RESEARCH ARTICLE | MAY 29 2025

Quantum smoothed particle hydrodynamics algorithm inspired by quantum walks

Special Collection: Proceedings of the 33rd Conference on Discrete Simulation of Fluid Dynamics

R. Au-Yeung 🖾 💿 ; V. M. Kendon 💿 ; S. J. Lind 💿

Check for updates

Physics of Fluids 37, 057141 (2025) https://doi.org/10.1063/5.0268240





Physics of Fluids

Special Topics Open for Submissions

Learn More



Export Citatio

View Online

Quantum smoothed particle hydrodynamics algorithm inspired by quantum walks

Cite as: Phys. Fluids **37**, 057141 (2025); doi: 10.1063/5.0268240 Submitted: 28 February 2025 · Accepted: 21 April 2025 · Published Online: 29 May 2025



AFFILIATIONS

¹Physics Department, University of Strathclyde, Glasgow G4 ONG, United Kingdom ²School of Engineering, Cardiff University, Cardiff CF24 3AA, United Kingdom

Note: This paper is part of the Special Topic, Proceedings of the 33rd Conference on Discrete Simulation of Fluid Dynamics. ^{a)}Author to whom correspondence should be addressed: rhonda.au-yeung@strath.ac.uk

ABSTRACT

Recent years have seen great progress in quantum computing, providing opportunities to overcome computational bottlenecks in many scientific applications. In particular, the intersection of computational fluid dynamics and quantum computing has become an active area of research with exponential computational speedup as an ultimate goal. In this work, we propose a quantum algorithm for the time-dependent smoothed particle hydrodynamics (SPH) method. Our algorithm uses concepts from discrete-time quantum walks to solve the one-dimensional advection partial differential equation via an SPH formalism. Hence, we construct a quantum circuit to carry out the calculations for a two-particle system over one, two, and three time steps. We compare its outputs with results from the classical SPH algorithm and show there is excellent agreement. The methodology and findings, here, are a key step toward developing a more general quantum SPH algorithm for solving practical engineering problems on gate-based quantum computers.

© 2025 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/4.0/). https://doi.org/10.1063/5.0268240

I. INTRODUCTION

Computational fluid dynamics (CFD) simulations are widely used in the automotive, aerospace, civil engineering, renewable energy, and defense industries. These applications typically rely on large-scale numerical simulations to solve the Navier–Stokes equations, running on millions of CPU cores at petaflop speeds. We are reaching the limits of what we can do with silicon chip technology and the available power for the largest high performance computing facilities. It has become clear that we need to develop new methods to perform larger and more complicated computations. Quantum computing is a particularly promising candidate for a range of computational problems. There is evidence it can surpass the most powerful high-performance computers (HPC),^{1–3} when more advanced quantum hardware has been engineered. CFD is well-placed to benefit from advances in quantum computing^{4–6} and the first steps are being taken to develop suitable quantum algorithms.

Quantum CFD algorithms fit into two broad categories. First, hybrid quantum-classical algorithms^{7–9} directly solve the equations of motion by outsourcing the parallelizable operations (e.g., solving linear systems¹⁰) to the quantum computer.^{8,11,12} Hybrid algorithms may be suitable for the current generation of noisy intermediate-scale

quantum (NISQ) computers.^{13,14} Bottlenecks in hybrid methods occur during the frequent data exchanges between classical and quantum computers—the encoding and readout processes can be more timeconsuming than the algorithm itself.^{15,16} In the second category, Hamiltonian simulation^{17,18} is better suited for fault-tolerant quantum computers. This method maps the fluid to a quantum system which evolves on the quantum processor. It does not require intermediate state measurements or re-initialization steps. For example, Hamiltonian simulation underpins the quantum lattice Boltzmann method (QLBM) developed by Succi, Fillion-Gourdeau, and Palpacelli^{19–21} and Budinski.²²

Quantum versions of random walks^{23,24} can be used for building powerful quantum algorithms.²⁵ They have already been used to develop quantum lattice Boltzmann schemes, which have been shown to be formally equivalent to quantum walks.¹⁹ Here, we use a quantum walk-based algorithm for the smoothed particle hydrodynamics (SPH) method.^{26,27}

In a previous work, we presented a proof-of-concept quantum SPH algorithm²⁸ for solving the one-dimensional advection and diffusion partial differential equations. Now, we address a major computational bottleneck in that algorithm: rather than performing quantum

encoding and readout at each time step, we explore how techniques from discrete-time quantum walks can generate multiple time steps on a quantum computer. This would make the quantum SPH algorithm more efficient by encoding the SPH parameters into the quantum computer, calculating several time steps, then, reading out the current state. We can repeat this process for longer simulations, with each readout providing a snapshot for data analysis.

This paper is intended for researchers in the fluid mechanics community who may have a limited background in quantum computing. We recommend Bharadwaj and Sreenivasan's lecture notes^{29,30} for a concise overview written from the perspective of CFD applications. Reviews by Givi *et al.*³¹ and Succi *et al.*¹⁵ also discuss the challenges facing the quantum CFD field. Nielsen and Chuang³² offer a more pedagogical introduction to quantum computing for further reference.

We want to point out some important differences between quantum and classical computing that appears in our work. A key postulate in quantum mechanics is that all quantum operations must be unitary, linear, and reversible. Quantum gates in quantum computers are implemented by unitary operators acting on one or more qubits. Unitary operators \hat{U} have the property that $\hat{U}^{-1} = \hat{U}^{\dagger}$ where \hat{U}^{-1} performs the inverse operation. Since \hat{U}^{\dagger} is well defined, the inverse operation must exist. Hence, quantum gates must be reversible. This presents a challenge for nonlinear CFD problems, such as when calculating the nonlinear transformation $u \to u^2$ (velocity u) for nonlinear flows. The treatment of nonlinearity in quantum computing remains an open question. Hence, we choose the linear advection equation as the application case in this paper.

The structure of our paper is as follows. Section I describes the different components required in building our quantum algorithm. We describe the core ideas of SPH in Sec. II A and summarize the novelty in our previous work²⁸ in Sec. II B. This involves converting the SPH formalism into expressions more suitable for quantum computers. Next, we describe how we adapt concepts from the quantum walk formalism (Sec. II C) into a quantum smoothed particle hydrodynamics (QSPH) algorithm (Sec. III). We provide a fully worked out example using a simple two-particle SPH system (Sec. IV), including numerical simulations using Qiskit software to build the quantum circuit. Then we discuss the results (Sec. IV E) and future work (Sec. V).

II. BACKGROUND

This section gives an overview of the SPH algorithm (Sec. II A), our previous work on building a quantum SPH algorithm (Sec. II B), and we describe important concepts in the quantum walk formalism (Sec. II C). We focus on the discrete-time quantum walk on a line, its coin and shift operations, and how we use them in the quantum SPH algorithm in this paper.

A. SPH core concepts

Ever since SPH was first developed in 1997 for astrophysics simulations,^{26,27} it has been refined and adapted to solve numerous other problems in science and engineering,³³ and more recently in fluid animations for computer graphics applications.³⁴ SPH is a Lagrangian method based on particle interpolation to calculate smooth field variables. These particles carry the physical properties of the system which we update at each time step. Because of its Lagrangian particle nature, SPH has certain advantages over traditional mesh-based methods. For example, SPH is generally more robust in highly deforming flows and does not suffer from mesh distortions that catastrophically affect the numerical accuracy and stability in mesh-based simulations. See Monaghan³⁵ and Lind *et al.*³⁶ for an introduction and review of the SPH method.

The foundation of SPH is interpolation theory, based on the Dirac sifting property $f(x) = \int_{\Gamma} f(y) \delta(x - y) dy$ for some smooth function f(x) and volume of the integral Γ that contains x. We take the kernel approximation by substituting the Dirac delta distribution with an interpolation function such that

$$f(x) = \int_{\Gamma} f(y) W(x - y, h) dy.$$
(1)

The smoothing length h defines the influence (support area) of the smoothing kernel W. The kernel must satisfy certain conditions, such as normalization and recovery of the delta function in the limit as h decreases to zero. Other conditions can also be imposed, e.g., compact support.³⁵ See Fig. 1 for an example sketch.

SPH relies on the kernel functions to find the derivatives of continuous fields in a discrete form. This is achieved by converting the continuous interpolation into a particle-based discretization. The resulting equations comprise a discrete mechanical system where particle interactions depend on their mutual distances and mechanical (and possibly thermodynamic) properties. As a result, SPH is consistent with both Lagrangian and Hamiltonian mechanics. This provides a further connection to quantum formalisms and quantum computing.²⁸ Note that in this work, we consider the Eulerian framework and fix the SPH particle positions so that they do not move in a Lagrangian manner. Not only is Eulerian SPH a valid computational tool in its own right, 38-40 the approach allows us to first focus on developing a fully quantum time-marching algorithm. We can also generalize to other methods, e.g., finite differences. Subsequent studies will then detail the process of moving and tracking particles in a quantum computing framework which will likely require additional quantum registers. We outline this further work in Sec. V.

kernel W(r)



37, 057141-2

particle of

interest

B. Summary of previous QSPH work

Our previous work²⁸ presents a proof-of-concept for solving the one-dimensional linear advection equation. It used quantum registers for the spatial derivative but with classical (Eulerian) explicit first-order time-stepping. We will follow the same prescription in the present work, but with the time-stepping now done in a fully quantum setting. This advection equation can be written as

$$\frac{\partial u(x,t)}{\partial t} + c \frac{\partial u(x,t)}{\partial x} = 0, \qquad (2)$$

with advection quantity u(x, t) as a function of position x and time t, and advection speed c. This equation can describe a one-dimensional soliton (for example) that moves along a line without change of form. In the SPH formalism, we can rewrite this as

$$u_j(t + \Delta t) = u_j(t) - c\Delta t \sum_k (u_k(t) - u_j(t)) \Delta x_k \nabla_j W_{jk}, \quad (3)$$

which defines the solutions u of SPH particle j in terms of neighbors k at time intervals defined by time step size Δt , with particle spacing Δx , and first derivative of smoothing kernel $W(r_j - r_k, h)$. Note that we use the zeroth-order consistent formulation for the first derivative.

The crux of our work²⁸ was to rewrite Eq. (3) into a quantum mechanical formalism,

$$u_j(t + \Delta t) = u_j(t) - c\Delta t \nu N ||\vec{a}|| \Re \langle a|\nabla_j W_{jk} \rangle, \qquad (4)$$

with normalization constant $\nu = \max |\nabla_j W_{jk}|$ and N neighbor particles inside region of compact support. We define an inner product $\langle a | \nabla_j W_{jk} \rangle$ that contains quantum states

$$\langle a| = \frac{\vec{a}^{*}}{||\vec{a}||}, \quad \vec{a}^{*} = \begin{bmatrix} (u_{1}(t) - u_{j}(t))\Delta x_{1} \\ (u_{2}(t) - u_{j}(t))\Delta x_{2} \\ \vdots \\ (u_{N}(t) - u_{j}(t))\Delta x_{N} \end{bmatrix},$$
(5)
$$|\nabla_{j}W_{jk}\rangle = \begin{bmatrix} \nabla_{j}W_{j1}/(\nu N) + ib_{j1} \\ \nabla_{j}W_{j2}/(\nu N) + ib_{j2} \\ \vdots \\ \nabla_{i}W_{jN}/(\nu N) + ib_{jN} \end{bmatrix},$$
(6)

where

$$||\vec{a}|| = \frac{1}{\sqrt{N}} \left(\int_{A}^{B} |u_{k}(t) - u_{j}(t)|^{2} dx \right)^{1/2}.$$
 (7)

We also introduce constant b to satisfy the normalization conditions of quantum states,

$$b_{jk} = \sqrt{\frac{1}{N} - \left(\frac{\nabla_j W_{jk}}{\nu N}\right)^2},\tag{8}$$

to ensure the largest absolute value is

$$\left|\frac{\nabla_j W_{jk}}{\nu N} + ib_{jk}\right|^2 = \frac{1}{N}.$$
(9)

Our notation essentially recasts the summation [Eq. (3)] into an inner product [Eq. (4)]. The latter can be calculated efficiently on a quantum processor using the swap test,^{41,42} for example.

It is possible to use quantum encoding procedures to load Eq. (4) into the quantum computer. For example, quantum amplitude encoding^{43,44} can load Eqs. (5) and (6) by storing the information in the amplitudes of the quantum state. Then, we may use various quantum algorithms to calculate the inner product, such as the swap test^{41,42} or one of its variants.⁴⁵ In the previous work, we pass the solution back to the classical computer and repeat the process to calculate the next time step. This is likely to be costly and inefficient. In this work we fix this problem by extending the quantum calculation to include several time steps.

We emphasize that this work was a proof-of-concept. We did not determine whether there is any appreciable quantum speed up. Our work solved a one-dimensional advection model to show how a quantum SPH algorithm could work (in theory), and pointed out the numerous issues that must be addressed. We also took many simplifications, such as fixing the SPH particles on a line with equal spacing. This is similar to meshbased methods: this one-dimensional SPH system is analogous to a finite difference scheme. The inherent strength of SPH is its freely moving particles, so this extra degree of freedom should be considered in future work. Extending the method to two- or three-dimensions is another priority which would make our method more suitable for solving real-world applications. However, we focus on developing a time-stepping mechanism in a quantum framework, inspired by discrete-time quantum walks.

C. Overview of quantum walks

Quantum walks (QWs)^{23,24} are important theoretical models for quantum computing.^{46,47} They form the basis of many quantum algorithms and applications,⁴⁸ such as simulating complicated fluid flows.^{49–51} QWs use quantum superposition and entanglement to provide a more powerful form of classical random walks, as the quantum walker can exist in a superposition of states. QWs are a universal quantum computation primitive⁵² and have been shown to give an exponential algorithmic speed-up.²⁵

QWs come in two flavors: continuous and discrete.⁵³ Discrete-time quantum walks (DTQWs) evolve through a sequence of coin operations that determines the quantum walker's direction of movement in position space. In continuous-time quantum walks (CTQWs), the state evolves continuously in time under a Hamiltonian defined by the graph of the position space. In this work, we use the DTQW on a line. It is realized by two procedures: the coin operator (to determine the direction of the walk) and shift operator (to transition to the new state determined by the coin). This model uses a qubit coin in \mathcal{H}_C Hilbert space and a set of position states in \mathcal{H}_P . The total Hilbert space is denoted $\mathcal{H} = \mathcal{H}_P \otimes \mathcal{H}_C$. The quantum walker evolves according to the unitary operator

$$\hat{U} = \hat{S}(1 \otimes \hat{C}), \tag{10}$$

with coin operator \hat{C} and conditional shift operator \hat{S} .

For the one-dimensional line example, the two-dimensional Hilbert space associated with the coin operator lets the walker choose two possible directions (left or right). Hence, in the unbiased case, the coin is a Hadamard operator,

$$\hat{H}_{c} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1\\ 1 & -1 \end{bmatrix}.$$
(11)

The shift operator that produces the transition $|j\rangle \rightarrow |j\pm 1\rangle$ is

$$\hat{S}_{c} = |0\rangle\langle 0| \otimes \sum_{j} |j+1\rangle\langle j| + |1\rangle\langle 1| \otimes \sum_{j} |j-1\rangle\langle j|.$$
(12)

Hence, each step made by the quantum walker corresponds to the unitary operation $\hat{U}_c = \hat{S}_c (1 \otimes \hat{H}_c)$. As we will explain in Sec. III, one QW step is analogous to one time step in the quantum SPH algorithm.

III. METHOD

We provide a high-level summary of the QSPH algorithm in Fig. 2. The idea is to solve the advection equation over several time steps. We construct a quantum circuit using gates that are analogous to quantum walk operations, namely the swap and coin operators. For convenience and to mirror notation in quantum walks, let $\Delta t = 1$ in Eq. (4). This gives an advection equation

$$u_{j}(t+1) = u_{j}(t) \left(1 + c\nu N \Re \sum_{k} \Delta x_{k} V_{jk} \right) - c\nu N \Re \sum_{k} \Delta x_{k} u_{k}(t) V_{jk},$$
(13)



FIG. 2. Schematics of quantum algorithm. Classical (quantum) procedures in orange (purple). Quantum register $|a\rangle$ contains advection velocities [Eq. (5)]. Other register contains kernel information $|\nabla_j W_{jk}\rangle$ [Eq. (6)]. Quantum algorithm encodes SPH parameters into quantum register, then calculates inner product using swap and unitary coin operations. These gates can be repeated for each time step iteration before we pass results back to classical computer.

where $V_{jk} = \nabla_j W_{jk}/(\nu N) + ib_{jk}$, \sum_k sums over neighbors k only and implies $k \neq j$. Of course, numerical studies typically require $\Delta t \ll 1$, but, equivalently, we may use $\Delta t = 1$ and instead adjust the advection speed to fulfill the Courant–Friedrichs–Lewy (CFL) condition,

$$\Delta t \le \frac{\Delta x}{c}.\tag{14}$$

For simplicity, we let $\Delta x_k = \Delta x$ so that

$$u_{j}(t+1) = u_{j}(t) \left(1 + c\nu N\Delta x \Re \sum_{k} V_{jk} \right) - c\nu N\Delta x \Re \sum_{k} u_{k}(t) V_{jk}.$$
 (15)

In the QW formalism, we define the amplitudes

$$\alpha_{jj} = 1 + c\nu N \Delta x \Re \sum_{k} V_{jk}, \qquad (16)$$

$$\alpha_{kj} = -c\nu N \Delta x \Re(V_{jk}), \qquad (17)$$

hence,

ı

$$u_j(t+1) = \alpha_{jj}u_j(t) + \sum_k \alpha_{kj}u_k(t).$$
(18)

Let there be M sites (nodes) in our QW system, analogous to number of SPH particles. There are N neighbors per site, and N < Mwith N neighbors for each site.

Next, we store scalars $u_j(t)$ in vector $\vec{u}(t)$ and use quantum amplitude encoding to create a register

$$|\vec{u}(t)\rangle = \sum_{j=1}^{M} u_j(t)|j\rangle, \qquad (19)$$

in computational basis $|j\rangle$ containing particle positions *j*. Summation over all neighbors is the most computationally expensive process that we can possible do. Hence, for each site *j*, we want to select a subset of size *N* neighbors (analogous to the kernel function selecting neighbors within its support), multiply *u* with corresponding amplitudes α , and finally, perform the summation.

We need another register to label the neighbors. At each site, we define vector $\vec{u}_j(t)$ containing N + 1 components for N neighbors with site indices k, k', k'', \ldots to produce a vector

$$\vec{u}_{j}(t) = \begin{vmatrix} u_{jj}(t) \\ u_{jk}(t) \\ u_{jk'}(t) \\ \vdots \end{vmatrix} .$$
(20)

Then, we encode the neighbor information by including a second register,

$$\vec{u}(t)\rangle = \sum_{j=1}^{M} \sum_{n}^{N} u_{jn}(t) |j,n\rangle, \qquad (21)$$

where *n* is an element of the neighbors subset. Note, we use notation $|j, k\rangle$ to represent two separate registers, where *j* encodes the particle

positions and k for the neighbors. This is equivalent to QW position and coin states.

We apply the unitary shift operator \hat{S} ,

$$\hat{S}|\vec{u}(t)\rangle = \sum_{j=1}^{M} \sum_{n}^{N} u_{nj}(t)|j,n\rangle, \qquad (22)$$

to reorganize the neighbors so that for site *j*,

$$\vec{u}_{j}(t) = \begin{bmatrix} u_{jj}(t) \\ u_{kj}(t) \\ u_{k'j}(t) \\ \vdots \end{bmatrix}.$$
(23)

Reordering the vector corresponds to a QW propagation operation. It transfers the neighbor information to particle *j*. This is needed for calculating the kernel interaction. Therefore, we need to solve

$$u_j(t+1) = \alpha_{jj}u_j(t) + \sum_n \alpha_{nj}u_n(t) = \vec{u}_j(t) \cdot \vec{\alpha}_j, \qquad (24)$$

where vector $\vec{\alpha}_j$ contains α_{jk} terms. Because this needs to be the same size as $|\vec{u}(t)\rangle$ to calculate an inner product, we let

$$|\vec{\alpha}\rangle = \sum_{j} |\vec{\alpha}_{j}\rangle = \sum_{j,n} \alpha_{nj} |j,n\rangle.$$
 (25)

The inner product is $\vec{u}_j(t) \cdot \vec{\alpha}_j \rightarrow \langle \vec{u}(t) | \vec{\alpha} \rangle$, where it is generally inefficient to prepare the $|\vec{\alpha}\rangle$ state from a classical description. Quantum algorithms typically calculate an inner product as final measurement. It is an irreversible procedure, and therefore, a non-unitary operation. In our previous work,²⁸ we outlined several methods for efficiently calculating the inner product to output a classical value. However, to compute several time steps within the quantum part of the algorithm, we need use a reversible unitary operation. We model this on the "coin operation" from quantum walks. This accomplishes the summation, but also produces extra "junk" terms * that contain the information required to reverse the operation,

$$\vec{u}_{j}(t+1) = \begin{bmatrix} u_{j}(t+1) \\ * \\ * \\ & \\ \vdots \\ \vdots \end{bmatrix}.$$
 (26)

However, the * terms reduce the probability of measuring the correct output. There are several options to reduce or remove the * entries. For example, we can use quantum amplitude amplification^{54,55} to maximize $u_j(t + 1)$ and minimize the * terms, then, discard the qubits carrying the * information. Another option involves the "uncomputation trick" to disentangle the neighbor register.^{56,57} In all cases, we need to use extra qubits and gate operations to proceed to the next time step. In our work, we leave the * terms associated with the old neighbor register, add a new neighbor register, entangle it with the site register, then calculate the next time step. We then post-select for the result we want at the end of the computation. This lowers the overall success probability, but simplifies the presentation of the method in the simple example in Sec. IV.

IV. SIMULATION OF TWO-PARTICLE SYSTEM

In this section, we present a fully worked out example using a simple two-particle model (Fig. 3). We consider two sites labeled "0" and "1." They, respectively, correspond to states u_0 and u_1 . The quantum walker can initially begin at site 0 (1), then, either jump to site 1 (0) or stay at site 0 (1).

For a two-particle system, it is suitable to use a triangular smoothing kernel (Fig. 4) of the form

$$W(r,h) = \begin{cases} 1/h - |r|/h^2, & |r| < h\\ 0, & \text{otherwise,} \end{cases}$$
(27)

$$\frac{dW(r,h)}{dr} = \begin{cases} -\text{sgn}(r)/h^2, & |r| < h\\ 0, & \text{otherwise.} \end{cases}$$
(28)

Given only two particles, this simple kernel structure essentially reproduces a first-order finite difference approximation for the spatial derivative. There is little merit in using higher-order (i.e., smoother, Gaussian-type kernels) kernels over this two-particle system. Any benefits smoother kernels may have on accuracy and stability are only apparent for many-particle systems.

A. Quantum registers

Let the system wavefunction at time *t* be

$$\psi(t) = \beta_{00}(t)|0,0\rangle + \beta_{01}(t)|0,1\rangle + \beta_{10}(t)|1,0\rangle + \beta_{11}(t)|1,1\rangle$$
(29)

for arbitrary amplitudes $\beta_{k\ell}$. The subscripts k and ℓ represent the quantum walker's initial and final position, respectively. The quantum state |position, coin \rangle contains information on the quantum walker position and coin operator.

The "velocity" register contains information on the SPH particle velocity, which we define as



FIG. 3. Example system with two sites (SPH particles) only.



FIG. 4. Sketch of triangular kernel function (with smoothing length h) and its first derivative.

$$|\vec{u}(t)\rangle = u_0(t)|0\rangle + u_1(t)|1\rangle = \mathscr{C}\begin{bmatrix} u_0(t)\\ u_1(t) \end{bmatrix},$$
(30)

with normalization constant \mathscr{C} . The neighbor register

$$|\nu\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),\tag{31}$$

is a normalized state containing all SPH particles inside the region of compact support. (Note that for brevity, we will use the term "velocity register" to refer to the advection quantities u which are solutions to the advection differential equation.) The tensor product gives entangled state

$$\psi(t) = |\vec{u}(t)\rangle \otimes |\nu\rangle,$$
(32)

$$=\frac{\mathscr{C}}{\sqrt{2}}(u_0(t)(|0,0\rangle+|0,1\rangle)+u_1(t)(|1,0\rangle+|1,1\rangle)),$$
 (33)

$$=\frac{\mathscr{C}}{\sqrt{2}}\begin{bmatrix}u_{0}\\u_{0}\\u_{1}\\u_{1}\end{bmatrix},$$
(34)

such that $\beta_{00} = \beta_{01} = \mathcal{C}u_0/\sqrt{2}$ and $\beta_{10} = \beta_{11} = \mathcal{C}u_1/\sqrt{2}$.

B. Shift operation

Applying the shift operator

$$\hat{S} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
(35)

to the entangled state gives

$$\hat{S}\psi(t) = \frac{\mathscr{C}}{\sqrt{2}} \begin{bmatrix} u_0\\ u_1\\ u_0\\ u_1 \end{bmatrix}.$$
(36)

This is analogous to the DTQW shift operation and transfers neighbor information between the SPH particles.

C. Coin

Now use Eq. (24) to define the next time step,

$$\begin{bmatrix} u_0(t+1)\\ u_1(t+1) \end{bmatrix} = \begin{bmatrix} \alpha_{00}u_0(t) + \alpha_{10}u_1(t)\\ \alpha_{11}u_1(t) + \alpha_{01}u_0(t) \end{bmatrix},$$
(37)

$$= \begin{bmatrix} \alpha_{00} & \alpha_{10} \\ \alpha_{01} & \alpha_{11} \end{bmatrix} \begin{bmatrix} u_0(t) \\ u_1(t) \end{bmatrix}.$$
(38)

As described above, we need to include extra "junk" terms and expand the state space so that we can do a "reversible" inner product operation. The matrix would have the structure

$$\begin{bmatrix} u_0(t+1) \\ * \\ u_1(t+1) \end{bmatrix} = \begin{bmatrix} \alpha_{00} & \alpha_{10} & & 0 \\ * & * & 0 \\ 0 & & * & * \\ 0 & & \alpha_{01} & \alpha_{11} \end{bmatrix} \begin{bmatrix} u_0(t) \\ u_1(t) \\ u_0(t) \\ u_1(t) \end{bmatrix}, \quad (39)$$

which contains amplitudes α defined in Eqs. (16) and (17), and unwanted terms *. Since the 2×2 sub-blocks must be unitary, we can let

$$\begin{bmatrix} u_{0}(t+1) \\ * \\ u_{1}(t+1) \end{bmatrix} = \begin{bmatrix} \alpha_{00} & \alpha_{10} & 0 \\ \alpha_{10} & -\alpha_{00} & 0 \\ 0 & -\alpha_{11} & \alpha_{01} \\ 0 & \alpha_{01} & \alpha_{11} \end{bmatrix} \begin{bmatrix} u_{0}(t) \\ u_{1}(t) \\ u_{0}(t) \\ u_{1}(t) \end{bmatrix}, \quad (40)$$
$$= \begin{bmatrix} \alpha_{00}u_{0}(t) + \alpha_{10}u_{1}(t) \\ \alpha_{10}u_{0}(t) - \alpha_{00}u_{1}(t) \\ \alpha_{01}u_{1}(t) - \alpha_{11}u_{0}(t) \\ \alpha_{11}u_{1}(t) + \alpha_{01}u_{0}(t) \end{bmatrix}. \quad (41)$$

This step is analogous to applying the coin operation,

$$1 \otimes \hat{C} = \begin{bmatrix} \hat{H}_0 & 0\\ 0 & \hat{H}_1 \end{bmatrix}, \tag{42}$$

where we define the coins

$$\hat{H}_0 = \begin{bmatrix} \alpha_{00} & \alpha_{10} \\ \alpha_{10} & -\alpha_{00} \end{bmatrix},\tag{43}$$

$$\hat{H}_1 = \begin{bmatrix} -\alpha_{11} & \alpha_{01} \\ \alpha_{01} & \alpha_{11} \end{bmatrix}.$$
(44)

We simplify the coins using Eqs. (16) and (17)

$$\alpha_{11} = \alpha_{00}, \quad \alpha_{10} = \alpha_{01}, \quad \alpha_{10} = 1 - \alpha_{00}.$$
 (45)

Hence,

$$1 \otimes \hat{C} = \begin{bmatrix} \alpha_{00} & 1 - \alpha_{00} & 0 & 0\\ 1 - \alpha_{00} & -\alpha_{00} & 0 & 0\\ 0 & 0 & -\alpha_{00} & 1 - \alpha_{00}\\ 0 & 0 & 1 - \alpha_{00} & \alpha_{00} \end{bmatrix}.$$
 (46)

The coin in its current form needs to be normalized. This gives

$$\hat{U}_C = \mathscr{N}(1 \otimes \hat{C}), \tag{47}$$

where normalization constant $\mathcal{N} = (\alpha_{00}^2 + (1 - \alpha_{00})^2)^{-1/2}$ and $\alpha_{00} = 1 + c\Delta x \nabla_0 W_{01}$. The α_{00} terms provide a mapping to SPH particles.

D. Quantum circuit

Given the initial states $u_0(0)$ and $u_1(0)$, the initial aim is to simulate the evolution across one time step to obtain $u_0(1)$ and $u_1(1)$. We construct a quantum circuit to perform the entanglement, shift, and coin operations (Fig. 5):

① First, we initialize the system. We need one "velocity" register, containing the advection quantity u, and one neighbor register.

Each contain one qubit. Both are in the ground state $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Hadamard gates create an equal superposition of quantum states,



FIG. 5. Circuit schematics for simulating SPH system containing two particles over multiple time steps.

$$|\text{velocity}\rangle = |\text{neighbors}\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1\\1 \end{bmatrix}.$$
 (48)

⁽³⁾ For the velocity register, we encode the initial condition $[u_0(0), u_1(0)]$ using amplitude encoding⁵⁸ such that

$$|\text{velocity}\rangle = \mathscr{C}\begin{bmatrix} u_0(0)\\ u_1(0) \end{bmatrix},$$
 (49)

with normalization constant $\mathscr{C} = (u_0(0)^2 + u_1(0)^2)^{-1/2}$. Since there is one qubit to initialize, we may use a rotation matrix to do the encoding. As a result, we have an entangled state

$$|\text{velocity}\rangle \otimes |\text{neighbors}\rangle = \frac{\mathscr{C}}{\sqrt{2}} \begin{bmatrix} u_0(0) \\ u_0(0) \\ u_1(0) \\ u_1(0) \end{bmatrix}.$$
(50)

④ Next, the swap operation [or shift operator in Eq. (35)] produces the state

$$\frac{\mathscr{C}}{\sqrt{2}} \begin{bmatrix} u_0(0) \\ u_1(0) \\ u_0(0) \\ u_1(0) \end{bmatrix}.$$
 (51)

Physically, it means the particle information is transferred (swapped) between the two particles. This is analogous to setting up the SPH kernel interaction.

(5) The coin operation [Eq. (47)] gives a final state that contains $u_{0,1}(1)$ plus two unwanted terms [Eq. (41)],

$$\frac{\mathscr{CN}}{\sqrt{2}} \begin{bmatrix} \alpha_{00}u_0(0) + (1 - \alpha_{00})u_1(0) \\ (1 - \alpha_{00})u_0(0) - \alpha_{00}u_1(0) \\ (1 - \alpha_{00})u_1(0) - \alpha_{00}u_0(0) \\ \alpha_{00}u_1(0) + (1 - \alpha_{00})u_0(0) \end{bmatrix}.$$
(52)

⑥ To calculate the next time step, we repeat the previous two gates (swap and coin) using the velocity register and a new neighbor register.

Note that if we want to calculate the time evolution over one time step, we omit step (6) and all subsequent gate operations. Instead, we would measure the velocity and neighbor qubit registers immediately after step (5), and extract the solutions $u_{0,1}(1)$. This is shown in Sec. IV E.

E. Results and discussion

We build a circuit using IBM's Qiskit software package (Fig. 6) that performs the calculations in Fig. 5 for one time step. As discussed in Sec. IV D, we use two quantum registers labeled "velocity" and "neighbor" plus two classical bits needed for measuring the final state. Each register contains one qubit in the $|0\rangle$ state. We use Qiskit's state preparation functions to initialize the velocity and neighbor statevectors, where the former contains the initial advection quantities $u_{0,1}(0)$. The CNOT gate creates an entangled state, and the swap gate which transfers neighbor information between the two SPH particles. Next, the unitary coin operator calculates the advection quantities at the next time step $u_{0,1}(1)$. Finally, we measure the velocity and neighbor qubits to obtain the system statevector. In the postprocessing step, we multiply the Qiskit solutions by the normalization constants ($\mathscr{CN}/\sqrt{2}$)⁻¹ [see constant in Eq. (52)].

In Fig. 7, we graphically compare the Qiskit and classical solutions [Eq. (3)] for which this simplified system reduces to

$$u_{0,1}(1) = \alpha_{00}u_{0,1}(0) + (1 - \alpha_{00})u_{1,0}(0).$$
(53)

Each row shows the Qiskit and classical solutions, whereas each column shows the solutions of u_0 and u_1 after one time step. Each graph color represents a different set of initial conditions $u_{0,1}(0)$. We vary the advection speed *c* from 10^{-4} to 2. For small *c*, the solutions



FIG. 6. Simulating two SPH particles over one time step. Qiskit circuit performs entanglement, swap, coin operation, disentanglement, then takes final measurement.

ARTICLE



FIG. 7. Results $u_{0,1}(1)$ after one time step with initial states $u_{0,1}(0)$, particle separation $\Delta x = 0.5$, kernel smoothing length h = 1.2, and varying advection speed c. Top and bottom rows show Qiskit and classical solutions, respectively. Left and right columns show $u_0(1)$ and $u_1(1)$ solutions.

 $u_0(1) \approx u_0(0)$ and $u_1(1) \approx u_0(0)$. As *c* increases, $u_0(1)$ and $u_1(1)$ gradually converge to $(u_0(0) + u_1(0))/2$. For all initial conditions in Fig. 7, $(u_0(0) + u_1(0))/2 = 0.5$. Hence, all graphs converge to 0.5 at some critical advection speed. By setting $u_0(1) = u_1(1)$, we deduce that this critical point occurs at

$$c = -\frac{1}{2} (\Delta t \Delta x \nabla_0 W_{01})^{-1}.$$
 (54)

When $\Delta x < h$, $\nabla_0 W_{01} = -1/h^2$ and we can simplify Eq. (54) to

$$c = \frac{h^2}{2\Delta t \Delta x}, \quad \Delta t = 1.$$
(55)

This crossover is indicated as gray dashed vertical lines in Fig. 7. Given $h \approx \Delta x$, this crossover is approximately half the CFL limit for explicit schemes for the advection equation. Hence, the critical point is consistent with the traveling solution peak $(u_0 + u_1)$ occupying a position halfway between the particles.

Next, we try a different set of parameters (Fig. 8). We vary the particle separation Δx and fix the advection speed *c* and smoothing length *h*. As with the results in Fig. 7, the Qiskit and classical solutions agree to machine precision (within 10^{-16}), so we do not show the classical solutions in Fig. 8 for clarity. Using Eq. (54), we expect a crossover to occur when

$$\Delta x = \frac{-1}{2c\Delta t \nabla_0 W_{01}},\tag{56}$$

shown as gray dashed vertical lines in Fig. 8. Hence, using the kernel property of compact support,

$$\Delta x_{\text{crossover}} = \begin{cases} h^2/(2c\Delta t) & \Delta x < h\\ \text{undefined} & \Delta x \ge h. \end{cases}$$
(57)

When $\Delta x \ge h$, outside the area of compact support, the kernel function has no effect on the SPH particles and $u_{0,1}(0) = u_{0,1}(1)$. It is interesting to note this behavior, even as we explicitly define the neighbor register [Eq. (31)] to contain only the SPH particles inside the area of compact support.

There are some cases where the solutions become negative (Fig. 8). When c = 2 and initial states are $(u_0(0), u_1(0)) = (0.2, 0)$ and $(u_0(0), u_1(0)) = (0, 0.2)$, we see that $u_0(1) < 0$ and $u_1(1) < 0$, respectively, for larger particle spacing values. Negative solutions indicate numerical instabilities. This behavior is unsurprising at larger Δx , and since there are only two SPH particles, the system is likely to be prone to instability in any case. We would have a clearer picture on stability behavior once we consider more particles.

Finally, we present the results after several time steps (Fig. 9). Each graph from left to right shows the solutions after t = 1, 2, and 3 time steps as we vary the advection speed *c* from 10^{-4} to 1. Qiskit and classical results are in good agreement.

There are unwanted terms in the statevector after simulating one time step [Eq. (52)]. These have a significant effect on the solution accuracy for the t = 2, 3 results. We need to remove the unwanted terms after measurement, in the post-processing stage (see the Appendix). Calculating $u_{0,1}(2)$ and $u_{0,1}(3)$ involve taking linear combinations of the statevector elements. This becomes non-trivial when the circuit contains many qubits or if we want to calculate many time steps. In this case, it will be useful to explore the techniques described

Physics of Fluids





FIG. 8. Qiskit results $u_{0,1}(1)$ after one time step with initial states $u_{0,1}(0)$, kernel smoothing length h = 1.4, and varying particle separation Δx . Rows show different advection speeds *c*. Left and right columns show $u_0(1)$ and $u_1(1)$ solutions.



FIG. 9. Results $u_{0,1}(t)$ after t = 1, 2, 3 time steps with initial states ($u_0(0), u_1(0)$) = (0.8, 0.4), particle separation $\Delta x = 0.2$, kernel smoothing length h = 1.2, and varying advection speed *c*. Blue and orange graphs show Qiskit and classical solutions, respectively.

at the end of Sec. III, such as quantum amplitude amplification or the "uncomputation trick." It is essential to address this issue before trying to simulate more time steps.

V. CONCLUSIONS AND FUTURE WORK

We develop a quantum algorithm based on QW operations as a framework to solve the advection equation in SPH form. We provide a fully worked out example using a simple two-particle system. Then we build a quantum circuit using Qiskit software to perform the simulation over three time steps. We present several graphs showing the results as we vary the initial advection quantity, advection speed, and SPH particle separation. There is excellent agreement with results from classical calculations. However, due to unwanted terms in the statevector after simulating one time step, the probability of measuring the solution can deteriorate for further time steps. This is addressed by removing the unwanted terms after measurement in the postprocessing stage.

Our progress sets up numerous future research avenues. For example, we could generalize the procedures in this paper to analyze the advection and diffusion equations containing many SPH particles over many time steps. Hence, leading to the application end goal of solving the Navier–Stokes equation.

A. Freely moving particles

We used quantum walk principles to develop a time step method that is already more general than a basic quantum walk. The next step is to move the particles off grid. One of the strengths of SPH over mesh-based methods is its freely moving SPH particles. Including these extra degrees of freedom would align the quantum algorithm closer to classical SPH formalism. This would go beyond quantum walks by introducing a third register for the particle positions which also needs to be updated appropriately. However, the same basic methods for calculating the updates to each vector can continue to be adapted to the new degrees of freedom.

To calculate the SPH kernel function, we need to determine the particles within the radius of the smoothing length (inside the interaction range). Numerous methods have been developed to efficiently create a list of neighbors, such as the cell-linked list and Verlet list schemes.⁵⁹ In the quantum domain, Grover's search algorithm⁶⁰ provides quadratic speedup over classical search methods. Combining Grover search with existing SPH neighbor-list approaches could offer a novel and effective search method.

B. More particles and time steps

We can generalize the two-particle example by considering several SPH particles organized into simple geometries. Examples include particles on a line (one-dimensional system)²⁸ or in a circular arrangement.⁶¹ We note that discrete-time quantum walks on cycle graphs^{47,61} have a convenient encoding when executed on digital quantum computers. We can also consider a system of disordered particles with a constrained quantum walk lattice. In any case, this will require more qubits.

To evolve the system over many time steps, we need more qubits and gates, hence increasing the quantum circuit depth. As mentioned above, one options is to perform quantum amplitude amplification. The aim is to discard the unwanted terms in the statevector. Hence, after performing the coin operation (step in Fig. 5), amplitude amplification would minimize the β_{01} and β_{10} terms in Eq. (52), while maximizing the β_{00} and β_{11} terms. Quantum algorithms that simulate more SPH particles and time steps require deep quantum circuits composed of highly accurate quantum gates. We do not expect our algorithm to be viable for noisy intermediate-scale quantum (NISQ) computers.¹³ Rather, we anticipate our future QSPH algorithms to be run on fully error-corrected, fault-tolerant quantum hardware. In such hardware, error correction^{62,63} and error mitigation techniques⁶⁴ are applied below the level of the algorithm so they are not application-specific, and we can assume we have almost perfect hardware to run our algorithm.

C. Resource requirements

Although it is important to consider real-world applications containing potentially millions of SPH particles evolving over many time steps, our focus in this work is on the time stepping. At this stage, it is premature to do large-scale resource estimates before further algorithm testing and refinement.

The SPH method does not require reading out the full information of all SPH particles. Rather, we can use the already smoothed/ interpolated function values for a subset of particles to provide a decent representation of the fluid. Since we expect to measure the quantum circuit at regular intervals, the number of time steps that we run before measurement and reinitialization will be roughly constant, even as the problem size increases.

At the start of the calculation, we encode the initial state into the velocity register (Fig. 5). For example, quantum amplitude encoding stores the information in the amplitudes of the quantum state. The qubit number scales logarithmically with the input vector length, whereas gate count scales exponentially with the input vector length. This is true when each amplitude is different. If adjacent amplitudes are related because we have a smoothed function, this can be mitigated to generate the SPH particles that represent the function.⁶⁵

Finally, we reiterate that the quantum walk formalism has given us the mechanism for calculating a pair-wise particle interaction. This is the fundamental unit of operation on which all SPH methods are built. The underpinning quantum algorithm is valid, regardless of whether the flow is turbulent or multiphase.

ACKNOWLEDGMENTS

R.A., S.J.L., and V.M.K. are supported by UK Research and Innovation (UKRI) Grant Nos. EP/Y004566/1 (R.A. and V.M.K.) and EP/Y004663/2 (S.J.L.) (QuANDIE) and EP/Z53318X/1 (QCI3). R.A. and V.M.K. are supported by EP/W00772X/2 (QEVEC) and EP/T001062/1 (QCS Hub). V.M.K. is supported by EP/T026715/2 (CCP-QC). For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author contributions

R. Au-Yeung: Conceptualization, data curation, formal analysis, funding acquisition, investigation, methodology, project administration, software, supervision, validation, visualization,

pubs.aip.org/aip/pof

writing – original draft, writing – review & editing. V. M. Kendon: Conceptualization, formal analysis, funding acquisition, methodology, project administration, resources, supervision, writing – original draft, writing – review & editing. S. J. Lind: Conceptualization, formal analysis, funding acquisition, methodology, project administration, supervision, writing – original draft, writing – review & editing.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

APPENDIX: POSTPROCESSING QISKIT OUTPUTS AFTER TWO TIME STEPS

In this section, we calculate the advection solution at t = 2. We start with the advection solution at t = 1 [Eq. (52), or between step (5) and (6) in Fig. 5]. At this point, the statevector for the two-qubit system is

$$\begin{bmatrix} \psi_{0} \\ \psi_{1} \\ \psi_{2} \\ \psi_{3} \end{bmatrix} = \frac{\mathscr{CN}}{\sqrt{2}} \begin{bmatrix} \alpha_{00}u_{0}(0) + (1 - \alpha_{00})u_{1}(0) \\ (1 - \alpha_{00})u_{0}(0) - \alpha_{00}u_{1}(0) \\ (1 - \alpha_{00})u_{1}(0) - \alpha_{00}u_{0}(0) \\ \alpha_{00}u_{1}(0) + (1 - \alpha_{00})u_{0}(0) \end{bmatrix}.$$
 (A1)

Alternatively, the three-qubit system containing the velocity register qubit and two neighbor register qubits can be expressed as entangled state,

$$\begin{bmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \psi_0 \\ \psi_0 \\ \psi_1 \\ \psi_1 \\ \psi_2 \\ \psi_2 \\ \psi_2 \\ \psi_3 \\ \psi_3 \end{bmatrix}.$$
(A2)

Next, applying the swap gate \hat{S} (step (5) in Fig. 5) gives the statevector

Note this gate only affects the velocity register qubit and the t = 2 neighbor register qubit.

Then, the coin operator, which again only affects the velocity and t = 2 neighbor qubits, gives us the solutions for t = 2. To construct this coin operator, recall how the coin operates on the base states [Eq. (46)]

$$\hat{U}_C |00\rangle = \alpha_{00} |00\rangle + (1 - \alpha_{00}) |01\rangle,$$
 (A4)

$$\hat{U}_C |01\rangle = (1 - \alpha_{00})|00\rangle - \alpha_{00}|01\rangle,$$
 (A5)

$$U_C |10\rangle = (1 - \alpha_{00})|11\rangle - \alpha_{00}|10\rangle,$$
 (A6)

$$U_C|11\rangle = \alpha_{00}|11\rangle + (1 - \alpha_{00})|10\rangle,$$
 (A7)

modulo some normalization factor. For the three-qubit circuit, this becomes

$$\hat{U}_{C}|0k0\rangle = \alpha_{00}|0k0\rangle + (1 - \alpha_{00})|0k1\rangle,$$
 (A8)

$$\hat{U}_C |0k1\rangle = (1 - \alpha_{00})|0k0\rangle - \alpha_{00}|0k1\rangle,$$
 (A9)

$$\hat{U}_C |1k0\rangle = (1 - \alpha_{00})|1k1\rangle - \alpha_{00}|1k0\rangle,$$
 (A10)

$$\hat{U}_C |1k1\rangle = \alpha_{00} |1k1\rangle + (1 - \alpha_{00}) |1k0\rangle,$$
 (A11)

for $k \in \{0, 1\}$. The coin operator only affects the first and third qubits corresponding to the velocity and t = 2 neighbor qubits, respectively. The second qubit denoted as *k* is not changed.

To build the matrix, we let

$\hat{U}_C =$	$\langle 000 \hat{U}_{C} 000 angle$	$\langle 000 \hat{{U}}_{C} 001 angle$	$\langle 000 \hat{U}_C 010 angle$	$\langle 000 \hat{U}_C 011 angle$	$\langle 000 {\hat U}_C 100 angle$	$\langle 000 \hat{{U}}_{C} 101 angle$	$\langle 000 \hat{\boldsymbol{U}}_{C} 110 \rangle$	$\langle 000 \hat{U}_C 111 \rangle$	
	$\langle 001 {\hat U}_C 000 angle$	$\langle 001 \hat{{U}}_{C} 001 angle$	$\langle 001 \hat{U}_C 010 angle$	$\langle 001 \hat{U}_C 011 angle$	$\langle 001 \hat{{U}}_{C} 100 angle$	$\langle 001 \hat{U}_C 101 angle$	$\langle 001 \hat{\boldsymbol{U}}_{C} 110 \rangle$	$\langle 001 {\hat U}_C 111 angle$	
	$\langle 010 {\hat U}_C 000 angle$	$\langle 010 \hat{{U}}_{C} 001 angle$	$\langle 010 \hat{U}_C 010 angle$	$\langle 010 \hat{{U}}_{C} 011 angle$	$\langle 010 {\hat U}_C 100 angle$	$\langle 010 \hat{U}_C 101 angle$	$\langle 010 \hat{U}_C 110 \rangle$	$\langle 010 {\hat U}_C 111 angle$	
	$\langle 011 {\hat U}_C 000 angle$	$\langle 011 \hat{U}_C 001 angle$	$\langle 011 \hat{U}_C 010 angle$	$\langle 011 \hat{U}_C 011 angle$	$\langle 011 \hat{{U}}_{C} 100 angle$	$\langle 011 \hat{U}_C 101 angle$	$\langle 011 \hat{U}_C 110 angle$	$\langle 011 \hat{U}_C 111 angle$	
	$\langle 100 {\hat U}_C 000 angle$	$\langle 100 \hat{{U}}_{C} 001 angle$	$\langle 100 \hat{U}_C 010 angle$	$\langle 100 \hat{U}_C 011 angle$	$\langle 100 {\hat U}_C 100 angle$	$\langle 100 \hat{U}_C 101 angle$	$\langle 100 \hat{U}_C 110 \rangle$	$\langle 100 \hat{U}_C 111 angle$,
	$\langle 101 {\hat U}_C 000 angle$	$\langle 101 \hat{U}_C 001 angle$	$\langle 101 \hat{U}_C 010 angle$	$\langle 101 \hat{U}_C 011 angle$	$\langle 101 \hat{U}_C 100 angle$	$\langle 101 \hat{U}_C 101 \rangle$	$\langle 101 \hat{\boldsymbol{U}}_{C} 110 \rangle$	$\langle 101 \hat{U}_C 111 \rangle$	
	$\langle 110 \hat{U}_C 000 angle$	$\langle 110 \hat{U}_C 001 angle$	$\langle 110 \hat{\boldsymbol{U}}_{C} 010 \rangle$	$\langle 110 \hat{U}_C 011 \rangle$	$\langle 110 \hat{U}_C 100 angle$	$\langle 110 \hat{U}_C 101 \rangle$	$\langle 110 \hat{U}_C 110 \rangle$	$\langle 110 \hat{U}_C 111 \rangle$	
	$\langle 111 \hat{U}_C 000 \rangle$	$\langle 111 \hat{\boldsymbol{U}}_{C} 001 \rangle$	$\langle 111 \hat{\boldsymbol{U}}_{C} 010 \rangle$	$\langle 111 \hat{\boldsymbol{U}}_{C} 011 \rangle$	$\langle 111 \hat{\boldsymbol{U}}_{C} 100 \rangle$	$\langle 111 \hat{\boldsymbol{U}}_{C} 101 \rangle$	$\langle 111 \hat{\boldsymbol{U}}_{C} 110 \rangle$	$\langle 111 \hat{U}_C 111\rangle$	
								(A12	2)

(A13)

	α_{00}	$1 - \alpha_{00}$	0	0	0	0	0	0
	$1 - \alpha_{00}$	$-\alpha_{00}$	0	0	0	0	0	0
	0	0	α_{00}	$1-lpha_{00}$	0	0	0	0
_ N	0	0	$1 - \alpha_{00}$	$-\alpha_{00}$	0	0	0	0
- 21	0	0	0	0	$-\alpha_{00}$	$1-lpha_{00}$	0	0
	0	0	0	0	$1-lpha_{00}$	α_{00}	0	0
	0	0	0	0	0	0	$-\alpha_{00}$	$1 - \alpha_0$
	0	0	0	0	0	0	$1 - \alpha_{00}$	α_{00}

Applying this coin then gives the statevector

$$\hat{U}_{C}\begin{bmatrix}\psi_{0}\\\psi_{2}\\\psi_{1}\\\psi_{3}\\\psi_{0}\\\psi_{2}\\\psi_{1}\\\psi_{3}\\\psi_{3}\end{bmatrix} = \frac{\mathcal{N}}{\sqrt{2}}\begin{bmatrix}\alpha_{00}\psi_{0} + (1-\alpha_{00})\psi_{2}\\(1-\alpha_{00})\psi_{0} - \alpha_{00}\psi_{2}\\\alpha_{00}\psi_{1} + (1-\alpha_{00})\psi_{3}\\(1-\alpha_{00})\psi_{1} - \alpha_{00}\psi_{3}\\(1-\alpha_{00})\psi_{0} + \alpha_{00}\psi_{2}\\(1-\alpha_{00})\psi_{0} + \alpha_{00}\psi_{2}\\(1-\alpha_{00})\psi_{1} + (1-\alpha_{00})\psi_{3}\\(1-\alpha_{00})\psi_{1} + \alpha_{00}\psi_{3}\end{bmatrix} = \begin{bmatrix}\chi_{0}\\\chi_{1}\\\chi_{2}\\\chi_{3}\\\chi_{4}\\\chi_{5}\\\chi_{6}\\\chi_{7}\end{bmatrix}.$$
(A14)

The classical solution [Eq. (3)] at t = 2 is

$$u_{0,1}(2) = \alpha_{00} u_{0,1}(1) + (1 - \alpha_{00}) u_{1,0}(1),$$
 (A15)

or, using Eq. (A1),

$$u_{0,1}(2) = \frac{\mathscr{N}}{\sqrt{2}} (\alpha_{00}\psi_{0,3} + (1 - \alpha_{00})\psi_{3,0}). \tag{A16}$$

Using Eqs. (A1) and (A14), we can take linear combinations of χ_j elements to build $u_{0,1}(2)$

$$\frac{1}{2}(\chi_0 - \chi_4) = \alpha_{00} u_0(1), \tag{A17}$$

$$\frac{1}{2}(\chi_1 + \chi_5) = (1 - \alpha_{00})u_0(1), \tag{A18}$$

$$\frac{1}{2}(\chi_2 + \chi_6) = (1 - \alpha_{00})u_1(1), \tag{A19}$$

$$\frac{1}{2}(\chi_7 - \chi_3) = \alpha_{00}u_1(1).$$
 (A20)

Hence,

$$u_0(2) = \frac{\mathscr{N}^2 \mathscr{C}}{2} \left(\frac{1}{2} (\chi_0 - \chi_4) + \frac{1}{2} (\chi_2 + \chi_6) \right), \qquad (A21)$$

$$u_1(2) = \frac{\mathscr{N}^2 \mathscr{C}}{2} \left(\frac{1}{2} (\chi_7 - \chi_3) + \frac{1}{2} (\chi_1 + \chi_5) \right), \qquad (A22)$$

where the $\mathcal{N}^2 \mathcal{C}/2$ constants arise from normalizing the initial "velocity" register statevector (\mathcal{C}), normalizing two "neighbor" registers $(1/\sqrt{2})^2$, and applying two coin operations (\mathcal{N}^2).

We use similar procedures when post-processing results for the next time step, $u_{0,1}(3)$.

REFERENCES

- ¹F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," Nature 574, 505–510 (2019).
 ²Y. Wu *et al.*, "Strong quantum computational advantage using a superconduct-
- ²Y. Wu *et al.*, "Strong quantum computational advantage using a superconducting quantum processor," Phys. Rev. Lett. **127**, 180501 (2021).
- ³L. S. Madsen, F. Laudenbach, M. F. Askarani, F. Rortais, T. Vincent, J. F. F. Bulmer, F. M. Miatto, L. Neuhaus, L. G. Helt, M. J. Collins, A. E. Lita, T. Gerrits, S. W. Nam, V. D. Vaidya, M. Menotti, I. Dhand, Z. Vernon, N. Quesada, and J. Lavoie, "Quantum computational advantage with a programmable photonic processor," Nature **606**, 75–81 (2022).
- ⁴X. Li, X. Yin, N. Wiebe, J. Chun, G. K. Schenter, M. S. Cheung, and J. Mülmenstädt, "Potential quantum advantage for simulation of fluid dynamics," Phys. Rev. Res. 7, 013036 (2025).
- ⁵F. Gaitan, "Finding solutions of the Navier-Stokes equations through quantum computing-recent progress, a generalization, and next steps forward," Adv. Quantum Technol. 4, 2100055 (2021).
- ⁶A. M. Dalzell, S. McArdle, M. Berta, P. Bienias, C.-F. Chen, A. Gilyén, C. T. Hann, M. J. Kastoryano, E. T. Khabiboulline, A. Kubica, G. Salton, S. Wang, and F. G. S. L. Brandão, *Quantum Algorithms: A Survey of Applications and End-to-End Complexities* (Cambridge University Press, Cambridge, 2025).
- ⁷S. S. Bharadwaj and K. R. Sreenivasan, "Hybrid quantum algorithms for flow problems," Proc. Natl. Acad. Sci. U.S.A. 120, e2311014120 (2023).
- ⁸D. Jaksch, P. Givi, A. J. Daley, and T. Rung, "Variational quantum algorithms for computational fluid dynamics," AIAA J. **61**, 1885–1894 (2023).
- ⁹S. Succi, W. Itani, C. Sanavio, K. R. Sreenivasan, and R. Steijl, "Ensemble fluid simulations on quantum computers," Comput. Fluids **270**, 106148 (2024).
- ¹⁰A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," Phys. Rev. Lett. **103**, 150502 (2009).
- ¹¹F. Oz, R. K. S. S. Vuppala, K. Kara, and F. Gaitan, "Solving Burgers' equation with quantum computing," <u>Quantum Inf. Process.</u> 21, 30 (2021).
- ¹²L. Lapworth, "A hybrid quantum-classical CFD methodology with benchmark HHL solutions," arXiv:2206.00419 (2022).
- ¹³J. Preskill, "Quantum computing in the NISQ era and beyond," Quantum 2, 79 (2018).
- ¹⁴K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C.

Kwek, and A. Aspuru-Guzik, "Noisy intermediate-scale quantum algorithms," Rev. Mod. Phys. 94, 015004 (2022).

- 15 S. Succi, W. Itani, K. R. Sreenivasan, and R. Steijl, "Quantum computing for fluids: Where do we stand?" Europhys. Lett. **144**, 10001 (2023). ¹⁶S. Aaronson, "Read the fine print," Nat. Phys. **11**, 291–293 (2015).
- ¹⁷Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala, "Evidence for the utility of quantum computing before fault tolerance," Nature 618, 500-505 (2023).
- ¹⁸I. M. Georgescu, S. Ashhab, and F. Nori, "Quantum simulation," Rev. Mod. Phys. 86, 1 (2014).
- ¹⁹S. Succi, F. Fillion-Gourdeau, and S. Palpacelli, "Quantum lattice Boltzmann is a quantum walk," EPJ Quantum Technol. 2, 12 (2015).
- ²⁰W. Itani, K. B. Sreenivasan, and S. Succi, "Quantum algorithm for lattice Boltzmann (QALB) simulation of incompressible fluids with a nonlinear collision term," Phys. Fluids 36, 017112 (2024).
- ²¹C. Sanavio and S. Succi, "Lattice Boltzmann-Carleman quantum algorithm and circuit for fluid flows at moderate Reynolds number," AVS Quantum Sci. 6, 023802 (2024).
- 22 L. Budinski, "Quantum algorithm for the advection-diffusion equation simulated with the lattice Boltzmann method," Ouantum Inf. Process. 20, 57 (2021).
- 23Y. Aharonov, L. Davidovich, and N. Zagury, "Quantum random walks," Phys. Rev. A 48, 1687-1690 (1993).
- ²⁴E. Farhi and S. Gutmann, "Quantum computation and decision trees," Phys. Rev. A 58, 915-928 (1998).
- ²⁵A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, "Exponential algorithmic speedup by a quantum walk," in Proceedings of the 35th ACM STOC (ACM, New York, NY, 2003), pp. 59-68.
- ²⁶L. B. Lucy, "A numerical approach to the testing of the fission hypothesis," Astron. J. 82, 1013-1024 (1977).
- 27 R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics: Theory and application to non-spherical stars," Mon. Not. R Astron. Soc. 181, 375-389 (1977).
- 28 R. Au-Yeung, A. J. Williams, V. M. Kendon, and S. J. Lind, "Quantum algorithm for smoothed particle hydrodynamics," Comput. Phys. Commun. 294, 108909 (2024).
- 29S. S. Bharadwaj and K. R. Sreenivasan, "An introduction to algorithms in quantum" computation of fluid dynamics," in Introduction to Quantum Computation of Fluid Dynamics (NATO Science & Technology Organization, 2023).
- ³⁰S. S. Bharadwaj and K. R. Sreenivasan, "Quantum computation of fluid dynamics," Indian Acad. Sci. Conf. Ser. 3, 77-96 (2020).
- ³¹P. Givi, A. J. Daley, D. Mavriplis, and M. Malik, "Quantum speedup for aeroscience and engineering," AIAA J. 58, 3715 (2020).
- 32 M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition (Cambridge University Press, 2010).
- ³³J. J. Monaghan, "Smoothed particle hydrodynamics and its diverse applications," Annu. Rev. Fluid Mech. 44, 323-346 (2012).
- ³⁴D. Koschier, J. Bender, B. Solenthaler, and M. Teschner, "A survey on SPH methods in computer graphics," Comput. Graph. Forum 41, 737-760 (2022).
- ³⁵J. J. Monaghan, "Smoothed particle hydrodynamics," Rep. Prog. Phys. 68, 1703-1759 (2005).
- ³⁶S. J. Lind, B. D. Rogers, and P. K. Stansby, "Review of smoothed particle hydrodynamics: Towards converged Lagrangian flow modelling," Proc. R. Soc. A 476, 20190801 (2020).
- 37 Abaqus, "Smoothed particle hydrodynamics," https://abaqus-docs.mit.edu/ 2017/English/SIMACAEANLRefMap/simaanl-c-sphanalysis.htm (2017) (last accessed: January 01, 2024).
- 38S. J. Lind and P. K. Stansby, "High-order Eulerian incompressible smoothed particle hydrodynamics with transition to Lagrangian free-surface motion," Comput. Phys. 326, 290-311 (2016).
- ³⁹A. M. A. Nasar, B. D. Rogers, A. Revell, P. K. Stansby, and S. J. Lind, "Eulerian weakly compressible smoothed particle hydrodynamics (SPH) with the immersed boundary method for thin slender bodies," J. Fluid. Struct. 84, 263-282 (2019).

- 40 A. M. A. Nasar, G. Fourtakas, S. J. Lind, J. R. C. King, B. D. Rogers, and P. K. Stansby, "High-order consistent SPH with the pressure projection method in 2-D and 3-D," J. Comput. Phys. 444, 110563 (2021).
- ⁴¹A. Barenco, A. Berthiaume, D. Deutsch, A. Ekert, R. Jozsa, and C. Macchiavello, "Stabilization of quantum computations by symmetrization," SIAM J. Comput. 26, 1541-1557 (1997).
- ⁴²H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, "Quantum fingerprinting," Phys. Rev. Lett. 87, 167902 (2001).
- ⁴³G.-L. Long and Y. Sun, "Efficient scheme for initializing a quantum register with an arbitrary superposed state," Phys. Rev. A 64, 014303 (2001).
- 44M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Transformation of quantum states using uniformly controlled rotations," Quantum Inf. Comput. 5, 467-473 (2005).
- ⁴⁵M. Fanizza, M. Rosati, M. Skotiniotis, J. Calsamiglia, and V. Giovannetti, "Beyond the swap test: Optimal estimation of quantum state overlap," Phys. Rev. Lett. 124, 060503 (2020).
- ⁴⁶A. Ambainis, "Quantum walks and their algorithmic applications," Int. J. Quantum Inf. 1, 507-518 (2003).
- 47S. E. Venegas-Andraca, "Quantum walks: A comprehensive review," Quantum Inf. Process. 11, 1015–1106 (2012).
- 48K. Kadian, S. Garhwal, and A. Kumar, "Quantum walk and its application domains: A systematic review," Comput. Sci. Rev. 41, 100419 (2021).
- ⁴⁹B. Claudon, J.-P. Piquemal, and P. Monmarché, "Quantum speedup for nonreversible Markov chains," arXiv:2501.05868 (2025).
- 50 R. Steijl and G. N. Barakos, "Parallel evaluation of quantum algorithms for computational fluid dynamics," Comput. Fluids 173, 22-28 (2018).
- ⁵¹J. Zylberman, G. D. Molfetta, M. Brachet, N. F. Loureiro, and F. Debbasch, "Quantum simulations of hydrodynamics via the Madelung transformation," Phys. Rev. A 106, 032408 (2022).
- ⁵²A. M. Childs, "Universal computation by quantum walk," Phys. Rev. Lett. 102, 180501 (2009).
- 53A. M. Childs, "On the relationship between continuous- and discrete-time quantum walk," Commun. Math. Phys. 294, 581-603 (2010).
- ⁵⁴G. Brassard and P. Høyer, "An exact quantum polynomial-time algorithm for Simon's problem," in Proceedings of the 5th ISTCS (IEEE, 1997), pp. 12-23.
- ⁵⁵L. K. Grover, "Quantum computers can search rapidly by using almost any transformation," Phys. Rev. Lett. 80, 4329 (1998).
- ⁵⁶C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," SIAM J. Comput. 26, 1510-1523 (1997).
- 57 R. Cleve, W. van Dam, M. Nielsen, and A. Tapp, "Quantum entanglement and the communication complexity of the inner product function," Theor. Comput. Sci. 486, 11-19 (2013).
- 58 J. Gonzalez-Conde, T. W. Watts, P. Rodriguez-Grasa, and M. Sanz, "Efficient quantum amplitude encoding of polynomial functions," Quantum 8, 1297 (2024).
- ⁵⁹J. M. Domínguez Alonso, "DualSPHysics: Towards high performance computing using SPH technique," Ph.D. thesis (Universidad de Vigo, Galicia, Spain, 2014).
- 60 L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proceedings of the 28th ACM SOTC (ACM, New York, NY, 1996), pp. 212-219.
- ⁶¹V. Wadhia, N. Chancellor, and V. Kendon, "Cycle discrete-time quantum walks on a noisy quantum computer," Eur. Phys. J. D 78, 29 (2024).
- ⁶²J. Roffe, "Quantum error correction: An introductory guide," Contemp. Phys. 60, 226-245 (2019).
- 63 B. Terhal, "Quantum error correction for quantum memories," Rev. Mod. Phys. 87, 307 (2015).
- ⁶⁴Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, and T. E. O'Brien, "Quantum error mitigation," Rev. Mod. Phys. 95, 045005 (2023)
- 65 F. Hayes, S. Croke, C. Messenger, and F. Speirits, "Quantum state preparation of gravitational waves," arXiv:2306.11073 (2023).