Quantum Information Processing (2022) 21:30
https://doi.org/10.1007/s11128-021-03391-8

Solving Burgers' equation with quantum computing

Furkan Oz[1] · Rohit K. S. S. Vuppala[1] · Kursat Kara[1] · Frank Gaitan[2]

Received: 5 July 2021 / Accepted: 14 December 2021 / Published online: 31 December 2021

Abstract
Computational fluid dynamics (CFD) simulations are a vital part of the design process in the aerospace industry. Although reliable CFD results can be obtained with turbulence models, direct numerical simulation of complex bodies in three spatial dimensions (3D) is impracticable due to the massive amount of computational elements. For instance, a 3D direct numerical simulation of a turbulent boundary-layer over the wing of a commercial jetliner that resolves all relevant length scales using a serial CFD solver on a modern digital computer would take approximately 750 million years or roughly 20% of the earth's age. Over the past 25 years, quantum computers have become the object of great interest worldwide as powerful quantum algorithms have been constructed for several important, computationally challenging problems that provide enormous speed-up over the best-known classical algorithms. In this paper, we adapt a recently introduced quantum algorithm for partial differential equations to Burgers' equation and develop a quantum CFD solver that determines its solutions. We used our quantum CFD solver to verify the quantum Burgers' equation algorithm to find the flow solution when a shockwave is and is not present. The quantum simulation results were compared to: (i) an exact analytical solution for a flow without a shockwave; and (ii) the results of a classical CFD solver for flows with and without a shockwave. Excellent agreement was found in both cases, and the error of the quantum CFD solver was comparable to that of the classical CFD solver.

Keywords Quantum algorithms · Computational fluid dynamics · Fluid mechanics · Burgers' equation

✉ Kursat Kara
kursat.kara@okstate.edu

[1] School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK 74078, USA

[2] Laboratory for Physical Sciences, 8050 Greenmead Dr., College Park, MD 20740, USA

1 Introduction

Computational fluid dynamics (CFD) plays an essential role in the design of flight vehicles. It provides detailed, accurate predictions of the airflow about such vehicles, and it is a cost-effective alternative to expensive wind tunnel testing. However, high-resolution CFD simulations of complex flows can take weeks or more to complete with existing supercomputers. Finding ways to speedup CFD simulations is a perennial problem.

Over the past 25 years, there has been growing interest in standing up a technology that will allow the construction of a robust, scalable quantum computer. The engineering challenge for quantum computer hardware is to reliably generate quantum entanglement and quantum state superposition in a scalable manner, while protecting these effects from decoherence, so that they can be exploited by quantum algorithms. If this can be done, quantum algorithms exist which, when run on a quantum computer, can speedup many important, computationally challenging problems. Perhaps the best-known example is Shor's quantum factoring algorithm [20] which provides an exponential speedup over the best existing classical factoring algorithms and, consequently, has greatly impacted the state of the art in cryptography.

It is natural to ask whether a quantum computer might allow a speedup of CFD simulations. Recently, a quantum algorithm for solving partial differential equations (PDE) has been introduced which was applied to the Navier–Stokes (NS) equations of fluid dynamics [8]. The resulting quantum algorithm was tested on a steady-state, inviscid, compressible nozzle flow problem which allows for shockwave formation and for which an exact solution is known. Excellent agreement with the exact solution was found, both when a shockwave was and was not present. The quantum NS algorithm was shown to provide a quadratic (exponential) speedup over classical random (deterministic) algorithms for non-smooth/turbulent flows.

Prior to Reference [8], Yepez used a quantum lattice-gas model to simulate the flow of a NS fluid [23–25], and to determine solutions of Burgers' equation [26]. The quantum lattice-gas model was shown to provide a significant advantage over a classical lattice-gas model. Steijl [21,22] used a hybrid quantum/classical approach to solve Poisson's equation which arises (inter alia) in incompressible NS flows. References [4,5,16,19] also examined fluid flow problems through quantum algorithms.

In this paper, we apply the quantum PDE algorithm of Reference [8] to Burgers' equation. We numerically simulate its application to flows in which a shockwave is and is not present. We find excellent agreement between our simulation results and: (i) an exact solution of a flow without a shockwave; and (ii) the results of a standard/classical CFD simulation of flows with and without a shockwave. The outline of this paper is as follows. In Sect. 2 we describe the application of the quantum PDE algorithm to Burgers' equation, while in Sect. 3 we present the results of our numerical simulation of the quantum Burgers' equation algorithm, and compare them to an exact solution and a classical CFD simulation as described above. Finally, we make closing remarks in Sect. 4.

2 Governing equations

Burgers' equation [2] (BE) is an important PDE that is often used in CFD as a simplified model for the Navier–Stokes dynamics. It contains both nonlinear and viscous terms and is widely used to test new techniques and algorithms. In one spatial dimension it is:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}. \tag{1}$$

Here $u(x,t)$ is the local flow velocity and ν is the kinematic viscosity. It proves convenient to rewrite Eq. (1) in terms of non-dimensional variables: $x^* = x/L$; $u^* = u/u_\infty$; and $t^* = tu_\infty/L$, where L is a characteristic length and u_∞ is a characteristic velocity. The resulting equation is:

$$\frac{\partial u^*}{\partial t^*} + u^*\frac{\partial u^*}{\partial x^*} = \frac{1}{Re}\frac{\partial^2 u^*}{\partial x^{*2}}, \tag{2}$$

where $Re = u_\infty L/\nu$ is the Reynolds number. In the remainder of this paper, we only work with non-dimensional variables and so will suppress the asterisk-superscript. For inviscid flow the RHS of Eq. (2) vanishes, and in conservation form it becomes:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left[\frac{u^2}{2}\right] = 0. \tag{3}$$

2.1 Quantum BE algorithm

Following the prescription of Reference [8], we begin by discretizing space: $x \to x_j$ $(1 \le j \le m)$ and $u(x,t) \to u(x_j,t) \equiv u(j,t)$. The spatial boundary points correspond to the grid-points x_1 and x_m. It is important to note that time t remains a continuous parameter. We also replace the spatial derivative by a first-order upwind scheme. This reduces the PDE, Eq.(3), to a coupled set of ordinary differential equations (ODEs):

$$\frac{du(j,t)}{dt} = -\left[\frac{u(j,t)^2 - u(j-1,t)^2}{2\Delta x}\right] \quad (2 \le j \le m-1), \tag{4}$$

where, for simplicity, the lattice spacing $\Delta x = x_j - x_{j-1}$ is assumed to be constant. Defining $f(u(j,t)) = -(u(j,t)^2 - u(j-1,t)^2)/2\Delta x$, Eq. (4) becomes:

$$\frac{du(j,t)}{dt} = f(u(j,t)) \qquad (2 \le j \le m-1). \tag{5}$$

We see that BE has been replaced by a coupled set of $m-2$ ODEs. We postpone discussion of the initial- and boundary-conditions to Sect. 3.

To solve the BE system of ODEs (Eq. (5)) we use a quantum algorithm for solving systems of nonlinear ODEs introduced by Kacewicz [14]. Specifically, we look for a bounded function $A(j,t)$ which approximates the exact solution $u(j,t)$ over the time interval $0 \le t \le T$. We require both $u(j,t)$ and $A(j,t)$ to satisfy the initial condition: $u(j,0) = A(j,0) = U_0(j)$.

The driver function $f(u)$ is assumed to have continuous, bounded derivatives to order r, with the r$^{\text{th}}$ derivative satisfying the Hölder condition:

$$\left| \left. \frac{d^r f}{du^r} \right|_{u_1} - \left. \frac{d^r f}{du^r} \right|_{u_2} \right| < H \left| u_1 - u_2 \right|^{\rho} . \tag{6}$$

Here $H > 0$ and $0 < \rho \le 1$. The driver function's smoothness is parameterized by $q = r + \rho$, with $q \gg 1$ ($q \sim 1$) corresponding to smooth (non-smooth) functions. Functions satisfying these conditions are known as Hölder class functions [7,9] and are elements of the Hölder space $\mathcal{F}^{r,\rho}$.

Kacewicz' quantum ODE algorithm begins by partitioning the time interval $[0, T]$ into n primary subintervals $T_i = [t_i, t_{i+1}]$ of duration $h = T/n$, where $t_i = ih$ ($0 \le i \le n$). To each primary subinterval T_i is associated the: (i) approximate solution $A_i(j,t)$; and (ii) parameters $\{y_i(j) |\ 0 \le i \le n-1\}$ which provide the initial condition for the primary subinterval T_i: $A_i(j,t_i) \equiv y_i(j)$. We will explain shortly how the $\{y_i(j)\}$ are assigned values. Next, each primary subinterval T_i is subdivided into $N_k = n^{k-1}$ secondary subintervals of duration $\bar{h} = h/N_k = T/n^k$ by introducing intermediate times $t_{i,m} = t_i + m\bar{h}$ ($0 \le m \le N_k$). We denote the m^{th} secondary subinterval in T_i as $T_{i,m} = [t_{i,m}, t_{i,m+1}]$, and the approximate solution within $T_{i,m}$ by $A_{i,m}(j,t)$. Taylor's method [1,11,17] is used to write $A_{i,m}(j,t)$ as a truncated Taylor series about $t_{i,m}$:

$$A_{i,m}(j,t) = A_{i,m}(j,t_{i,m}) + \sum_{v=0}^{r} \frac{1}{v!} \frac{d^v f(j,t_{i,m})}{dt^v} \left(t - t_{i,m}\right)^v + \mathcal{O}\left(\bar{h}^{r+1}\right). \tag{7}$$

For Hölder class functions $f \in \mathcal{F}^{r,\rho}$, the parameter r is given. For a quasi-smooth driver function $f(u)$, the parameter r is chosen so the error $\mathcal{O}(\bar{h}^{r+1})$ is sufficiently small. The approximate solutions $\{A_{i,m}(j,t)\}$ are required to be continuous at the intermediate times $t_{i,m}$: $A_{i,m}(j,t_{i,m+1}) = A_{i,m+1}(j,t_{i,m+1})$. As noted earlier, we required that the $\{y_i(j)\}$ provide the initial condition for the approximate solution $A_i(j,t)$ for the i$^{\text{th}}$ primary subinterval T_i. Thus, at $t = t_i \equiv t_{i,0}$, we require: $A_i(j,t_i) \equiv A_{i,0}(j,t_{i,0}) = y_i(j)$. These two requirements determine $A_i(j,t)$ throughout the subinterval T_i. Specifically, if $t \in T_{i,m}$, then $A_i(j,t) = A_{i,m}(j,t)$. Once the $\{A_i(j,t) |\ 0 \le i \le n-1\}$ are known, the global, approximate solution is known: $A(j,t) = A_i(j,t)$ for $t \in T_i$. We see that once the parameters n, k, and $\{y_i(j) |\ 0 \le i \le n-1\}$ are assigned values, the above construction determines the approximate solution $A(j,t)$. How the parameters n and k are chosen is discussed in Sect. 3. We now explain how the $\{y_i(j)\}$ are chosen.

To that end, Eq. (5) is integrated over T_i:

$$u(j,t_{i+1}) = u(j,t_i) + \sum_{m=0}^{N_k-1} \int_{t_{i,m}}^{t_{i,m+1}} d\tau\, f(A_{i,m}(j,\tau)) + \sum_{m=0}^{N_k-1} \int_{t_{i,m}}^{t_{i,m+1}} d\tau \left[f(u(j,\tau)) - f(A_{i,m}(j,\tau)) \right], \tag{8}$$

which is exact (the second term has been added and subtracted). To obtain an equation that relates the $\{y_i(j)\}$, Kacewicz replaces $u(j,t_i) \approx A_i(j,t_i) \equiv y_i(j)$ with $y_i(j)$; discards the third term on the RHS as it is $\mathcal{O}(\bar{h}^{r+1})$; and writes $\tau = \bar{h}z$ so that Eq. (8) becomes:

$$y_{i+1}(j) = y_i(j) + N_k \sum_{m=0}^{N_k-1} \frac{\bar{h}}{N_k} \int_0^1 dz\; f(A_{i,m}(j,z)), \tag{9}$$

for $0 \le i \le n-2$. Equation (9) determines $y_{i+1}(j)$ from $y_i(j)$ and the Taylor polynomials $\{A_{i,m}(j,t)\}$. The $\{y_i(j)\}$ are determined iteratively. The first step sets $y_0(j)$ equal to the initial condition: $y_0(j) = U_0(j)$. The $\{y_0(j)\}$ then determine $A_0(j,t)$ throughout the primary subinterval $T_0 = [0, t_1]$ as described above. Equation (9) then determines $y_1(j)$ from $y_0(j)$, once the integral on the RHS is evaluated. To that end, Kacewicz introduces N_k knot times $\{z_{m,p}\}$ in each secondary subinterval $T_{i,m}$ and approximates the integral by its average value over the knot times:

$$\sum_{m=0}^{N_k-1} \frac{\bar{h}}{N_k} \int_0^1 dz f(A_{i,m}(j,z)) = \frac{\bar{h}}{N_k^2} \sum_{m,p=0}^{N_k-1} f(A_{i,m}(j,z_{m,p})) \tag{10}$$

The Quantum Amplitude Estimation Algorithm [3] (QAEA) is used to estimate the average value of f appearing on the RHS of Eq. (10). Note that this is the only task in Kacewicz' quantum algorithm that requires a quantum computer. Finally, before the QAEA can be used to evaluate Eq. (10), the summand f must be shifted and rescaled so that the new summand takes values in the range [0, 1]. Novak [18] and Heinrich [10] showed how the QAEA could be used to evaluate a function average, and Reference [8] explains how the shift and rescaling is implemented. In this way the $\{y_1(j)\}$ are determined. They, in turn, determine the $\{A_{1,m}(j,t)\}$ throughout $T_1 = [t_1, t_2]$ as described above. This allows the RHS of Eq. (9) to be evaluated (using the QAEA to approximate the integral) giving the $\{y_2(j)\}$. Iterating this procedure over the remaining primary subintervals T_i gives the approximate solution $A(j,t)$, where $A(j,t) = A_i(j,t)$ for $t \in T_i$ and $0 \le i \le n-1$. Reference [14] shows that for Hölder class functions the solution error ϵ satisfies (for $n \ge 5$):

$$\epsilon \equiv sup_{\{j,t\}} |u(j,t) - A(j,t)| = \mathcal{O}\left(\frac{1}{n^{\alpha_k}}\right), \tag{11}$$

Table 1 Kacewicz bounds on the ϵ-complexity for different families of ODE algorithms applied to Hölder class functions. Here ϵ is the error tolerance on the ODE solution; $q = r + \rho$ is the driver function smoothness parameter introduced in Sect. 2.1, with r the highest order derivative appearing in the truncated Taylor series (see Eq. (7)), and $0 \leq \rho \leq 1$; and $0 < \gamma \leq 1$

Bounds	Quantum	Classical random	Classical deterministic
Upper	$\mathcal{O}\left[(1/\epsilon)^{1/(q+1-\gamma)}\right]$	$\mathcal{O}\left[(1/\epsilon)^{1/(q+(1/2)-\gamma)}\right]$	$\mathcal{O}\left[(1/\epsilon)^{1/q}\right]$
Lower	$\Omega\left[(1/\epsilon)^{1/(q+1)}\right]$	$\Omega\left[(1/\epsilon)^{1/(q+(1/2))}\right]$	$\Omega\left[(1/\epsilon)^{1/q}\right]$
	(Almost) Optimal	(Almost) Optimal	Optimal

with probability $1 - \delta$. Here $\alpha_k = k(q + 1) - 1$ and $q = r + \rho$ is the driver function smoothness parameter. To reach this level of performance Kacewicz requires the upper bound ϵ_1 on the QAEA estimate of the function average and the probability $1 - \delta_1$ that this bound is satisfied [14] be given by $\epsilon_1 = 1/n^{k-1}$ and $1 - \delta_1 = (1 - \delta)^{1/n^k}$. In Sect. 3 we discuss how ϵ_1 and δ are assigned values.

2.2 Complexity analysis

Both quantum and classical algorithms for BE must discretize the spatial continuum, and in the interests of an apples-to-apples comparison, we assume both algorithms use the same discretization procedure. This guarantees the discretization costs for both are the same, and so comparison of the complexity of the quantum and classical BE algorithms reduces to the relative complexity of their respective ODE solvers.

Numerical solution of an ODE requires multiple evaluations of its driver function $f(u)$. We assume that an oracle is available. For a classical algorithm, the oracle is a black-box function/subroutine that returns $f(u)$, while for a quantum algorithm, the oracle is an operator whose action encodes f(u) in the quantum state. The quantum oracle used in the QAEA is described in [8,18].

The computational cost of an algorithm A over a family of driver functions $\mathcal{F}$ is the maximum number of oracle calls needed by A to compute an approximate solution $A(j, t)$ over all driver functions $f \in \mathcal{F}$. It is thus the number of oracle calls needed to solve the hardest driver function in $\mathcal{F}$. For a given $\epsilon > 0$, the ϵ-complexity $comp^t(\mathcal{F}, \epsilon)$ for an algorithm of type $t = \{\text{deterministic, random, quantum}\}$ is the minimum (computational) cost over all algorithms A of type t whose error satisfies $e^t(A, \mathcal{F}) < \epsilon$. It is thus the cost of the best algorithm of type t on driver functions $f \in \mathcal{F}$.

Over a twenty year period, Kacewicz [12–15] determined upper and lower bounds on the ϵ-complexity for quantum, classical random, and classical deterministic ODE algorithms applied to Hölder class driver functions. We reproduce his results in Table 1.

The bounds depend on the error tolerance ϵ, and on the parameters (introduced in Sect. 2.1): (i) $0 < \gamma \leq 1$; (ii) $0 < \rho \leq 1$; and (iii) the smoothness parameter $q = r + \rho$, where r is the highest order derivative kept in the truncated Taylor series (see Eq. (7)). The upper bounds for the quantum and classical random ODE algorithms are for the algorithms introduced in [14]. The quoted lower bounds for quantum and

classical random algorithms apply to all respective algorithms in these two classes, as shown in [12,13]. The algorithms in [14] are thus (almost) optimal as the exponents in the respective upper and lower bounds agree up to a small parameter γ, and in the quantum case, to within a logarithmic factor which we have suppressed. Kacewicz' classical deterministic ODE algorithm is seen to be optimal.

The ϵ-complexity is seen to be exponentially sensitive to the smoothness parameter $q = r + \rho$. For smooth driver functions, r is large (derivatives exist to high order) and so $q \gg 1$. For non-smooth driver functions, $r, q = \mathcal{O}(1)$. In the most extreme case of a continuous, but non-differentiable function, $r = 0$ and so $q = \rho \leq 1$.

For smooth functions ($q \gg 1$), Table 1 gives

$$comp^{quant}(A, \mathcal{F}, \epsilon) \sim comp^{ran}(A, \mathcal{F}, \epsilon) \sim comp^{det}(A, \mathcal{F}, \epsilon) \sim \left(\frac{1}{\epsilon}\right)^{\frac{1}{q}}. \tag{12}$$

Thus, for smooth driver functions, the complexity of all three types of ODE algorithm are the same slowly varying function of ϵ, and so there is no quantum speed-up in this case. However, for non-smooth driver functions, there is a quantum speed-up. This is apparent from the upper bounds in Table 1. We see that the exponent in the quantum upper bound is smaller than the corresponding exponent for the classical random and classical deterministic algorithms. Its complexity is thus smaller, corresponding to less oracle calls, and so to a shorter runtime. Thus for non-smooth driver functions there is a quantum speed-up. The degree of speed-up increases with decreasing q, being largest for $q, \gamma \ll 1$. In this limit,

$$\begin{aligned} comp^{quant}(A, \mathcal{F}, \epsilon) &\sim \left(\frac{1}{\epsilon}\right); \\ comp^{ran}(A, \mathcal{F}, \epsilon) &\sim \left(\frac{1}{\epsilon}\right)^2; \\ comp^{det}(A, \mathcal{F}, \epsilon) &\sim \left(\frac{1}{\epsilon}\right)^{1/q}. \end{aligned} \tag{13}$$

The quantum algorithm thus has a square-root reduction in its ϵ-complexity (oracle calls) over classical random algorithms, and so a quadratic speed-up in runtime. The quantum speed-up is more pronounced when compared to classical deterministic algorithms where the speed-up is exponential in $1/q \gg 1$.

To summarize: the quantum ODE algorithm provides a substantial quantum speed-up over classical random and classical deterministic ODE algorithms for a non-smooth driver function, which is the computationally most challenging case. The largest speed-up occurs in the regime $q, \gamma \ll 1$. As noted at the beginning of this subsection, the same conclusion applies to the quantum BE algorithm.

3 Results

In this section, we present the results of a simulation of the quantum BE algorithm applied to two BE flows. In Sect. 3.1 we examine a smooth flow problem for which an exact analytical solution can be found, while in Sect. 3.2 we examine a flow which contains a travelling shockwave discontinuity. We also carried out a classical CFD simulation of the BE dynamics whose results we compare with the results of our quantum BE simulation.

Before entering into a presentation of our results, we specify the input parameters for our simulation. These parameter values were used in both simulations. We consider flows along the x-axis with $0 \leq x \leq 3$. Recall that we are working with non-dimensional flow velocities, positions, and times. We discretize space by introducing a lattice with $m = 61$ grid-points. We chose the error bound $\epsilon_1 = 0.005$, and the probability $\delta = 0.005$ which is the probability that the quantum BE algorithm returns a solution that violates Eq. (11). We noted in Sect. 2.1 that Kacewicz requires $\epsilon_1 = 1/n^k$. Solving this expression for k gives:

$$k = 1 + \lceil \ln(1/\epsilon_1) / \ln(n) \rceil, \tag{14}$$

where $\lceil z \rceil$ denotes the smallest integer greater than z. To obtain a second relation between n and k, we require that the duration $\bar{h}$ of a secondary subinterval be consistent with the Courant–Friedrichs–Lewy (CFL) stability condition. [6] This condition introduces a local time increment $\Delta t(j)$ at each grid-point j,

$$\Delta t(j) = C \frac{\Delta x}{u(j,t)} \quad (2 \leq j \leq m-1), \tag{15}$$

with $C < 1$. We chose $C = 0.1$, and the CFL time-increment is then $\Delta t_{CFL} = min_{\{2 \leq j \leq m-1\}} \Delta t(j)$. The duration time of the simulation is then $T = N_t \Delta t_{CFL}$, where $N_t = 1000$ was chosen. As discussed in Sect. 2.1, $T = \bar{h} n^k$. Setting these two expressions for T equal to each other (with minor rearrangement) gives:

$$\frac{\bar{h}}{\Delta t_{CFL}} = \frac{N_t}{n^k}. \tag{16}$$

To insure our time partition is consistent with the CFL stability condition we require that n and k be chosen so that $\bar{h}/\Delta t_{CFL} < 1$. This requires

$$\frac{N_t}{n^k} < 1. \tag{17}$$

Equations (14) and (17) allow n and k to be determined iteratively. Begin by choosing a value n_0 for n. Then use Eq. (14) to determine k_0. Next evaluate $N_t/n_0^{k_0}$ and test whether it is less than 1. If yes, then set $n = n_0$ and $k = k_0$ and stop. Otherwise, set $n_1 = n_0 + 1$ and evaluate k_1 using Eq. (14). Then, test whether $N_t/n_1^{k_1} < 1$. If yes, stop with $n = n_1$ and $k = k_1$. Otherwise, continue to iterate in this fashion until first

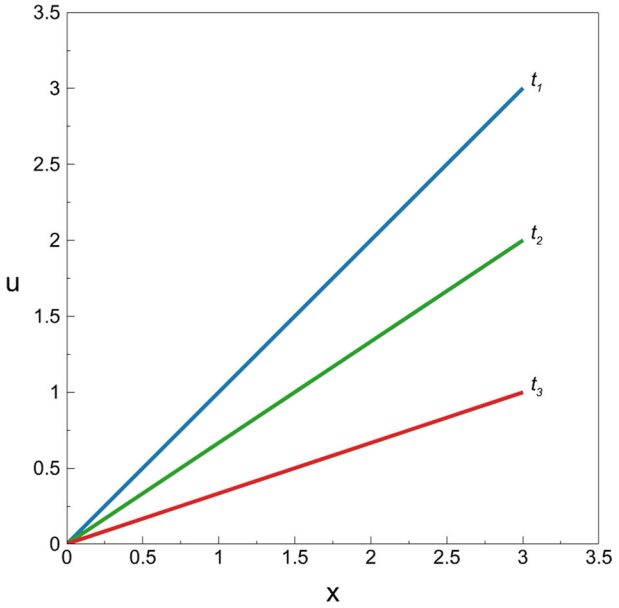

Fig. 1 An exact analytical solution of Burgers' equation. We plot the exact analytical solution of Burgers' equation for the flow problem discussed in Sect. 3.1 at three times: $0 \leq t_1 < t_2 < t_3 \leq T$. As explained in Sect. 2, the flow velocity u and position x are dimensionless

finding n_i and k_i such that $N_t/n_i^{k_i} < 1$. When this occurs, set $n = n_i$ and $k = k_i$ and stop. In this manner we arrived at $n = 18$ and $k = 3$.

3.1 Smooth flow

Here we look for a solution of BE for a smooth inviscid flow. The boundary condition at $x_1 = 0$ is $u(x_1, t) = 0$. At $x_{61} = 3$, because the flow is out of the simulation region, the boundary condition is a floating boundary condition. The flow velocity $u(x_{61}, t)$ is found by linear extrapolation from the flow velocity at x_{59} and x_{60}: $u(x_{61}, t) = 2u(x_{60}, t) - u(x_{59}, t)$. Finally, we impose the initial condition $u(x, 0) = x$, which satisfies both boundary conditions.

An exact solution for this flow problem can be found using separation of variables: $u(x, t) = X(x)T(t)$. Inserting this ansatz into BE, carrying out the separation, and imposing the initial condition gives $X(x) = x$ and $T(t) = 1/(1 + t)$. The exact analytic solution is then:

$$u_a(x, t) = \frac{x}{1 + t}. \tag{18}$$

We plot the exact solution in Fig. 1

at three different times: $0 \leq t_1 < t_2 < t_3 \leq T$. The flow is seen to go asymptotically to zero everywhere with increasing time.

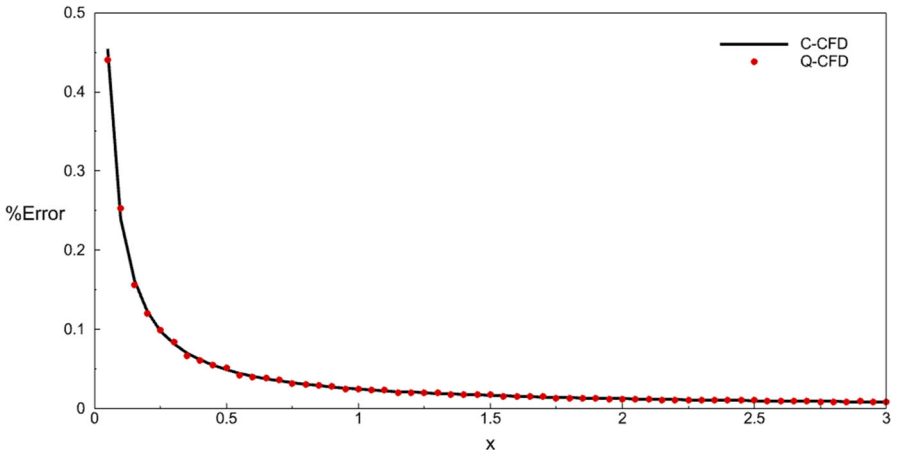

Fig. 2 Percent-error of quantum and classical BE solvers relative to the exact analytical solution. We plot the percent-error of our quantum CFD solver (Q-CFD) and a classical CFD solver (C-CFD) for BE versus spatial position at the end of the simulation ($T = 1.667$). As explained in Sect. 2, the position x is dimensionless

To examine the performance of the quantum BE algorithm we wrote a quantum CFD solver (Q-CFD) to simulate the application of the quantum BE algorithm to this problem. The solver's output $u_q(j,t)$ was compared with the exact analytical solution $u_a(j,t)$. To assess the accuracy of the quantum BE algorithm solution we calculated the percent-error:

$$\%\text{Error} = \frac{|u_a(j) - u_q(j)|}{u_a(j)} \tag{19}$$

at the end of the simulation: $T = 1.667$. We plot the percent-error in Fig. 2.

We see that the quantum BE algorithm result is in excellent agreement with the exact solution. The largest percent-error occurs near $x = 0$ which is to be expected because of our use of an upwind scheme. The percent-error in that region is of the order of a few tenths of a percent. For $x > 0.5$, the percent-error drops to a few hundredths of a percent. Figure 2 also plots the percent-error for a classical CFD solver (C-CFD) applied to this problem. We see that that the quantum and classical solvers produce comparable errors.

3.2 Travelling shockwave

Here we examine the solution of BE when the initial condition contains a shockwave discontinuity:

$$u(x,0) = \begin{cases} 1 & (0 \le x \le 1.5) \\ 0 & (1.5 < x \le 3.0). \end{cases} \tag{20}$$

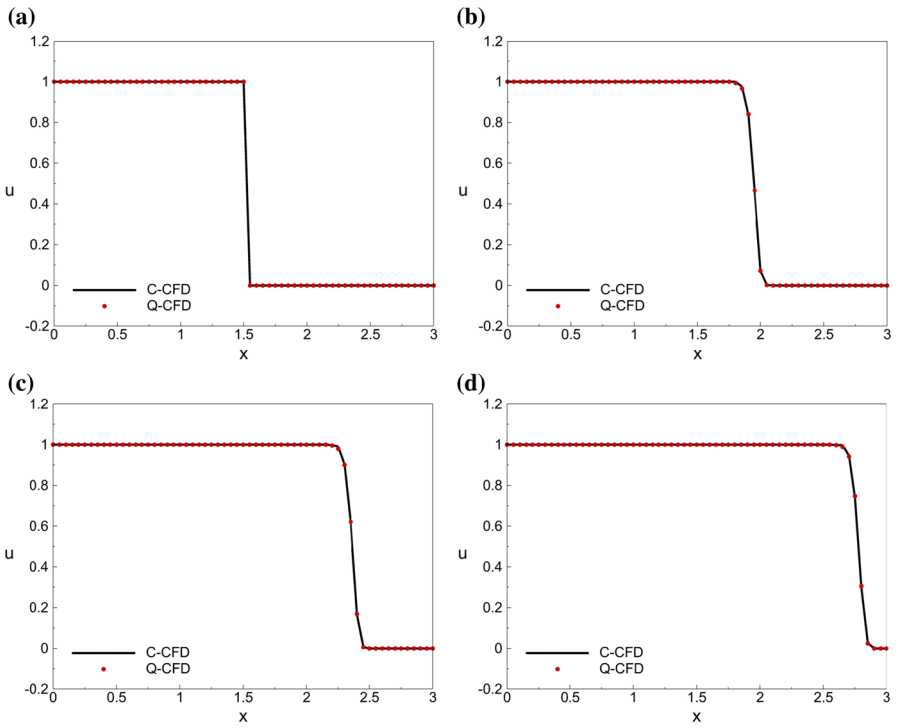

Fig. 3 Comparing quantum and classical BE solvers for a travelling shockwave flow. We plot the results of our quantum BE solver (Q-CFD) and a classical BE solver (C-CFD) at four times: **a** $t = 0.00$; **b** $t = 0.83$; **c** $t = 1.67$; and **d** $t = 2.50$. We see that the results of both solvers are in excellent agreement. As explained in Sect. 2, the flow velocity u and the time t are dimensionless

The boundary conditions are $u(0, t) = 1$ and $u(3, t) = 0$. These conditions require the initial shockwave to travel in the direction of increasing x. We ran the simulation until the shockwave discontinuity arrived at $x = 3.0$. This insures that the boundary condition at $x = 3.0$ is obeyed throughout the simulation. For this flow problem, no exact analytical solution is known. Thus we compared the results of our Q-CFD solver to that of a C-CFD solver. We plot the results for both solvers in Fig. 3

for $t = 0, 0.83, 1.67$, and 2.5. We see that: (i) the quantum BE algorithm successfully converged even though the flow contained a travelling shockwave discontinuity; and (ii) the Q-CFD results are in excellent agreement with that of the C-CFD solver. Finally, notice that both solvers slightly rounded the kinks in the flow solution by comparable amounts.

4 Conclusion

In this paper we have applied the quantum PDE algorithm of Reference [8] to BE in one-spatial dimension. We numerically simulated the quantum algorithm's application to inviscid flows with and without a shockwave. The results found were compared to

an exact solution which is known when no shockwave is present in the flow, and to a standard CFD simulation when a shockwave is and is not present. Agreement was excellent, and the quantum algorithm's error was seen to be of the same order as that of the classical CFD simulation. We underscore that the quantum algorithm successfully converged to the solution even when a shockwave was present. We also discussed the computational complexity and speedup of the quantum BE algorithm which was found to be identical to that of the quantum PDE [8] and ODE [14] algorithms. It was shown that there is a quadratic (exponential) speedup over classical random (deterministic) algorithms for non-smooth driver functions which is the computationally most challenging case. It is important to note that this speedup is only expected to occur when the quantum algorithm is run on a quantum computer. Numerical simulation of the quantum algorithm on a classical computer is not expected to show a quantum speedup as classical computers do not generate the quantum entanglement or state superposition that underlie the speedup.

To close, we list a few useful directions for future work:

- Application of the quantum PDE, NS, and BE algorithms to problems in two or more spatial dimensions;
- Determining the quantum circuit implementation of the quantum PDE algorithm; and
- Applying the quantum NS and BE algorithms to more complex flows.

Acknowledgements F. Gaitan thanks T. Howell III for continued support.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Atkinson, K.: Elementary Numerical Analysis. Wiley, New York (1985)
2. Bateman, H.: Some recent researches on the motion of fluids. Mon. Weather Rev. **43**(4), 163–170 (1915)
3. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemp. Math. **305**, 53–74 (2002)
4. Cao, Y., Papageorgiou, A., Petras, I., Traub, J., Kais, S.: Quantum algorithm and circuit design solving the Poisson equation. New J. Phys. **15**(1), 013021 (2013)
5. Chen, Z.Y., Xue, C., Chen, S.M., Lu, B.H., Wu, Y.C., Ding, J.C., Huang, S.H., Guo, G.P.: Quantum finite volume method for computational fluid dynamics with classical input and output. arXiv preprint arXiv:2102.03557 (2021)

6. Courant, R., Friedrichs, K., Lewy, H.: Über die partiellen differenzengleichungen der mathematischen physik. Math. Ann. **100**(1), 32–74 (1928)
7. Evans, L.C.: Partial Differential Equations. American Mathematical Society, Providence (1998)
8. Gaitan, F.: Finding flows of a Navier–Stokes fluid through quantum computing. NPJ Quantum Inf. **6**(1), 1–6 (2020)
9. Gilbarg, D., Trudinger, N.: Elliptic Partial Differential Equations of Second Order. Springer, Berlin (1983)
10. Heinrich, S.: Quantum summation with an application to integration. J. Complex. **18**(1), 1–50 (2002)
11. Iserles, A.: A First Course in the Numerical Analysis of Differential Equations. Cambridge University Press, Cambridge (2009)
12. Kacewicz, B.: Randomized and quantum algorithms yield a speed-up for initial-value problems. J. Complex. **20**(6), 821–834 (2004)
13. Kacewicz, B.: Improved bounds on the randomized and quantum complexity of initial-value problems. J. Complex. **21**(5), 740–756 (2005)
14. Kacewicz, B.: Almost optimal solution of initial-value problems by randomized and quantum algorithms. J. Complex. **22**(5), 676–690 (2006)
15. Kacewicz, B.Z.: Optimal solution of ordinary differential equations. J. Complex. **3**(4), 451–465 (1987)
16. Mezzacapo, A., Sanz, M., Lamata, L., Egusquiza, I., Succi, S., Solano, E.: Quantum simulator for transport phenomena in fluid flows. Sci. Rep. **5**(1), 1–7 (2015)
17. Moursund, D.G., Duris, C.S.: Elementary Theory and Application of Numerical Analysis. Dover, New York (1988)
18. Novak, E.: Quantum complexity of integration. J. Complex. **17**(1), 2–16 (2001)
19. Ray, N., Banerjee, T., Nadiga, B., Karra, S.: Towards solving the Navier–Stokes equation on quantum computers. arXiv preprint arXiv:1904.09033 (2019)
20. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
21. Steijl, R.: Quantum algorithms for fluid simulations. In: Advances in Quantum Communication and Information, p. 31. IntechOpen (2019)
22. Steijl, R., Barakos, G.N.: Parallel evaluation of quantum algorithms for computational fluid dynamics. Comput. Fluids **173**, 22–28 (2018)
23. Yepez, J.: Lattice-gas quantum computation. Int. J. Mod. Phys. C **9**(08), 1587–1596 (1998)
24. Yepez, J.: Quantum computation of fluid dynamics. In: NASA International Conference on Quantum Computing and Quantum Communications, pp. 34–60. Springer (1998)
25. Yepez, J.: Quantum lattice-gas model for computational fluid dynamics. Phys. Rev. E **63**(4), 046702 (2001)
26. Yepez, J.: Quantum lattice-gas model for the burgers equation. J. Stat. Phys. **107**(1), 203–224 (2002)

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com