical random algorithm A, its error over Hölder class functions is

$$e^{\mathrm{ran}}(A,\mathcal{F}^{r,\rho}) = \sup_{f\in\mathcal{F}^{r,\rho}}\sqrt{E\left(e(A,f)^2\right)}, \tag{8}$$

where E denotes expectation value. For a quantum algorithm the Hölder class error is defined probabilistically: given $\delta \in (0, 1/2)$:

$$e^{\mathrm{quan}}(A,\mathcal{F}^{r,\rho},\delta) = \sup_{f\in\mathcal{F}^{r,\rho}}\inf\{\alpha|\mathrm{Prob}(e(A,f)>\alpha)\le\delta\}. \tag{9}$$

Equation (6) is used on the RHS of Eqs. (7–9).

For an algorithm A, its computational cost is defined to be the maximum number of oracle queries needed to compute an approximate solution $l(j,t)$ (over all $f\in\mathcal{F}^{r,\rho}$). For a given $\varepsilon>0$, the ε-complexity $\mathrm{comp}^{t}(\mathcal{F}^{r,\rho},\varepsilon)$ for an algorithm of type t = {deterministic, random, quantum} is the minimum cost over all algorithms A of type t such that

$$e^{t}(A,\mathcal{F}^{r,\rho})<\varepsilon, \tag{10}$$

where the LHS of Eq. (10) is defined in Eqs. (7–9).

Kacewicz[23] introduced a classical random algorithm with the following asymptotic upper bound for its ε-complexity,

$$\mathrm{comp}^{ran}(\mathcal{F}^{r,\rho},\varepsilon) = \mathcal{O}\left(\left(\frac{1}{\varepsilon}\right)^{\frac{1}{q+\frac{1}{2}-\gamma}}\right), \tag{11}$$

where $q = r+\rho$ is the Hölder class smoothness parameter, and the result holds for any $\gamma\in(0,1)$. He also introduced a quantum algorithm which, for $\delta\in(0,1/2)$, has ε-complexity

$$\mathrm{comp}^{\mathrm{quan}}(\mathcal{F}^{r,\rho},\varepsilon,\delta) = \mathcal{O}\left(\left(\frac{1}{\varepsilon}\right)^{\frac{1}{q+1-\gamma}}\right), \tag{12}$$

where we suppress a factor of $\log(1/\delta)$ and γ is as above. Kacewicz[28,29] also found asymptotic lower bounds that apply to all classical random and quantum algorithms, respectively,

$$\mathrm{comp}^{\mathrm{ran}}(\mathcal{F}^{r,\rho},\varepsilon) = \Omega\left(\left(\frac{1}{\varepsilon}\right)^{\frac{1}{q+\frac{1}{2}}}\right) \tag{13A}$$

$$\mathrm{comp}^{\mathrm{quan}}(\mathcal{F}^{r,\rho},\varepsilon,\delta) = \Omega\left(\left(\frac{1}{\varepsilon}\right)^{\frac{1}{q+1}}\right), \tag{13B}$$

where $0<\delta<1/4$. These random and quantum algorithms are thus (almost) optimal, as their respective upper and lower bounds match-up to a small parameter γ in the exponent, and in the quantum case, to within a logarithmic factor. The same then

applies to our quantum PDE algorithm. Finally, Kacewicz[27] found asymptotic tight bounds for worst-case classical deterministic algorithms where the oracle only returns $f(U)$ and/or its partial derivatives:

$$\mathrm{comp}^{\mathrm{det}}(\mathcal{F}^{r,\rho},\varepsilon)=\Theta\left(\left(\frac{1}{\varepsilon}\right)^{\frac{1}{q}}\right). \tag{14}$$

With Eqs. (11–14) in hand we are now ready to compare the ε-complexity of these three types of algorithms. From the above results we see that this complexity is controlled by the Hölder class smoothness parameter q. For smooth functions $f(U)$ which have continuous, bounded derivatives to very high order r, the smoothness parameter $q \gg 1$. On the other hand, for rough/non-smooth functions such as continuous, non-differentiable functions, $r=0$, and $q=\rho\leq 1$. If we insist that $f(U)$ be a Lipschitz function, then $\rho=1$. In the smooth limit ($q \gg 1$) the deterministic, random, and quantum algorithms all have the same complexity $\Theta((1/\varepsilon)^{(1/q)})$ which is a slowly varying function of ε. In this limit the quantum ODE algorithm has no computational advantage/speed-up. However, in the rough limit, the quantum ODE algorithm does give a speed-up. This follows since the exponent in its complexity upper bound is smaller than that in the upper bounds for random and deterministic algorithms. The amount of speed up depends on the Hölder parameter q, the smaller the better. The rough limit speed-up is most pronounced for q, $\gamma \ll 1$. Here the quantum algorithm's complexity is $(1/\varepsilon)$, while a classical random algorithm has complexity $((1/\varepsilon)^2)$. The quantum algorithm thus has a square-root reduction in ε-complexity over the classical random algorithm. This corresponds to a square-root reduction in oracle calls and to a quadratic speed-up in runtime. The speed-up is significantly larger when compared to a classical deterministic algorithm whose complexity is $((1/\varepsilon)^{(1/q)})$ which is exponential in $1/q \gg 1$. The quantum algorithm's complexity, on the other hand, is independent of q and so, in this regime, the speed-up over classical deterministic is exponential.

We see that the quantum ODE algorithm does provide a computational advantage over classical deterministic and random algorithms in the interesting/difficult case of rough/non-smooth driver functions $f(U)$. We have also seen that a rough/non-smooth regime exists ($r=0$; q, $\gamma \ll 1$), where a quadratic (exponential) speed-up is possible over classical random (deterministic) algorithms. To indicate more clearly how large this speed-up can be, if a classical random ODE algorithm uses 1 billion (10^9) oracle calls the quantum ODE algorithm will use approximately 30,000 oracle calls. Similarly, if $q=0.1$ and a classical deterministic algorithm uses one billion oracle calls, the quantum ODE algorithm will use approximately eight oracle calls. As argued at the beginning of this section, the same remarks apply to the quantum PDE algorithm.

DISCUSSION

In this article, we have presented a quantum algorithm for solving arbitrary sets of PDEs and applied it to the Navier–Stokes equations. We tested the resulting quantum Navier–Stokes algorithm by applying it to finding the steady-state inviscid, compressible flow through a de Laval nozzle with and without a shockwave. We found excellent agreement between our numerical simulation results and the exact solutions, including shock-wave capture when present. We examined the computational complexity of the quantum PDE algorithm and compared it to the complexity of classical random and deterministic algorithms. We saw that there is a quantum speed-up for rough/non-smooth flows (which is the computationally interesting/difficult case) and showed that a regime exists, where the speed-up is quadratic (exponential) over classical random (deterministic) algorithms. This suggests that the quantum Navier–Stokes algorithm run on a scalable quantum computer may provide a quantum speed-up for direct numerical simulation of fully-developed turbulence at large Reynolds number which track the flow dynamics from the integral-scale down to the Kolomogorov-scale.

The significance of the quantum Navier–Stokes algorithm lies with a number of economically important problems, whose central task is solving the Navier–Stokes equations: designing aerospace flight vehicles (and their engines) for flight at subsonic to hypersonic speeds; weather-forecasting; determining non-equilibrium high-temperature gas flows; and determining the magneto-hydrodynamics of engineered-plasmas. As pointed out earlier, the aerospace industry is a trillion (US) dollar industry[18]. Finding Navier–Stokes solutions is also central to important scientific problems such as understanding turbulence, and determining stellar structure and dynamics. Furthermore, our quantum PDE algorithm makes possible the quantum simulation of general relativistic systems, as well as interacting-soliton systems, and can be used to study the time-dependent properties of non-equilibrium systems by determining solutions to the Fokker–Planck equation or the Boltzmann equation. The quantum algorithms presented here thus give rise to a large new application area for quantum computing with significant economic and scientific interest: quantum simulation of classical nonlinear/linear continuum systems.

To close we list some important problems for future work. (1) Extending the explicit time-evolution procedure of the quantum algorithm introduced here to implicit methods, which can produce an unconditionally numerically stable quantum PDE algorithm. (2) Incorporating our quantum PDE algorithm into the method of characteristics used to solve hyperbolic PDEs. (3) Applying the quantum Navier–Stokes algorithm to finding the steady-state hypersonic flow through a de Laval nozzle which introduces vibrational modes and chemical reactions into the analysis due to the elevated temperatures that occur. (4) Using the quantum Navier–Stokes algorithm (with a sufficiently fine spatial grid) to estimate the roughness of a turbulent flow, and so the degree of quantum speed-up that might be possible for such flows. (5) Finally, we would like to work out the quantum circuit implementation of our quantum PDE algorithm.

METHODS

In an attempt to make the arguments presented in the "Results" section more transparent, we have at times pointed the reader to sections of the Supplementary Information where technical details could be found. For the reader's convenience, we collect those pointers here so that they are easier to find.

Quantum algorithm

The detailed construction of the quantum Navier–Stokes algorithm is presented in the accompanying Supplementary Information. Specifically, the derivation of the conservation form of the Navier–Stokes equations in 1-spatial dimension is given in Supplementary Information Section SI-2, and the procedure used in the spatial discretization of these equations is described in Supplementary Information Section SI-3. A presentation of Kacewicz' quantum ODE algorithm[23] is given in Supplementary Information Section SI-4, including the derivation of Eq. (5) in the "Results" section. The manner in which the second term on the RHS of Eq. (5) is evaluated using the QAEA is described in Supplementary Information Section SI-6C and the construction of the QAEA is explained in Supplementary Information Section SI-6A, B.

de Laval nozzle flow problem

The details associated with solving the de Laval nozzle flow problem considered in the "Results" section is given in Supplementary Information Section SI-5. The equations of motion in the quasi-1D approximation are derived in Supplementary Information Section SI-5A. This Section also explains how the computational flow variables $U(j,t)$ and driver function $f(U)$ are identified. A careful presentation of the exact solution to this flow

problem is given in Supplementary Information Section SI-5B. Comparison of this solution with the numerical simulation results of the quantum Navier–Stokes algorithm applied to this problem allowed a direct test of the quality of the solution found by the quantum algorithm. The boundary and initial conditions used to determine a unique solution of the equations of motion appear in Supplementary Information Sections SI-5C, D, respectively, and finally, how the time partition parameters n and k are chosen is explained in Supplementary Information Section SI-5E.

DATA AVAILABILITY

All data generated or analyzed during this study are included in this published article.

CODE AVAILABILITY

Computer code used in the numerical simulations is available from the author.

Received: 10 December 2019; Accepted: 19 June 2020;
Published online: 16 July 2020

REFERENCES

1. Feynman, R. P. Simulating physics with computers. *Int. J. Theor. Phys.* **21**, 467–488 (1982).
2. Landau, L. & Lifshitz, E. M. *Fluid Mechanics* 2nd edn (Pergamon, Oxford, 1987).
3. Anderson, J. D. *Fundamentals of Aerodynamics* 6th edn (McGraw Hill, New York, 2017).
4. Anderson, J. D. *Modern Compressible Flow* 3rd edn (McGraw Hill, New York, 2003).
5. White, F. M. *Viscous Fluid Flow* 3rd edn (McGraw Hill, New York, 2006).
6. Yepez, J. Lattice-gas quantum computation. *Int. J. Mod. Phys. C* **9**, 1587–1596 (1998).
7. Yepez, J. Quantum computation of fluid dynamics. In quantum computing and quantum communication. *Lect. Notes Comput. Sci.* **1509**, 34–60 (1999).
8. Yepez, J. Quantum lattice-gas model for computational fluid dynamics. *Phys. Rev. E* **63**, 1–18 (2001).
9. Yepez, J. Quantum lattice-gas model for Burgers equation. *J. Stat. Phys.* **107**, 203–224 (2002).
10. Steijl, R. Quantum algorithms for fluid simulations. *IntechOpen* https://doi.org/10.5772/Intechopen.86685 (2019).
11. Steijl, R. & Barakos, G. N. Parallel evaluation of quantum algorithms for computational fluid dynamics. *Comput. Fluids* **173**, 22–28 (2018).
12. Succi, S. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. (Oxford University Press, New York, 2001).
13. Ray, N., Banerjee, T., Nadiga, B. T., & Karra, S. Towards solving the Navier-Stokes equation on quantum computers. https://arXiv.org/abs/1904.09033 (2019).
14. Mezzacapo, A. et al. Quantum simulator for transport phenomena in fluid flows. *Sci. Rep.* **5**, 13153 (2015).
15. Cao, Y., Papageorgiou, A., Petras, I., Traub, J. & Kais, S. Quantum algorithm and circuit design solving the Poisson equation. *N. J. Phys.* **15**, 013021 (2013).
16. Arrazola, J. M., Kalajdzievski, T., Weedbrook, C. & Lloyd, S. Quantum algorithm for nonhomogeneous linear partial differential equations. *Phys. Rev. A* **100**, 1–9 (2019).
17. Meyers, R. E. & Deacon, K. Simulation of applications in quantum computing. *Proceedings of the Society of Photo-Optical Instrumentation Engineers SPIE 5551, Quantum communications and quantum imagining II*. https://doi.org/10.1117/12.564279 (2004).
18. Reichmann, K. How did the aerospace and defense industry perform in 2018? Defense News, Newsletter. http://defensenews.com/industry/2019/06/10/how-did-the-aerospace-and-defense-industry-perform-in-2018/ (2019).
19. Heinrich, S. The quantum query complexity of elliptic PDE. *J. Complex.* **22**, 220–249 (2006).
20. Pletcher, R. H. Tannehill, J. C. & Anderson, D. A. *Computational Fluid Mechanics and Heat Transfer* 3rd edn. (CRC Press, Boca Raton, 2013).
21. Fletcher, C. A. J. *Computational Techniques for Fluid Dynamics* 2nd edn, Vol. 1 (Springer, New York, 1991).
22. Anderson, J. D. *Computational Fluid Dynamics* (McGraw Hill, New York, 1995).
23. Kacewicz, B. Almost optimal solution of initial-value problems by randomized and quantum algorithms. *J. Complex.* **22**, 676–690 (2006).
24. Brassard, G., Hoyer, P., Mosca, M. & Tapp, A. Quantum amplitude amplification and estimation. http://arXiv.org/quant-ph/0005055 (2000).
25. Evans, L. C. *Partial Differential Equations*. (American Mathematical Society, Providence, 1998).
26. Gilbarg, D. & Trudinger, N. *Elliptic Partial Differential Equations of Second Order*. (Springer, New York, 1983).
27. Kacewicz, B. Optimal solution of ordinary differential equations. *J. Complex.* **3**, 451–465 (1987).
28. Kacewicz, B. Randomized and quantum algorithms yield a speed-up for initial value problems. *J. Complex.* **20**, 821–834 (2004).
29. Kacewicz, B. Improved bounds on randomized and quantum complexity of initial-value problems. *J. Complex.* **21**, 740–756 (2005).

ACKNOWLEDGEMENTS

The author thanks T. Howell III for continued support, and T. R. Govindan for discussions.

COMPETING INTERESTS

The author declares no competing interests.

ADDITIONAL INFORMATION

Supplementary information is available for this paper at https://doi.org/10.1038/s41534-020-00291-0.

Correspondence and requests for materials should be addressed to F.G.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.