RESEARCH ARTICLE | APPLIED PHYSICAL SCIENCES

Hybrid quantum algorithms for flow problems

Sachin S. Bharadwaj[a] and Katepalli R. Sreenivasan[a,b,c,d,1]

Contributed by Katepalli R. Sreenivasan; received June 29, 2023; accepted October 17, 2023; reviewed by Luca Moriconi, Aditya Konduri, and Susan Kurien

For quantum computing (QC) to emerge as a practically indispensable computational tool, there is a need for quantum protocols with end-to-end practical applications—in this instance, fluid dynamics. We debut here a high-performance quantum simulator which we term QFlowS (Quantum Flow Simulator), designed for fluid flow simulations using QC. Solving nonlinear flows by QC generally proceeds by solving an equivalent infinite dimensional linear system as a result of linear embedding. Thus, we first choose to simulate two well-known flows using QFlowS and demonstrate a previously unseen, full gate-level implementation of a hybrid and high precision Quantum Linear Systems Algorithms (QLSA) for simulating such flows at low Reynolds numbers. The utility of this simulator is demonstrated by extracting error estimates and power law scaling that relates T_0 (a parameter crucial to Hamiltonian simulations) to the condition number κ of the simulation matrix and allows the prediction of an optimal scaling parameter for accurate eigenvalue estimation. Further, we include two speedup preserving algorithms for a) the functional form or sparse quantum state preparation and b) in situ quantum postprocessing tool for computing nonlinear functions of the velocity field. We choose the viscous dissipation rate as an example, for which the end-to-end complexity is shown to be $\mathcal{O}(\text{polylog}(N/\epsilon)\kappa/\epsilon_{QPP})$, where N is the size of the linear system of equations, ϵ is the solution error, and ϵ_{QPP} is the error in postprocessing. This work suggests a path toward quantum simulation of fluid flows and highlights the special considerations needed at the gate-level implementation of QC.

quantum computation | computational fluid mechanics (CFD) | quantum linear systems algorithms (QLSA) | fluid dynamics

Significance

Quantum computing (QC) has advantages of speed and storage over classical computing, but it is based on a linear paradigm. However, many problems of interest are nonlinear. A viable QC algorithm includes a suitable preparation of a nonlinear problem into a linearized setting, doing computations quantum mechanically and reading out the results in classical terms. This end-to-end process usually diminishes the quantum advantage. We solve Poiseuille and Couette flows by introducing an in-house quantum simulator, named Quantum Flow Simulator, for computational fluid dynamics, and highlight the limitations of QC while preserving the quantum advantage. The quantum algorithmic as well as the classical software machinery developed here sets the stage for future quantum simulations.

Computer simulations of nonlinear physical systems—such as turbulent flows, glassy systems, climate physics, molecular dynamics, and protein folding—are formidably hard to perform on even the most powerful supercomputers of today or of foreseeable future. In particular, the state-of-the-art Direct Numerical Simulations (DNS) of turbulent flows (1–3) governed by the Navier–Stokes equations, or of turbulent reacting flow problems and combustion (4), both of which involve massive simulations with high grid resolutions, not only reveal fine details of the flow physics (5, 6), but also constantly contend with the limits of supercomputers on which the codes run (7). However, simulation sizes required to settle fundamental asymptotic theories, or simulate turbulent systems such as the Sun or cyclones or to simulate flows around complex geometries of practical interest, would require computing power that is several orders of magnitude higher than is currently available. Reaching such computational targets calls for a paradigm shift in computing technology.

One such potential candidate is quantum computing (QC) (8), which has striven to establish its advantage over classical counterparts by promising polynomial or exponential speedups (9). Even though QC has been around for the last two decades, the subject is still nascent. In this nascent era, which has been called the Noisy Intermediate Scale Quantum (NISQ) era, QC's applications already extend (10, 11) across finance, chemistry, biology, communication, and cryptography, but not as much in areas that are governed by nonlinear partial differential equations, such as fluid dynamics.

This work attempts to pave the way for utilizing QC in computational fluid dynamics (CFD) research, which we have termed (12) quantum computation of fluid dynamics (QCFD). An initial comprehensive survey of various possible directions of QCFD was made in ref. 12. Realistic CFD simulations with quantum advantages require one to quantumly solve general nonlinear PDEs such as the Navier–Stokes equations. However, it is worth noting that the fundamental linearity of quantum mechanics itself blockades encoding of nonlinear terms, thus forcing a linearization of some kind (13–15), which typically results in an infinite dimensional linear system. In such cases, the inaccessibility to the required large number of qubits (and thus exponentially large vector spaces) leads to inevitable truncation errors, limiting the focus to weakly nonlinear problems (13).

Author affiliations: [a]Department of Mechanical and Aerospace Engineering, New York University, New York, NY 11201; [b]Courant Institute of Mathematical Sciences, New York University, New York, NY 10012; [c]Department of Physics, New York University, New York, NY 10012; and [d]Center for Space Science, New York University, Abu Dhabi 129188, United Arab Emirates

Author contributions: S.S.B. and K.R.S. designed research; S.S.B. performed research; S.S.B. and K.R.S. analyzed data; and S.S.B. and K.R.S. wrote the paper.

Reviewers: L.M., Universidade Federal do Rio de Janeiro; A.K., Indian Institute of Science, Bangalore; and S.K., Los Alamos National Laboratory.

The authors declare no competing interest.

[1]To whom correspondence may be addressed. Email: katepalli.sreenivasan@nyu.edu.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2311014120/-/DCSupplemental.

Published December 1, 2023.

Therefore, the ability to solve high dimensional linear systems in an end-to-end manner* while capturing the flow physics is crucial to simulating nonlinear flow problems. Our goal here is to present various steps involved in the process of solving simple and idealized problems, including providing estimates of scaling and errors involved.

To achieve this goal, we present here a high-performance quantum simulator which we call QFlowS (Quantum Flow Simulator), designed specifically to simulate fluid flows. Built on a C++ platform, it offers both QC and CFD tools in one place. With QFlowS we implement a modified version of the class of algorithms, now termed quantum linear systems algorithms (QLSA). Under some caveats, these algorithms promise to solve a linear system of equations given by the matrix inversion problem $A\mathbf{x} = \mathbf{b}$, with up to an exponential speedup compared to known classical algorithms. In recent years, a number of efforts based on continuum methods using QLSA (16, 17), variational quantum algorithms (18, 19), amplitude estimation methods (20, 21), lattice based methods (22–24), and quantum-inspired methods (25) have been undertaken to solve linear and nonlinear PDEs. However, most of these efforts have been theoretical, lacking gate-level quantum numerical simulations and analysis of the resulting flow field or proper estimates of the actual errors involved.

In particular, a full-gate-level quantum simulation is implemented on QFlowS to solve the unsteady Poiseuille and Couette flow problems. We implement both the fundamental form of QLSA, called the Harrow–Hassidim–Lloyd (HHL) algorithm (26) and its more recent counterpart (27), based on the linear combination of unitaries (LCU). In addition, we prescribe suitable quantum state preparation protocols and propose a hiterto unseen quantum postprocessing (QPP) protocol to compute in situ nonlinear functions of the resulting flow solution. In particular, we obtain the viscous dissipation rate $\varepsilon = \nu(\frac{\partial u}{\partial y})^2$ averaged over the flow field u, ν being the viscosity. Together, this forms an end-to-end implementation, which alleviates, to some extent, the restrictions posed by both quantum state preparation and the measurement of qubits—which are otherwise the major limiters of the theoretical quantum advantage (11, 16, 26, 28). Although the proposed algorithms are far from realistic fluid simulations, they make quantum implementations more amenable for small systems and inform the running of codes on near-term NISQ machines while attempting to preserve the quantum advantage.

By necessity, the paper uses a number of acronyms. For convenient reference, a glossary is provided in *SI Appendix*, section 5.

Linear Flow Problems

We consider the well-known 1D unsteady Poiseuille and Couette flows (schematic shown in *SI Appendix*, Fig. S1*A* that are linear dissipative flows which describe, for instance, microchannel flows (e.g., in microchips, blood capillaries, and syringes) or lubricant flows around bearings. The framework outlined in this work is readily extendable to the linear advection-diffusion with constant advection velocity. More generally, this algorithm caters to the class of elliptic and parabolic PDEs described by d-dimensional Laplace, Poisson, and heat equations. Under certain boundary conditions, the flows under discussion admit exact analytical solutions, thus making them ideal candidates for evaluating the performance of the quantum solver. Some earlier works such as refs. 29 and 30 made some important observations in possible implementations on QC for similar problems and estimated theoretical upper bounds of their complexities. The general form of the governing PDEs considered here (assuming no body forces or source terms) is given by the momentum and mass conservation relations:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{C}\cdot\nabla\mathbf{u} = \frac{1}{Re}\nabla^2\mathbf{u} - \nabla\mathbf{p}, \qquad [1]$$

$$\nabla\cdot\mathbf{u} = 0, \qquad [2]$$

where $\mathbf{u} = (u, v, w)$ is the velocity field, $\mathbf{C}$ is a constant advection velocity (which is zero in the fully developed state of flows considered here), p is pressure field, $Re = UD/\nu$ is the Reynolds number, U is the characteristic velocity, ν is the kinematic viscosity, and D is the separation between the boundaries. The so-called fully developed 1D case (applicable for all discussions from here on) reduces to:

$$\frac{\partial u}{\partial t} = \frac{1}{Re}\frac{\partial^2 u}{\partial y^2} - \frac{\partial p}{\partial x}, \qquad [3]$$

where the velocity varies only along y (wall-normal direction), and the pressure gradient $\frac{\partial p}{\partial x}$ is a constant. The boundary conditions are no-slip with $u(0,t) = u(D,t) = 0$ for the Poiseuille flow and $u(0,t) = 0$ and $u(D,t) = 1$ for the Couette flow. The initial condition for the temporal evolution is set to be a uniform flow $u(y,0) = u_{in} = 1$.

We reiterate that this problem is simple from the standpoint of the sophisticated advances of classical CFD. Nevertheless, this is an excellent starting point for demonstrating the viability of quantum algorithms for CFD—this being the spirit of the present work.

Hybrid Quantum-classical Numerical Setup. The goal now is to solve Eq. **3** by means of QLSA, which thus necessitates Eq. **3** to be recast as a linear system of equations. To do this, we consider the method of finite differences to discretize the computational domain in both space and time. Details of these schemes, their stability considerations, and the resulting matrix equations that form the input to the quantum algorithm are outlined in *SI Appendix*, section 1A. The well-known second-order central difference scheme is used to discretize the Laplacian operator for N_g grid points, while both forward and backward Euler (FE and BE from here on) schemes are implemented to discretize time. This procedure yields a set of three possible matrix equations $\mathbf{A}_{be1}\bar{u} = \mathbf{b}_{be1}$, $\mathbf{A}_{be2}\bar{u} = \mathbf{b}_{be2}$ and $\mathbf{A}_{fe}\bar{u} = \mathbf{b}_{fe}$, where be1 corresponds to the iterative backward-Euler scheme (system solved for every time step iteratively), and be2 and FE correspond to the one-shot backward Euler and forward Euler schemes (system solved for all time steps in one shot), respectively.

To solve these equations, a hybrid quantum-classical method is developed (schematic flow chart is shown in Fig. 1*B*). The preconditioning and computations of the elements of the matrices $\mathbf{A}_{be1}, \mathbf{A}_{be2}, \mathbf{A}_{fe}$ and vectors $\mathbf{b}_{be1}, \mathbf{b}_{be2}, \mathbf{b}_{fe}$ are done classically. Certain parameters required (as elucidated later) for quantum state preparation (e.g., rotation angles and decision trees) and for Hamiltonian simulation (time T_0^*) (collectively termed β) are precomputed classically as well. N from here on refers to the dimension of final matrix system that results from these considerations. With this on hand, the inputs are first loaded on the QC by the quantum state preparation algorithms (QSP-1,-2) and the resulting linear system of equations is then solved by QLSA. In the case of iterative BE, $\mathbf{A}_{be1}\bar{u} = \mathbf{b}_{be1}$

*By this, we mean an algorithm that efficiently prepares a quantum state, processes it, and outputs a result by measurement while retaining all or some net quantum advantage.

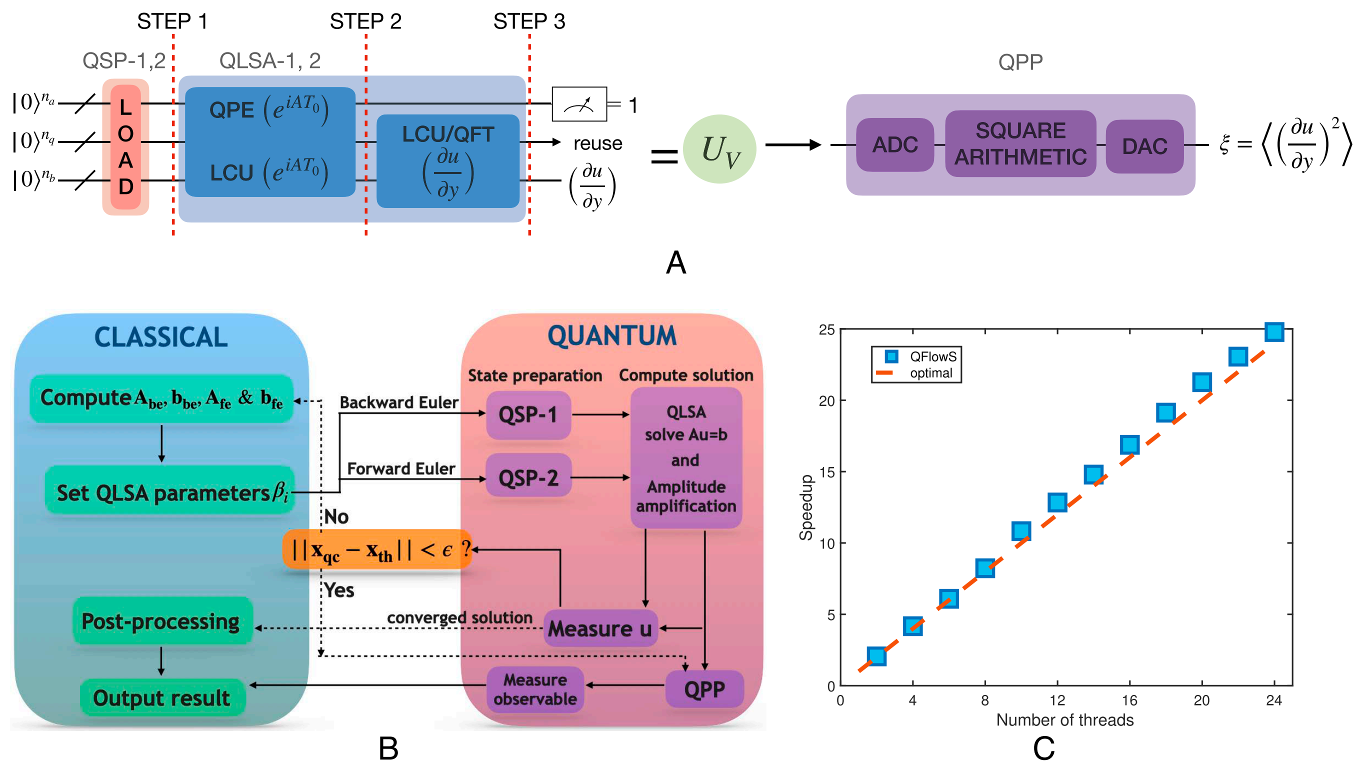

Fig. 1. Panel (*A*) shows the modified QLSA circuit with QSP, forming an oracle U_V that is employed recursively in the QPP protocol for computing viscous dissipation rate by a combination of quantum analog–digital converters (QADCs). (*B*) The working flowchart of the hybrid quantum-classical algorithm. (*C*) The strong scaling performance of QFlowS with superlinear speedup when run on single node, parallelized with OpenMP up to a total of 24 threads on NYU's Greene supercomputer. The performance is for simulations of a 20 qubit circuit of depth 422, performing a quantum Fourier transform (QFT) and inverse QFT algorithm. The standard deviations are smaller than the blue square symbols around the mean, computed over an ensemble of 8,000 simulations with random initial quantum states.

is solved for velocities $\bar{u}$, at every time step until convergence (residue reaching a tolerance $\leq \epsilon_{tol} = 10^{-6}$), which is checked classically. In a contrasting setup, BE and FE are used to set up, respectively, $\mathbf{A}_{be2}\tilde{u} = \mathbf{b}_{be2}$ and $\mathbf{A}_{fe}\tilde{u} = \mathbf{b}_{fe}$, giving $\tilde{u} = [u(y,0), u(y,dt), \cdots, u(y,T)]$, in one shot, $\forall t \in [0,T]$. It is important to note that even the BE method can be set up such that the solution is computed for all t at once. However, in the absence of efficient state preparation and measurement protocols, measuring the solution and repreparing the state for the next time step are $\mathcal{O}(N_g)$ operations that eliminate any quantum advantage, making the overall algorithm no better than classical solvers (and with additional errors due to quantum measurements). In any case, it is still worthwhile establishing how the method fares as a plausible alternative to classical simulations.

The end solution is either: a) simply read by quantum measurements for postprocessing on a classical device, or b) postprocess, in situ, a quantum device using the QPP protocol introduced here. The former, at the level of a simulator, allows one to validate the correctness of solutions and redesign the circuit as required. In the latter case, only a single target qubit and few ancillas are measured, outputting one observable—which is a real-valued nonlinear function of the velocity field. Apart from computing nonlinear functions, this circumvents expensive and noisy measurements of entire quantum states and more importantly preserves quantum advantage (to the extent possible).

QFlowS

In refs. 12 and 31, several commercially available quantum simulation packages are listed. Most of them, for instance, Qiskit (IBM), Quipper (32) and QuEST (33), are constructed for general purpose quantum simulations and are highly optimized for such operations, making it hard to customize the fundamental subroutines and data structures for CFD calculations. On the other hand, there are software applications such as ANSYS and OpenFOAM that perform solely classical CFD simulations. With the motivation of having a single bespoke quantum simulator for CFD, we unveil here a high-performance, gate-level quantum-simulation toolkit, which we call QFlowS; it is based on a C++ core and designed to be used both independently or as part of other software packages. It has a current capability of 30+ qubit simulation of custom quantum circuits. It also has several built-in gates and quantum circuits that could be used readily, while also being able to probe different quantum state metrics (such as the norm, density matrix, and entanglement). Along with these, it includes basic CFD tools needed to set up flow problems, making it versatile for QCFD simulations. Noise modelling is in progress and forms the major part of future software development. QFlowS is also being increasingly parallelized for optimal performance on supercomputers. For instance, Fig. 1*C* shows the strong scaling performance using OpenMP. The performance is measured while running on NYU's Greene supercomputing facility. On a single medium-memory computer node (48 cores: 2x Intel Xeon Platinum 8268 24C 205W 2.9GHz Processor) and for a choice of 20 qubits, we measure the run-time (by omitting the one-time initial overhead processes) of a QFT-IQFT circuit action on an ensemble of randomly initialized quantum states. We observe near-optimal and at times superoptimal scaling with increasing number of threads up to 24. Superoptimality arises when the quantum circuit is sparse, causing lesser quantum entanglement. Every single circuit layer operation is distributed over many worker threads, whose cache size exceeds the size of quantum state subspace being handled, thus making them closely parallel. *SI Appendix*, section 2, summarizes features of QFlowS.

QLSA

One of the first quantum protocols for solving equations of the form $A\vec{x} = \vec{b}$ is the HHL algorithm (26), which is the basis for what we refer to here as QLSA-1. In ref. 26, it was shown that For a Hermitian and nonsingular matrix $A \in \mathbb{C}^{2^n \times 2^n}$, vector $b \in \mathbb{C}^{2^n}$ $(N = 2^n)$, given oracles to prepare A and b in $\mathcal{O}(polylog(N))$, and a prescribed precision of $\epsilon > 0$, there exists an algorithm that computes a solution x such that $|||x\rangle - |A^{-1}b\rangle|| \le \epsilon$ in $\mathcal{O}(polylog(N)s^2\kappa^2/\epsilon)$, where κ is the condition number of the matrix and s is the sparsity. This shows that the algorithm is exponentially faster than classical alternatives, but there are important caveats (28)). Later works (34, 35) attempted to address these caveats, while some others (36–38) fundamentally improved the method by reducing error complexity from $poly(1/\epsilon)$ to $poly(\log(1/\epsilon))$. Consequently refs. 16, 37, and 39 led to a more precise class of QLSA methods based on the LCU (27) which we shall refer to as QLSA-2. In (27), it was shown that under similar caveats of QLSA-1, we have the following: For a Hermitian and invertible matrix $A \in \mathbb{C}^{2^n \times 2^n}$, vector $b \in \mathbb{C}^{2^n}$, given oracles to prepare A and b in $\mathcal{O}(polylog(N))$, and a prescribed precision of $\epsilon > 0$, there exists an algorithm that computes a solution x such that $|||x\rangle - |A^{-1}b\rangle|| \le \epsilon$ in $\mathcal{O}(polylog(N/\epsilon)\kappa)$. This work implements modified algorithms derived from both these methods (26, 27) by developing an efficient LCU decomposition strategy. In the QCFD context, we now explore methods suitable for preparing $\{\mathbf{b}_{be1}, \mathbf{b}_{be2}, \mathbf{b}_{fe}\}$ and the matrices $\{\mathbf{A}_{be1}, \mathbf{A}_{be2}, \mathbf{A}_{fe}\}$, to enable the postprocessing of the solution $\tilde{u}$, in order to construct an end-to-end method.

Quantum State Preparation. To prepare quantum states that encode $\mathbf{b}_{be1}$, $\mathbf{b}_{be2}$ and $\mathbf{b}_{fe}$, we implement two different methods, both offering subexponential circuit depth complexity:

QSP-1: In the case of iterative BE, the vector $\mathbf{b}_{be1}$, prepared at every time step, is generally fully dense with sparsity $s_b \sim \mathcal{O}(N_{be1})$. In the specific cases of Poiseuille and Couette flows, and for the specific initial conditions considered here, the state prepared at every time step forms a discrete log-concave distribution (i.e., $\frac{\partial^2 \log(b)}{\partial y^2} < 0$ for $\forall t \ge 0$), which could also be confirmed from the analytical solution given by Eq. **4** known for this case as:

$$u(y,t) = \sum_{k=1}^{\infty}\left[\frac{2(1-(-1)^k)}{k\pi}\left(1 + \frac{\partial p}{\partial x}\frac{Re}{(k\pi)^2}\right)\right. \qquad [4]$$

$$\left.\sin\left(\frac{k\pi y}{D}\right)e^{\frac{-t}{Re}\left(\frac{k\pi}{D}\right)^2}\right] - \frac{Re}{2}\frac{\partial p}{\partial x}y(1-y).$$

Even if the exact solution is not known, provided the initial condition is the only state preparation involved in the algorithm, flexibility exists for most flow simulations in choosing initial conditions that are log-concave. Consequently, one could invoke a Grover-Rudolph state preparation (40) technique [or its more evolved off-spring (41, 42)] to offer an efficient way to encode data. Two comments are useful. i) Though this method could be used for arbitrary state-vectors (at the cost of exponential circuit depth), for an efficient state preparation, some information on the functional form of the state needs to be known a priori—from analytical solutions, classical CFD, or by the measurement of the quantum circuit at intermittent time-steps, peeking into its instantaneous functional form. Here, we implement a similar method, which we shall refer to as QSP-1, based on (40, 43), where it was shown that: Given a vector $\mathbf{b}_{be1} \in \mathbb{R}^{N_{be1}}$, state $\mathbf{b'}_{be1}$ can be prepared such that, $||\mathbf{b}_{be1}\rangle - |\mathbf{b'}_{be1}\rangle| < \mathcal{O}(1/poly(N_{be1}))$ in $\mathcal{O}(\log(N_{be1}))$ steps. ii) Measuring all qubits of the register $(\sim \mathcal{O}(N_{be1}))$ at every time step compromises the exponential speed-up and could introduce measurement errors. However, such a method of recursive state preparation and measurement could still prove to be useful with quantum advantage for a very small number of qubits (44). In any case, we implement this method here to explore whether such a BE scheme gives accurate results with or without quantum advantage.

QSP-2: In the case of the one-shot methods, an alternative quantum state preparation method can be considered since $\mathbf{b}_{fe}$ is generally larger in size $\sim \mathcal{O}((m+p)N_g)$ (this discussion applies similarly to $\mathbf{b}_{be2}$; refer to *SI Appendix*, section 1A for definition of terms used here). When considered together with all other registers that are initially set to $|0\rangle$, $\mathbf{b}_{fe}$ is a highly sparse state vector with $s_b \sim \mathcal{O}(N_g m)$. For such states, we implement a sparse state preparation protocol (45), which we shall refer to as QSP-2. It provides an optimal circuit depth that scales only polynomially with vector size. This method involves constructing decision trees forming an alternative way to represent quantum states. Careful optimization on the structure of these trees leads to efficient state preparation whose complexity depends on the number of continuous pathways in the resulting tree structure. Thus, rephrasing here the result in ref. 45, we have the following: Given an n-qubit initial state of size $N = 2^n$, all set to $|0\rangle$, except for a sparse vector subspace $\mathbf{b}_{fe} \in \mathbb{R}^{N_{fe}}$ $(N_{fe} = \mathcal{O}((m+p)N_g))$, with sparsity $s_b = m \ll N$, then with only single qubit and CNOT gates, one can prepare such a state within $\mathcal{O}(2kn)$ time, $k \times \mathcal{O}(n)$ CNOT gates and using 1 ancillary qubit, where $k(\le m)$ is the number of branch paths of the decision tree.

Both QSP-1 and -2 are elucidated with examples in *SI Appendix*, section 3.

Flow Simulation Results

We construct and solve the system given by Eq. **3** for $N_g = 10$ and $Re = 10$. We observe that the quantum solutions for the velocity field capture the physics both qualitatively and quantitatively. To discuss closely the utility of QFlowS, we consider results from QLSA-1. As shown in Fig. 2 *A* and *B*, the converged steady state solution (using iterative BE) undershoots the analytic solution for 7 qubits and performs better with a higher number of qubits $(Q_{PE} \ge 12)$. These qubits refer to the ones allocated for the quantum phase estimation (QPE) algorithm, which in turn decides the quantum numerical precision; Q is the total number of qubits in the circuit. Similarly, the converged solution for the one-shot FE and BE cases also becomes more accurate with respect to both analytical and classical CFD solutions, with increasing number of qubits, as seen in Fig. 2 *C* and *D*, respectively.

Our experience is that, between the three schemes, the one-shot FE and BE turns out to be more accurate than iterative BE for increasing number of qubits, as shown in Fig. 3*A*, where the error ϵ_{rms} is computed with respect to the analytical solution. Among the two, one-shot schemes, though quantitatively their error behavior is nearly the same, i) we see some spurious oscillation–

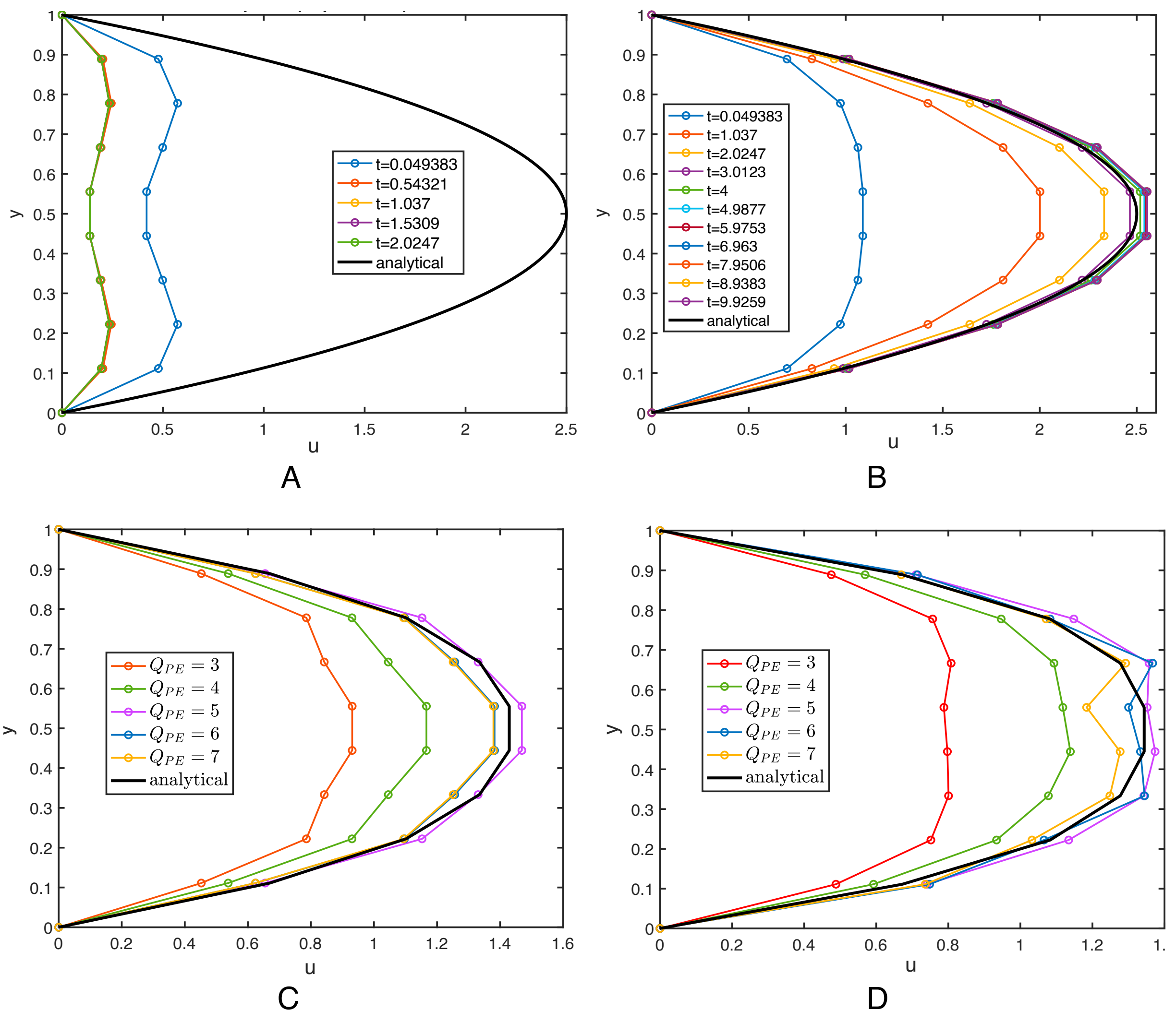

Fig. 2. (*A* and *B*) show the quantum simulation of the flow field evolving forward in time toward steady state (analytical parabolic solution shown as solid black line) using BE scheme that uses 7 and 14 qubits [3 and 10 QPE qubits (Q_{PE}), respectively], for $N = 10$, $Re = 10$, $\partial p/\partial x = -2$ and $dt = 0.01$. The accuracy of the converged solution improves for higher Q_{PE}. Here, the velocity field is solved for at every time step; panel (*C*) shows increasingly accurate converged solutions with increasing $Q_{PE} \in \{3, 4, 5, 6, 7\}$, but solved using the FE scheme where the velocity field is solved for all time steps in one shot, and only the final solution is extracted. Here, $\alpha = 0.5$ is set to meet the von Neumann stability criterion, and the parameter $T_0^* = 5.0$ is fixed. Panel (*D*) also shows the one-shot method, solved with a BE scheme and $T_0^* = 8.5$. Though $\alpha = 0.5$ is set to maintain the same time step size as (*C*) for comparison, there is no hard constraint given that the method is unconditionally stable.

like error in the velocity profile for the BE case as seen in Fig. 2*D* for higher Q_{PE}. ii) The BE case, however, has no stability-based restrictions as the FE case making it more flexible on the choice of dt. iii) When accuracy in temporal discretization is of the concern, the one-shot FE fares better. The performance can also be measured by computing fidelity, which quantifies the degree of overlap of the quantum solution with respect to the classical ($|\mathbf{u_Q} \cdot \mathbf{u_C}| \leq 1$.) This is plotted in the *Inset* of Fig. 3*A*, which shows the BE to perform better than FE. However, fidelity might not always be a good indicator to performance as illustrated in *SI Appendix*, section 1C. Both QLSA-1 and QLSA-2 rely on a variant of the phase estimation algorithm which contributes most to the total error QPE; in QLSA-2, the Gapped Phase Estimation (GPE) is rather computationally inexpensive and less erroneous than in QLSA-1 (16). In the case of phase estimation, the operator/matrix under consideration is exponentiated first as e^{iAT_0}, where T_0 is the Hamiltonian simulation time. An optimal choice T_0^* (unknown a priori) arranges the eigenvalues λ_j that are spectrally decomposed in the basis of A as $\sum_j e^{i\lambda_j T_0^*}|u_j\rangle\langle u_j|$ producing the best Q_{PE}-bit binary representation $|\tilde{\lambda}_j\rangle_{Q_{PE}}$. This choice also minimizes possible truncation errors and any spurious quantum numerical diffusion.

In case of QLSA-1, it is important to note that the smallest eigenvalue will contribute most to the error, which eventually focuses our interest in estimating λ^{-1}. Therefore, ensuring that the smallest value representable with Q_{PE} qubits (*least count* of QPE) $= 2^{-Q_{PE}}$ is $\leq \tilde{\lambda}_{min}$ is essential. The error for all cases shown in Fig. 3*A* has a gradual step-like decay because increasing Q_{PE} in small steps (of $\mathcal{O}(1)$) does not lower the least count appreciably (in $\log_{10}$ or $\log_e$) as Q_{PE} gets larger. In case of QLSA-2, though it avoids a full-blown QPE, the right choice of T_0 [for the Fourier approach (16)], the LCU coefficients and basis, along with quantum amplitude amplification, is still crucial for better accuracy.

When we probed further at the level of the flow field, the choice of T_0 seemed to exhibit nontrivial effects; for instance, in the iterative BE case, when gradually increasing Q_{PE} qubits, the converged solution either undershot or overshot the analytical

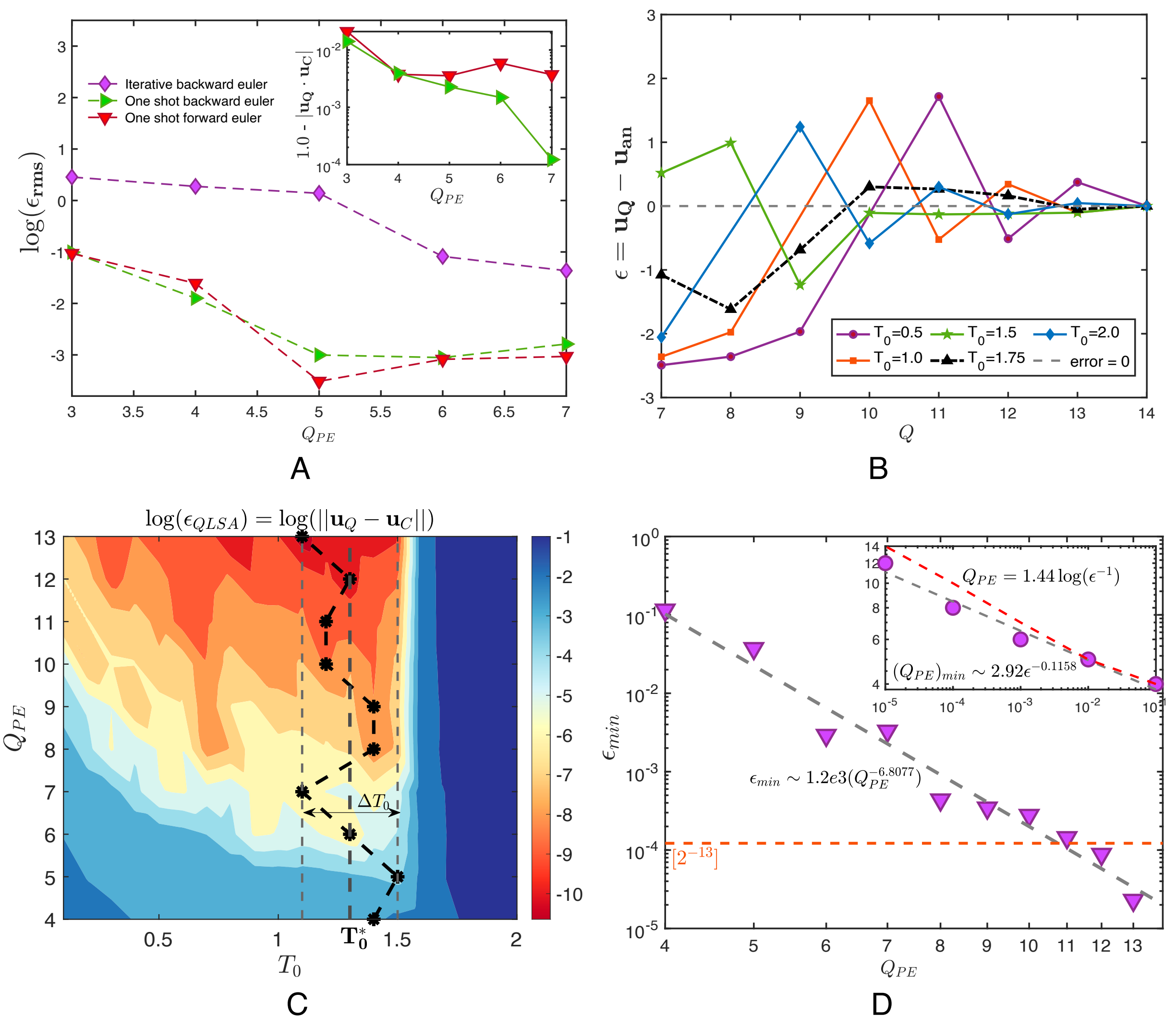

Fig. 3. (*A*) shows the absolute RMSE for the BE and FE cases shown in Fig. 2 *A–D* against QPE qubits of both the step-by-step and one-shot methods. To compare directly the one-shot BE and FE methods, we show in the *Inset* the quantity (1-fidelity) as a function of Q_{PE}. Panel (*B*) shows the absolute error computed with respect to the analytical solution as a function of Q for varying T_0. The black dotted line joining upright triangles shows the case for which the error magnitude and oscillation around the zero line are the least. Panel (*C*) shows the contour plot of the QLSA root mean square error ϵ_{QLSA} computed with respect to the classical inversion solution for different $TQ = (Q_{PE}, T_0)$ pairs. The dotted black line traces the locus of the least/minimum error for each (T, Q) pair in that range, which shows an oscillation around a unique median value T_0^* specific to a given matrix (or κ). The color code is given by the thin vertical bar to the right. Panle (*D*) shows the decay in the $(\epsilon_{QLSA})_{min}$, extracted from (*C*) with respect to only Q_{PE} for a fixed $\kappa = 18.8795$ and $T_0^* = 1.3$. The *Inset* plots the Q_{PE} (ordinate) required to achieve a specified ϵ_{QLSA} (abscissa).

solution initially. This is captured in Fig. 3*B*, where the error ϵ (with respect to the analytical solution $\mathbf{u_{an}}$ of the center line velocity solution) oscillates around $\epsilon = 0$ before converging to it for $Q_{PE} > 12$. For the specific case shown here, a choice of $T_0^* = 1.75$ (dotted black line) has the least oscillation of the error and best accuracy.

We can now ask what combination $TQ = (T_0, Q_{PE})$ gives the least error. To answer this better, we take a sample matrix equation system (of size 8×8 and $\kappa = 18.8795$) and solve it for different TQ. We then make a contour plot of the QLSA error $\epsilon_{QLSA} = ||u_Q - u_C||$, as shown in Fig. 3*C* and trace the path of least error ϵ_{min} for each TQ. Further, the range of the T_0 scan can be reduced with some initial estimates to the lower and upper bounds of λ_{min} and λ_{max} (46) such as, $\beta_1 - \beta_2\sqrt{N-1} \leq \lambda_{min} \leq \beta_1 - \beta_2/\sqrt{N-1}$ and $\beta_1 + \beta_2/\sqrt{N-1} \leq \lambda_{max} \leq \beta_1 + \beta_2\sqrt{N-1}$, where $\beta_1 = \mathrm{Tr}(A)/N$ and $\beta_2 = (\mathrm{Tr}(A^2)/N - \beta_1^2)^{1/2}$. We observe that the optimal T_0^* for all combinations lies in a fairly small range $\Delta T_0 \sim 0.1$. This is a unique value lying along the median of this range, $T_0^* \approx 1.3$ for which the system performs best. This means that for such a T_0^*, all or most eigenvalues are best represented in binary form with Q_{PE} qubits (one or some of the eigenvalues could also turn out to be represented exactly). Further, given T_0^*, with increasing number of qubits, the minimum error exhibits a power law decay $\epsilon_{min} \sim Q_{PE}^{-6.81}$ as shown in Fig. 3*D*, reaching $\sim 10^{-5}$ at around 13 qubits. The exponent becomes increasingly negative with decreasing κ, since the range of eigenvalues becomes smaller and more eigenvalues tend to be easily representable with a given number of qubits. The thick horizontal red line shows the least count for $Q_{PE} = 13$; here, $\epsilon_{min} < 2^{-13}$.

This favorable possibility arises because a subspace of the solution set could have had near exact representation using the given number of qubits, which lowers the overall L2 error $\leq 2^{-Q_{PE}}$. This means the minimum number of qubits needed to attain an error ϵ grows as a power law $(Q_{PE})_{min} \sim 2.92\epsilon^{-0.1158}$, as shown in the *Inset* of Fig. 3*D*. If not for the right choice of T_0^*, for $Q_{PE} > 3$, one would spend a larger number of qubits $\mathcal{O}(1.44\log(\epsilon^{-1}))$ to lower the overall error. We note that ϵ as computed here suppresses the error from finite differences, which is $\epsilon_{fd} \sim \mathcal{O}((\Delta y)^2, \Delta t)$ and it is important to note that this error plagues both quantum and classical solutions.

Thus, being able to estimate T_0^* fairly accurately reduces the overall computational resources required as well as the error, making it amenable for NISQ devices. Though there have been several analytical, asymptotic prescriptions for the choice of T_0^* (16, 26, 47), the exact choice remains elusive. To shed better light, QFlowS is equipped with QPE optimizer subroutine which, on the basis of the nature of the flow problem and the numerical method (finite differences) used, estimates T_0^* by very minimal classical preprocessing. Since κ decides the range of eigenvalues and the invertibility of the matrix, it forms a common link that characterizes matrices for different systems with similar sparsity. Therefore, a relation that uniquely connects κ with T_0^* would make a reasonable basis for prescribing T_0^* for different system configurations. We provide such a relation which, though generalizable in behavior, is specific to: 1) the class of elliptic and parabolic linear PDEs considered here; 2) finite difference–based numerical formulations that give rise to either sparse, band diagonal, lower triangular, Toeplitz or circulant matrices.

Since the one-shot FE and BE seem to perform better than iterative BE, we take the matrix system of the former (one-shot FE) for $N_g = 10$ and characterize how κ varies with the matrix size (here, we drop the constant factor N_g), $m = \lceil \log_2(T/dt) \rceil$ and viscosity $\nu = 1/Re$. Fig. 4*A* shows for a specific case of $T = 1$, $dt = 0.001$, κ grows as a stretched exponential with decreasing (increasing) ν (Re) and saturates for very low ν, while for a fixed $\nu = 0.1$, κ grows exponentially with m as shown in the *Inset* of Fig. 4*A*. However, the overall behavior of κ with both ν and the system size m is shown in Fig. 4*B*, where the $\kappa - \nu$ curves transition from exponential to stretched exponential fits as plotted for increasing m obtaining the relation $\kappa \leq m(e^{-0.02m\nu} + 2)$. This relation confirms that κ is bounded and is not exponentially large. Here, we explicitly highlight the dependence of κ on ν and note that it is also bounded from above by $\kappa = 3(m + p + 1)$, as given in ref. 13 for nonlinear PDEs that generally have higher κ than linear PDEs. In both cases, κ increases with Re and m.

We now proceed to compute T_0^* for increasing κ, by extracting, as before, the TQ phase diagram and obtaining the relation $T_0^* \sim -0.363\log(\kappa) + 0.918$. We explore only a small zone of the phase diagram. The effect of ν is as expected, with T_0^* decreasing and saturating for very low values. This relation between T_0^* and κ obtained via simulations on QFlowS serves as a basis for choosing T_0^* to perform accurate simulations for bigger circuit sizes as well. The earlier relation (despite a slight variation with matrix structure) forms a reasonable approximation for T_0^* for the system considered here. However, the process outlined here should be repeated to evaluate a problem-specific estimate of T_0^*; we have only presented a working level estimate of T_0^* within a certain range to aid the eigenvalue estimation process. It predicts optimal T_0^* for Hamiltonian simulation algorithms for both QLSA-1 and -2. In case of QLSA-2, QFlowS also efficiently generates the set of the LCU coefficients for optimal performance. Further, even for other classes of problems (different PDEs and discretization schemes), QFlowS's QPE optimizer could be employed to perform similar low-cost classical preprocessing to suggest optimal T_0^* for accurate and efficient fluid flow simulations. Further, barring minor quantitative differences, on performing a similar analysis on the Couette flow case, we find that the qualitative outcome and inferences drawn are nearly the same as that of the Poiseuille flow case as seen here. The corresponding velocity profiles for the Couette flow are shown in *SI Appendix*, section 1C.

Quantum Postprocessing Protocol (QPP)

Once the velocity field is obtained, measuring it by repeated execution [excluding the requirements of quantum amplitude amplification (48)] of the quantum circuit ($\mathcal{O}(N)$ complexity) will compromise any quantum advantage and also introduce measurement errors. Here, we examine a QPP that produces just one real-valued output of the average viscous dissipation rate per unit volume, $\varepsilon = \nu\langle(\frac{\partial u}{\partial y})^2\rangle$. Given that we are equipped with

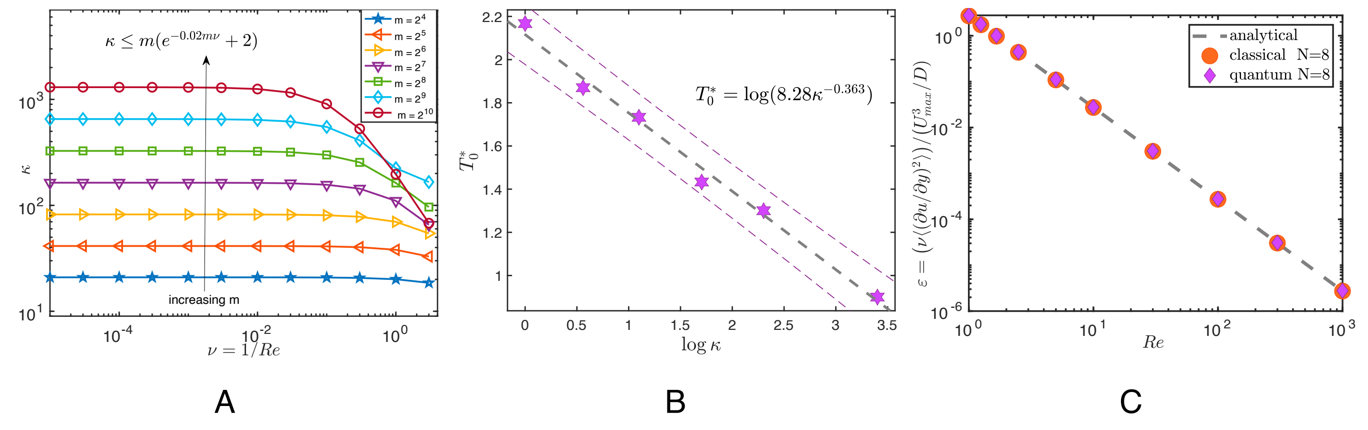

Fig. 4. Panel (*A*) shows the behavior of κ with increasing m and decreasing ν. For $\nu < 0.01$, κ saturates to a constant that is decided largely by m. This is expected since for Poiseuille flow, by appropriately scaling time with ν (using $t^* = \nu t/L^2$), ν can be absorbed into the equation. However, this is not the case when $\mathbf{C} \neq 0$ in Eq. **1**, in which case ν would have a much stronger effect on κ. (*B*) Using QFlowS, a small sample space of low dimensional matrices for varying κ is solved to compute T_0^* in each case, as shown in Fig. 3*C*, which yields a power law scaling of T_0^* against κ, as marked in the figure. This feature can be used to predict T_0^* for higher dimensional matrices. T_0^* is insensitive to κ below a threshold value of ν for these problems. Thin dashed lines indicate 99% levels. Panel (*C*) shows the mean viscous dissipation rate ε, normalized by the center-line velocity ($U_{max} = (-\partial p/\partial x)/(4\nu)$), with respect to increasing Re (decreasing ν) computed from the steady-state analytical, classical and quantum solutions.

an U_V oracle[†] that prepares a quantum register with the velocity field solution, we append to it a derivative module that computes first $\frac{\partial u}{\partial y}$ of the solution depicted in the circuit diagram shown in Fig. 1*A*. This is done by either 1) the LCU method where a finite difference matrix of first derivative is decomposed as LCU or 2) a spectral method in which an IQFT is first applied to enter the conjugate space and the first derivative is now a simple scalar multiplication with the corresponding wavenumber, k. Finally, the application of QFT transforms it back into real space. Here, we implement the former method, given that the boundary conditions are nonperiodic.

At this point, if one is interested in general nonlinear functions such as trigonometric, logarithmic, square root, or higher powers, we implement the following procedure. The derivatives that are stored as quantum amplitudes are first converted into an n_m-bit binary representation using QADC (49). Following this step, a direct squaring algorithm outlined in *SI Appendix*, section 4, or a binary quantum arithmetic squaring circuit (an inverse of the square-root algorithm, which is more expensive than the former; see ref. 50) is used to compute $(\frac{\partial u}{\partial y})^2$, which are finally converted back into amplitude encoding using a quantum digital–analog converter (QDAC) (49). This algorithm requires $\mathcal{O}(1/\epsilon_{QPP})$ calls to the controlled-U_V oracles that has a complexity of $\mathcal{O}\, polylog(N/\epsilon)\kappa$, thus an overall complexity of $\mathcal{O}(\text{polylog}(N/\epsilon)\kappa/\epsilon_{QPP})$), one query to the bit-squaring algorithm with complexity $\mathcal{O}((\log_2 N)^2)$ and $\mathcal{O}((\log_2 N)^2/\epsilon_{QPP})$ single- and two-qubit gates. While computing the dissipation using QPP, we employ QFlowS with all its submodules, except the squaring step that is implemented as a separate circuit for proof-of-concept. In general, for larger system sizes, each step of the QPP has to be implemented and simulated as distinct entities, since running them together exceeds the current simulator capacity. The outline of the QPP algorithm introduced here along with circuit implementation is given in circuit *SI Appendix*, section 4.

As a final step, we apply a matrix U_{avg} that computes the sum of derivatives at all points into one qubit, the measurement of which, along with a few ancillas,[‡] produces an output of the desired ε (after some normalization and multiplication by ν). Applying QPP on the quantum solution yields the variation of ε with *Re*, computed near the steady-state, as shown in Fig. 4*C*. The transient from the uniform to the final, near-parabolic state is characterized by large gradients and warrants higher resolution. To keep the quantum resources within limits, we choose the well-behaved, nearly steady-state profiles to compute dissipation. Further, even though in Fig. 4*A* we explore large $Re \leq 10^5$ (where it is treated purely as a parameter to study bounds of κ), we compute dissipation only up to $Re = 1{,}000$. This is because, physically, Eq. **3** is well suited to describe only low *Re* flows. The normalized dissipation computed from the quantum solutions follows closely the classical and analytical results. The same finite difference–based postprocessing is used for all the three solutions to enable reasonable comparisons. In addition to the finite difference errors of classical solution, the quantum solution suffers from an additional error due to the quantum algorithms. Therefore, by trading off with a smaller resolution ($N = 8$), we obtain the quantum solutions by employing a larger number of QPE qubits, so that the overall error due to finite differences is at least comparable to its quantum counterparts (error in the quantum estimates, for instance, with a modest $Q_{PE} = 8$ and $Re \sim 10$, is $\epsilon \sim 10^{-3}$). This could be made more accurate with more of the QPE qubits and the resolution N_g, as seen before. In essence, this demonstrates the possibility for computing quantities such as ε effectively as a quantum postprocessing step.

SI Appendix, section 4 includes a table of complexity formulae for the relevant QC subroutines used in this paper.

Discussion

We have demonstrated here a possible quantum algorithmic strategy and its full implementation using gate-based quantum circuits on QFlowS, to simulate Poiseuille and Couette flows in an end-to-end manner. First, we identify suitable quantum state preparation algorithms by considering the sparsity and functional forms of the initial velocity data being encoded. In CFD, it is generally admissible to choose a relatively simple form and a sparse initial condition, which would result in a relatively low cost of state preparation. QSP-1,2 could both be used as shown here, by assessing the form of input to encode initial and boundary conditions with subexponential complexity ($\mathcal{O}(\log(N_{be}))$ and $\mathcal{O}(kn)$, respectively), as well as to reinitialize instantaneous velocity fields. However, data that are dense with no functional form would force an exponential circuit depth.

Second, using finite difference schemes, the governing equations were discretized to form a linear system of equations and solved by implementing QLSA-1 and -2, which are state-of-the-art, high-precision algorithms with potentially up to an exponential advantage compared to classical schemes. Here, we have made a detailed analysis of the behavior of the velocity solutions and the attendant errors, which have revealed that FE outperforms BE. Further, we proposed the role of T_0 and discussed algorithms to prescribe the optimal value, T_0^*. The power-law and exponential form relations of $(\epsilon_{min} - Q_{PE})$ and $(T_0^* - \kappa)$, respectively, given by QFlowS, form a well-informed basis to choose T_0^* and the minimum required qubits to perform accurate (up to ϵ_{min}) and qubit-efficient flow simulations. Though QLSA-2 avoids a standard QPE and can provide exponential advantage in precision $\mathcal{O}(\text{polylog}(N/\epsilon)\kappa)$, other methods based on adiabatic QC (51) could have potentially simpler implementations, while offering similar performance, motivating further investigations.

To keep the discussion compact, the data reported here are taken mainly from QLSA-1; a similarly detailed discussion on QLSA-2 forms the bulk of the upcoming work. In QLSA-2, the critical factors computed are the coefficients for LCU, GPE parameters, and a comparison between the Fourier and Chebyshev approaches. Further, we have introduced a QPP protocol, where we propose the computation of the viscous dissipation rate ε, using a specific combination of QFT, IQFT, QADC, QDAC, and bit-arithmetic, with an overall complexity that scales as $\mathcal{O}(\text{polylog}(N/\epsilon)\kappa/\epsilon_{QPP} + ((\log_2 N)^2/\epsilon_{QPP}))$. The QPP introduces an extra $\mathcal{O}(1/\epsilon_{QPP})$ scaling that brings down the performance of QLSA-2 to the level of QLSA-1 (if $\epsilon \approx \epsilon_{QPP}$). Added to this, for purposes of quantum amplitude amplification, QFlowS is capable of repeating circuit runs in parallel (currently tested up to ~8,000 shots, where a shot simply means a repeated run of the quantum circuit to extract quantum state statistics).

We should point out that this method avoids measuring the entire velocity field, thus protecting it from compromising the quantum advantage and escalating possible measurement errors. We observe that ε computed from the resulting quantum

[†]Oracles are black-boxes that represent a quantum algorithm or subroutine with a specific operation. Here, by U_V, we refer to QSP-1,-2 or QLSA-1,-2).

[‡]Ancilla qubits are auxiliary or scratch-pad qubits used to facilitate a specific quantum operation.

simulations captures the known analytical results. This method can be extended to compute other nonlinear functions of the velocity field.

In summary, we have demonstrated a complete implementation of an end-to-end algorithm to perform fluid flow simulations using QC, which paves the way for future QCFD simulations of both linear and nonlinear flows. We have also introduced here a quantum simulator package QFlowS—designed mainly for CFD applications. It is important to note that we have not addressed other key challenges such as noise and quantum error correction, which we emphasize are critical to simulations on near-term quantum devices. The complexities of the algorithms provided here (*SI Appendix*, section 4) are estimates. Ongoing efforts include investigations of higher-order finite difference schemes, comparative studies of different amplitude amplification approaches, detailed error, and complexity analyses, along with computing exact gate counts and circuit depths. Extending these methods and tools to nonlinear systems such as Burgers equations and Navier–Stokes equations is also part of this effort.

Data, Materials, and Software Availability. The package QFlowS is available as an open-source package on GitHub (52). Readers interested in using QFlowS can write to the authors. All other data are included in the manuscript and/or *SI Appendix*.

ACKNOWLEDGMENTS. We wish to thank Dhawal Buaria, Yigit Subaśi, Jörg Schumacher, Balu Nadiga, Patrick Rebentrost, Stefan Wörner, and Philipp Pfeffer for insightful discussions and the referees for helpful comments. S.S.B acknowledges the computational resources provided by the New York University Greene supercomputing facility on which these simulations were performed.

1. K. P. Iyer, J. D. Scheel, J. Schumacher, K. R. Sreenivasan, Classical 1/3 scaling of convection holds up to $Ra = 10^{15}$. *Proc. Natl. Acad. Sci. U.S.A.* **117**, 7594–7598 (2020).
2. P. K. Yeung, K. Ravikumar, Advancing understanding of turbulence through extreme-scale computation: Intermittency and simulations at large problem sizes. *Phys. Rev. Fluids* **5**, 110517 (2020).
3. P.-K. Yeung, K. Ravikumar, S. Nichols, Turbulence simulations on the verge of exascale: GPU algorithms and an alternative to long simulations at high resolutions. *Bull. Am. Phys. Soc.* **Q49**, 004 (2022).
4. J. S. Rood *et al.*, *Enabling Combustion Science Simulations for Future Exascale Machines* (National Renewable Energy Laboratory NREL, Golden, CO, 2021).
5. K. P. Iyer, S. S. Bharadwaj, K. R. Sreenivasan, The area rule for circulation in three-dimensional turbulence. *Proc. Nat. Acad. Sci. U.S.A.* **118**, e2114679118 (2021).
6. D. Buaria, K. R. Sreenivasan, Scaling of acceleration statistics in high Reynolds number turbulence. *Phys. Rev. Lett.* **128**, 234502 (2022).
7. G. Bell, D. H. Bailey, J. Dongarra, A. H. Karp, K. Walsh, A look back on 30 years of the Gordon bell prize. *Int. J. High Perform. Comput. Appl.* **31**, 469–484 (2017).
8. M. A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, ed. 10th Anniversary, 2010).
9. S. Jordan, Quantum algorithm zoo (2022). https://quantumalgorithmzoo.org/. Accessed 29 June 2023.
10. D. D. Awschalom *et al.* "A roadmap for quantum interconnects" (Tech. Rep., Argonne National Lab (ANL), Argonne, IL, 2022).
11. Y. Alexeev *et al.*, Quantum computer systems for scientific discovery. *PRX Quantum* **2**, 017001 (2021).
12. S. S. Bharadwaj, K. R. Sreenivasan, Quantum computation of fluid dynamics. *Ind. Acad. Sci. Conf. Ser.* **3**, 77–96 (2020).
13. J.-P. Liu *et al.*, Efficient quantum algorithm for dissipative nonlinear differential equations. *Proc. Natl. Acad. Sci. U.S.A.* **118**, e2026805118 (2021).
14. Y. T. Lin, R. B. Lowrie, D. Aslangil, Y. Subaşı, A. T. Sornborger, Koopman von Neumann mechanics and the Koopman representation: A perspective on solving nonlinear dynamical systems with quantum computers. arXiv [Preprint] (2022). http://arxiv.org/abs/2202.02188 (Accessed 29 June 2023).
15. D. Giannakis, A. Ourmazd, P. Pfeffer, J. Schumacher, J. Slawinska, Embedding classical dynamics in a quantum computer. *Phys. Rev. A* **105**, 052404 (2022).
16. A. M. Childs, J.-P. Liu, A. Ostrander, High-precision quantum algorithms for partial differential equations. *Quantum* **5**, 574 (2021).
17. G. T. Balducci, B. Chen, M. Möller, M. Gerritsma, R. De Breuker, Review and perspectives in quantum computing for partial differential equations in structural mechanics. *Front. Mech. Eng.* **8**, 75 (2022).
18. F. Y. Leong, W.-B. Ewe, D. E. Koh, Variational quantum evolution equation solver. *Sci. Rep.* **12**, 10817 (2022).
19. M. Lubasch, J. Joo, P. Moinier, M. Kiffner, D. Jaksch, Variational quantum algorithms for nonlinear problems. *Phys. Rev. A* **101**, 010301 (2020).
20. F. Gaitan, Finding flows of a Navier–Stokes fluid through quantum computing. *npj Quantum Inf.* **6**, 61 (2020).
21. F. Oz, O. San, K. Kara, An efficient quantum partial differential equation solver with Chebyshev points. *Sci. Rep.* **13**, 7767 (2023).
22. B. N. Todorova, R. Steijl, Quantum algorithm for the collisionless Boltzmann equation. *J. Comput. Phys.* **409**, 109347 (2020).
23. W. Itani, K. R. Sreenivasan, S. Succi, Quantum algorithm for lattice Boltzmann (QALB) simulation of incompressible fluids with a nonlinear collision term. arXiv [Preprint] (2023). http://arxiv.org/abs/2304.05915 (Accessed 29 June 2023).
24. L. Budinski, Quantum algorithm for the advection-diffusion equation simulated with the lattice Boltzmann method. *Quantum Inf. Process.* **20**, 57 (2021).
25. N. Gourianov *et al.*, A quantum-inspired approach to exploit turbulence structures. *Nat. Comput. Sci.* **2**, 30–37 (2022).
26. A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **103**, 150502 (2009).
27. A. M. Childs, R. Kothari, R. D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM J. Comput.* **46**, 1920–1950 (2017).
28. S. Aaronson, Read the fine print. *Nat. Phys.* **11**, 291–293 (2015).
29. Y. Cao, A. Papageorgiou, I. Petras, J. Traub, S. Kais, Quantum algorithm and circuit design solving the Poisson equation. *New J. Phys.* **15**, 013021 (2013).
30. A. Montanaro, S. Pallister, Quantum algorithms and the finite element method. *Phys. Rev. A* **93**, 032324 (2016).
31. Quantiki, List of quantum simulators (2023). https://quantiki.org/wiki/list-qc-simulators. Accessed 29 June 2023.
32. A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, B. Valiron, "Quipper: A scalable quantum programming language" in *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation* (2013), pp. 333–342.
33. T. Jones, A. Brown, I. Bush, S. C. Benjamin, Quest and high performance simulation of quantum computers. *Sci. Rep.* **9**, 1–11 (2019).
34. A. C. Vazquez, S. Woerner, Efficient state preparation for quantum amplitude estimation. *Phys. Rev. Appl.* **15**, 034027 (2021).
35. V. Giovannetti, S. Lloyd, L. Maccone, Architectures for a quantum random access memory. *Phys. Rev. A* **78**, 052310 (2008).
36. I. M. Georgescu, S. Ashhab, F. Nori, Quantum simulation. *Rev. Mod. Phys.* **86**, 153 (2014).
37. D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, R. D. Somma, "Exponential improvement in precision for simulating sparse hamiltonians" in *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing* (2014), pp. 283–292.
38. D. W. Berry, A. M. Childs, R. Kothari, "Hamiltonian simulation with nearly optimal dependence on all parameters" in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science* (IEEE, 2015), pp. 792–809.
39. A. Ambainis, Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. arXiv [Preprint] (2010). http://arxiv.org/abs/1010.4458 (Accessed 29 June 2023).
40. L. Grover, T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions. arXiv [Preprint] (2002). http://arxiv.org/abs/quant-ph/0208112 (Accessed 29 June 2023).
41. A. G. Rattew, B. Koczor, Preparing arbitrary continuous functions in quantum registers with logarithmic complexity. arXiv [Preprint] (2022). http://arxiv.org/abs/2205.00519 (Accessed 29 June 2023).
42. A. C. Vazquez, R. Hiptmair, S. Woerner, Enhancing the quantum linear systems algorithm using Richardson extrapolation. *ACM Trans. Quantum Comput.* **3**, 1–37 (2022).
43. A. Prakash, *Quantum Algorithms for Linear Algebra and Machine Learning* (University of California, Berkeley, CA, 2014).
44. P. Pfeffer, F. Heyder, J. Schumacher, Hybrid quantum-classical reservoir computing of thermal convection flow. *Phys. Rev. Res.* **4**, 033176 (2022).
45. F. Mozafari, G. De Micheli, Y. Yang, Efficient deterministic preparation of quantum states using decision diagrams. *Phys. Rev. A* **106**, 022617 (2022).
46. H. Wolkowicz, G. P. H. Styan, More bounds for eigen values using traces. *Linear Algebra Appl.* **31**, 1–17 (1980).
47. A. Scherer *et al.*, Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2D target. *Quantum Inf. Process.* **16**, 1–65 (2017).
48. G. Brassard, P. Hoyer, M. Mosca, A. Tapp, Quantum amplitude amplification and estimation. *Contemp. Math.* **305**, 53–74 (2002).
49. K. Mitarai, M. Kitagawa, K. Fujii, Quantum analog–digital conversion. *Phys. Rev. A* **99**, 012301 (2019).
50. S. Wang *et al.*, Quantum circuits design for evaluating transcendental functions based on a function-value binary expansion method. *Quantum Inf. Process.* **19**, 1–31 (2020).
51. Y. Subasi, R. D. Somma, D. Orsucci, Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing. *Phys. Rev. Lett.* **122**, 060504 (2019).
52. S. S. Bharadwaj, Rustling-leaves. QFlowS. Github. https://github.com/rustling-leaves/QFlowS.git. Deposited 16 November 2023.