Quantum algorithm for nonlinear Burgers' equation for high-speed compressible flows

Esmaeil Esmaeilifar (اسماعیل اسماعیلی فر)[a], Doyeol Ahn (안도열)[b], Rho Shin Myong (명노신)[a,*]

[a]School of Mechanical and Aerospace Engineering, and ACTRC, Gyeongsang National University, 501 Jinjudaero, Jinju, Gyeongnam 52828, South Korea

[b]Department of Electrical and Computer Engineering, University of Seoul, 163 Seoulsiripdae-Ro, Tongdaimoon-Gu, Seoul 02504, South Korea

*Corresponding author. E-mail address: *myong@gnu.ac.kr* (R.S. Myong)

Abstract

Recent advances in quantum hardware and quantum computing algorithms promise significant breakthroughs in computational capabilities. Quantum computers can achieve exponential improvements in speed versus classical computers by employing principles of quantum mechanics like superposition and entanglement. However, designing quantum algorithms to solve the nonlinear partial differential equations governing fluid dynamics is challenging due to the inherent linearity of quantum mechanics, which requires unitary transformation. In this study, we first address in detail several challenges that arise when trying to deal with nonlinearity using quantum algorithms and then propose a novel pure quantum algorithm for solving a nonlinear Burgers' equation. We employed multiple copies of the state vector to calculate the nonlinear term, which is necessary due to the no-cloning theorem. By reusing qubits from the previous time steps, we significantly reduced the number of qubits required for multi-step simulations, from exponential/quadratic scaling in earlier studies to linear scaling in time in the current study. We also employed various advanced quantum techniques, including block-encoding, quantum Hadamard product, and the linear combination of unitaries, to design a quantum circuit for the proposed quantum algorithm. The quantum circuit was executed on quantum simulators, and the obtained results demonstrated excellent agreement with those from classical simulations.

Keywords: Quantum computing; nonlinear partial differential equation; Burgers' equation; computational fluid dynamics

1. Introduction

Computational fluid dynamics (CFD) is a crucial tool for simulating fluid flows in a diverse range of scientific and industrial fields. However, in applications ranging from aerospace engineering to weather predictions, CFD simulations are highly demanding of computational resources, since they require multiscale simulations. For instance, the computational cost of a turbulent flow simulation grows proportionally to Re^3, and to perform direct numerical simulations on a full airplane requires huge exascale supercomputers [1, 2]. On the other hand, quantum computers can achieve an exponentially improved speed compared to classical computers, by employing the inherent parallelism in quantum mechanics and overcoming the classical limitations of dimensionality. While utilizing quantum computers to solve fluid dynamics can be a significant advance, designing efficient quantum algorithms (QA) for CFD is highly challenging, given the constraints imposed by the postulates of quantum mechanics, such as unitary transformations and the no-cloning theorem. It is not surprising that most studies regarding quantum algorithms for CFD are limited to solving linear partial differential equations, such as the heat equation [3, 4], Lattice Boltzmann Method (LBM) [5-9], collisionless Boltzmann equation [10], Poisson equation [11-13], linear advection-diffusion equations [14, 15], while for nonlinear partial differential equations hybrid classical-quantum approaches [16-20] are used.

The potential exponential speed-up of quantum computers compared to the classical counterpart is based on the features of quantum superposition, entanglement, and interference. While classical computers use bits as the basic unit of information, quantum computers use qubits. The main difference is that classical bits can be either in state zero or one, while qubits can be in a state in superposition of their basis states. Based on the postulates of quantum computing [21], the state of any quantum system can be described by a state vector, a complex unit vector in Hilbert space. Any change in the state of the quantum system is described by unitary transformations, which are linear maps that preserve the norm. Although the state of a quantum system is described by a superposition of its basis states, it collapses to one of its basis states once it is measured. In other words, the realization of the state vector requires multiple measurements to retrieve the amplitudes of the state vector based on the probability of measurement outcomes. Finally, the state space of a multi-qubit system is the tensor product of state spaces.

While in classical computers, information is encoded according to its computational basis, in quantum computers, the information can be encoded in amplitudes of state vectors as well as the computational basis. For an *n*-qubit system, there are N=2^n basis states, and the information can be encoded using the amplitudes of the state vector. For instance, if we have a classical vector with *N* elements, we only need $n = \log N$ qubits to encode the information in the amplitudes of the state vector as $\left|\psi\right\rangle = \sum_{i=1}^{N} \alpha_i \left|i\right\rangle$ with $\sum_{i=1}^{N} \alpha_i^2 = 1$, where α_i is the amplitude and α_i^2 is the probability of collapsing the vector state to the basis state of $\left|i\right\rangle$. The amplitudes are complex numbers which are closely related to the probabilities.

Despite the potential benefits of quantum algorithms over classical ones, implementing efficient quantum algorithms for scientific calculations, especially nonlinear partial differential equations like Navier-Stokes can be challenging because of the constraints imposed by the postulates of quantum mechanics.

First, any operation on the state vector must be a unitary transformation, which is not the case for most scientific calculations. Second, the no-cloning theorem prevents duplicating the state vector. In classical computing, we always use copies of vectors to calculate terms like u^2 or $u\,\partial u/\partial x$, but this is not directly possible in quantum computing. This leads to another difficulty when calculating scientific functions like sine, cosine, exponentials, and logarithms, which are repeatedly used in scientific calculations.

Third, retrieving information from the state vector is performed probabilistically, requiring multiple measurements to construct the classical data. This can be challenging when we need to post-select the measurements to retrieve the information hidden in the desired state from the whole state vector. Since getting the desired results is probabilistic, we may need to take an enormous number of measurements to obtain the results with acceptable accuracy.

Fourth, in CFD algorithms, we usually need to march in time, which requires multiple iterations. However, we do not have classical access to the state vector when executing the quantum algorithm. As a result, implementing a time-dependent quantum operator is extremely challenging. The naïve approach of measuring the state vector, updating the operator based on the classical data, and reinitializing the quantum circuit increases the computational cost and destroys the quantum advantage.

Finally, even if we could design an efficient quantum algorithm that is faster than the classical algorithm, an inefficient initialization of the quantum state vector can destroy the quantum benefit. In summary, designing an efficient quantum algorithm for fluid dynamics requires efficient initialization, a quantum algorithm for handling non-unitary and nonlinear transformations, a time marching strategy for multiple iterations, and post-selecting the desired state to obtain the classical results.

In addition to the challenges of developing quantum algorithms, quantum computer hardware limitations are also obstacles to conducting real simulations with quantum computers. Today's quantum computers are still relatively noisy intermediate-scale quantum (NISQ) devices, characterized by their limited qubit numbers and high error rates. Effective quantum error correction (QEC) is essential for these systems to perform reliably, but that requires a vast number of physical qubits to encode a smaller number of logical qubits [22]. This is because QEC protocols must handle various error types and magnitudes, necessitating redundancy. Additionally, optimizing quantum circuits to reduce the number of elemental gates can significantly enhance performance by lowering error rates and improving computational efficiency [23, 24]. This dual approach of robust error correction and circuit optimization is vital for the future scalability and practical implementation of quantum computing technologies.

Quantum computing is an active research area, and researchers from various fields are attempting to develop quantum algorithms to achieve the speed advantage promised by quantum computers. Quantum computing is based on the postulates of quantum mechanics, which makes it highly aligned with the simulation of quantum physics.

Shor’s algorithm [25], which can find the prime factors of an integer exponentially faster than any classical algorithm, was a breakthrough that promised scientific calculations in areas other than quantum mechanics could benefit from quantum computers. More advanced developments in quantum computing algorithms like quantum phase estimation (QPE) [26], quantum Fourier transform (QFT) [27], and quantum amplitude amplification (QAA) [28] have opened horizons for finding new applications for solving scientific and engineering problems. For instance, the HHL algorithm [29], which was developed based on QFT and QPE, solves linear systems of equations exponentially faster than classical algorithms for sparse matrices. The HHL algorithm was a milestone in quantum algorithms for scientific computations since computationally efficient solutions of linear systems of equations are crucial in different fields including data science, machine learning, and optimization. In addition, the HHL algorithm provides a way to solve linear differential equations by transforming the linear differential equations into a linear system of equations utilizing a multi-step discretization [30-32].

Despite the progress in developing quantum algorithms for linear PDEs and ordinary differential equations (ODEs), less attention has been paid to the nonlinear differential equations (DEs) that appear in various scientific fields. Developing efficient quantum algorithms for nonlinear DEs is challenging since quantum computing only allows unitary operations. Leyton and Osborne [33] employed multiple copies of the state vector to implement nonlinearity. However, in their approach, the required resources scale exponentially in time. In another study, Lloyd et al. [34] employed the mean-field nonlinear quantum algorithm, where the linear Schrödinger evolution was applied to multiple copies of the original system to approximate the nonlinear evolution of the state vector. In their approach, the required resources grow quadratically with time.

Quantum algorithms for CFD applications mostly focus on solving linear equations like the Poisson equation, the linear advection-diffusion equation, and the heat conduction equation. For instance, Cao et al. [3] used Hamiltonian simulations, while Wang et al. [13] utilized an HHL-based approach to solve the Poisson equation. Developing quantum algorithms for the Lattice Boltzmann method is also popular since it has a simple mathematical structure involving only arithmetic computations over a single variable, the distribution function [7]. Budinski [7] developed a quantum algorithm for the linear advection-diffusion equation using LBM. A block-encoding technique was used to encode the linear collision operator, and a quantum walk was utilized for the propagation step. In his work, the process involved in a single time step was encoded in a quantum computer while re-normalizing the post-selected state vector was performed classically. He extended his work to solve incompressible Navier-Stokes equations by developing a quantum LBM algorithm for the stream function-vorticity equation [8]. Another approach is based on the analogy between quantum mechanics and hydrodynamics proposed by Madelung, where the goal is to solve fluid dynamics using the Schrödinger equation [35, 36].

Despite these efforts, studies that solve equations with nonlinear terms are limited, as summarized in Table 1. Gaitan [18] utilized the quantum ODE solver introduced by Kacewicz [37] to solve the Navier-Stokes equations. The algorithm starts with space discretization while leaving time as a continuous parameter. Replacing all of the spatial partial derivatives with approximate algebraic expressions, flux gradients reduce to a function of the conserved variables. The result of spatial discretization is to reduce the NS PDEs to a system of coupled ODEs, $d\boldsymbol{u} / dt = f[\boldsymbol{u}]$.

The only part of the quantum ODE solver that requires a quantum computer is for approximating the time integral, as the Riemann summation uses the quantum amplitude estimation algorithm (QAEA). Because QAEA can achieve a quadratic speed-up over classical methods when performing the numerical integration, the Q-ODE solver can reach an exponentially higher speed than classical deterministic algorithms, and quadratically faster than classical random algorithms. Surprisingly, this straightforward use of the quantum ODE solver for solving Navier-Stokes PDEs went unnoticed for 14 years. Following this study, Oz et al. [38] utilized the quantum ODE algorithm to solve Burgers' equation, and Basu et al. [39] utilized the Q-ODE algorithm to compute the dispersal of submarine volcanic tephra.

Table 1 Quantum algorithms for the quantum computation of nonlinear fluid dynamics.

Approach	Details of approach	Implementation remarks
Nonlinear Q-ODE Solver	Employs a classical approach to calculate the nonlinear flux functions and QAE algorithm for the numerical integration	QA for NSEs [18] and Burgers' eq. [38] without quantum circuit implementation
Hybrid Quantum-classical	Evaluate the cost function in a quantum computer and optimization of variational parameters λ in a classical computer	QA and circuit design to solve NSEs [42]
Carleman Linearization	Employing the Carleman approach to linearize the nonlinear PDEs, suitable for LBM, but not NSEs	1-QA and circuit design for Burgers' eq. [40] (small Reynolds numbers) 2-QA and circuit design for LBM for incompressible fluids [6] with nonlinear collision
Encoding nonlinear terms in the comp. basis	Converts the solution to the computational basis for nonlinear terms, once computed using classical approaches, converts back to amplitudes	Quantum algorithm and circuit design for LBM for Burgers' eq. [41] by nonlinear lattice model
Multiple copies of the state vector (present)	Employing multiple copies of the state vector to calculate nonlinear terms directly with significant improvement over complexity in time	Pure QA and circuit implementation for Burgers' equation

One popular approach is to circumvent nonlinear terms by employing variational quantum computing, where a cost function C(λ) is evaluated in the quantum computer. In contrast, the variational parameter λ is optimized using a classical computer [42]. Another approach is to find a map between the original nonlinear equation and the corresponding linearized equation in higher dimensions. The Carleman linearization technique approximates the nonlinear equation by an infinite sequence of linear equations. However, the ratio between nonlinearity and dissipation, R (similar to Reynolds number, qualitatively) should be relatively weak to efficiently find a solution for a dissipative nonlinear equation [40]. For instance, Liu et al. [40] could solve the viscous Burgers equation for small Reynolds numbers and low nonlinearity to dissipation ratios ($R < 40$) using the quantum Carleman technique. Recently, Itani et al. [6] developed a quantum algorithm for LBM to simulate incompressible fluids, and the Carleman linearization technique was used to encode nonlinear collisions in the quantum algorithm. Steijl [43] introduced another approach that encodes the nonlinear terms on a computational basis. For evaluating the nonlinear terms in this approach, the solution, which is encoded in amplitudes of the quantum state, is converted to a computational basis using analog-to-digital conversion. Once the information is available on a computational basis, strategies in the classical algorithms can be employed to calculate the nonlinear terms. Finally, the information is converted to amplitudes of the quantum state using the digital-to-analog conversion. Steijl [41] utilized this approach to design a quantum circuit for a nonlinear lattice model, which facilitates the evaluation of nonlinear terms using encoding on a computational basis.

However, none of the existing quantum algorithms mentioned in Table 1 directly dealt with the nonlinear terms appearing in the nonlinear partial differential equations of fluid dynamics. *To address this issue, we proposed a novel pure quantum algorithm that directly calculates the nonlinear term by employing multiple copies of the state vector. We also developed complete quantum circuits for the proposed algorithm and then executed them on a quantum simulator.* In the proposed quantum algorithm, the number of resources required for multi-step simulation grows linearly in time, which shows significant improvement over the algorithms proposed by Leyton and Osborne [33] (exponential growth) and Lloyd et al. [34] (quadratic growth).

In this study, we chose Burgers' equation to develop a quantum algorithm for fluid dynamics for application to high-speed compressible flows. This equation is a simplified model of the conservation laws for the compressible NSEs [44, 45]. Despite being a single PDE, it has a similar structure and the same type of quadratic nonlinearity as NSEs [46]. Moreover, the nonlinear term in the Burgers' equation captures two critical nonlinear phenomena in high-speed gas flows: the self-steeping shock structure [47, 48] for continuous and discontinuous initial data, and the evolution of turbulent structures for random initial data. Burgers' turbulence is the study of the solutions to the Burgers' equation with random initial conditions or random forcing, which appear as one of the simplest instances of a nonlinear system out of equilibrium. As a result, Burgers' equation itself has interesting features that have attracted renewed interest in the last decades for applications in cosmology, statistical physics, and fluid dynamics [46].

The quantum algorithm for nonlinear Burgers' equation was implemented as a quantum circuit using advanced quantum linear algebra techniques, including block-encoding for encoding non-unitary matrices, quantum Hadamard product for elementwise multiplication, a linear combination of unitaries to add/subtract the terms, and a compression gadget technique for time evolution. We also solved the linear advection equation using a quantum algorithm based on the block-encoding technique as a preliminary study.

2. Quantum algorithm for solving the linear advection equation

2.1. Governing equation and discretization

The linear advection equation (LAE) is the simplest model in fluid dynamics and provides an excellent opportunity to start building a quantum framework for computational fluid dynamics simulations. The linear advection equation describes the transport of a quantity $u(x,t)$ with a constant velocity of *c* as follows,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0. \tag{1}$$

This equation models the propagation of waves or material along the *x*-axis without any change in shape. For an initial condition of $u_0(x) = u(x,0)$, it has an exact solution of $u(x,t) = u_0(x-ct)$. We employed the finite difference method (FDM) to convert the PDE to a set of algebraic equations as follows,

$$u_i^{n+1} = \left(1 - c\frac{\Delta t}{\Delta t}\right)u_i^n + c\frac{\Delta t}{\Delta t}u_{i-1}^n, \tag{2}$$

where the superscript *n* shows time and subscript *i* indicates the space. The explicit Euler scheme was utilized for temporal discretization, and the upwind scheme was used for spatial discretization. For quantum implementation, it is convenient to convert this set of algebraic equations into a matrix form as follows, where $\boldsymbol{u}^{n+1} = A\boldsymbol{u}^n$,

$$A=\begin{bmatrix} 1-c\dfrac{\Delta t}{\Delta x} & 0 & \cdots & 0 & c\dfrac{\Delta t}{\Delta x} \\ c\dfrac{\Delta t}{\Delta x} & 1-c\dfrac{\Delta t}{\Delta x} & 0 & \vdots & 0 \\ 0 & c\dfrac{\Delta t}{\Delta x} & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & 1-c\dfrac{\Delta t}{\Delta x} & 0 \\ 0 & 0 & 0 & c\dfrac{\Delta t}{\Delta x} & 1-c\dfrac{\Delta t}{\Delta x} \end{bmatrix}. \tag{3}$$

The linear advection equation is a linear transformation, aligning with quantum mechanics' linear nature. However, to design a quantum algorithm for solving LAE using a quantum computer, we must deal with three challenges: quantum state preparation (QSP), non-unitary transformation, and time evolution. Subsections 2.2, 2.3, and 2.4 will discuss these challenges and their corresponding solutions.

2.2. Encoding data in a quantum computer

2.2.1. Encoding information in the computational basis

The first step in designing a quantum algorithm for LAE is initializing the vector $u_0(x)=u(x,0)$ in the quantum computer. There are two approaches to encoding information in a quantum computer: encoding in the computational basis of the state vector and encoding in amplitudes of the state vector. To develop quantum algorithms, we need both encodings. It is worth mentioning that encoding in the computational basis follows the same process as in classical computers. Encoding any arbitrary decimal number on a computational basis requires converting the number to binary. For instance, the binary representation of 50.8 can be written as 110010.11001100110011001101, where the number of qubits required for encoding the integer part (*n_{int}*) depends on how big the number is. On the other hand, depending on the desired accuracy we need, we can allocate some qubits to truncate the fractional part (*n_{frac}*), which can be shown as follows [43],

$$\left|\omega\right\rangle = \overbrace{\left|\omega_1\right\rangle \otimes \left|\omega_2\right\rangle \otimes \cdots \otimes \left|\omega_{n,\mathrm{int}}\right\rangle}^{\text{integer}} \otimes \overbrace{\left|\omega_1\right\rangle \otimes \left|\omega_2\right\rangle \otimes \cdots \otimes \left|\omega_{n,\,frac}\right\rangle}^{fractional}. \tag{4}$$

2.2.2. Encoding information in amplitudes of basis states

Encoding information in the amplitudes of the state vector is a specific feature of quantum computing. To encode an *N*-dimensional vector like $\boldsymbol{u} = \begin{bmatrix} u_0 & u_1 & \cdots & u_{N-1} \end{bmatrix}^T$, we only need *n*=log *N* qubits. First, we normalize the vector to have a unit vector, $\hat{\boldsymbol{u}} = \dfrac{\boldsymbol{u}}{\|\boldsymbol{u}\|_2}$. The goal is to encode the elements of the normalized vector as the amplitudes of basis states in the state vector as follows,

$$\left|\hat{\boldsymbol{u}}\right\rangle = \sum_{i=0}^{N-1} \hat{u}_i \left|i\right\rangle = \hat{u}_0 \left|0\right\rangle + \hat{u}_2 \left|1\right\rangle + \cdots + \hat{u}_{N-1} \left|N-1\right\rangle. \tag{5}$$

Fig. 1 shows a schematic representation of this normalized vector, computational domain, and basis states for encoding an 8-dimensional vector in three qubits.

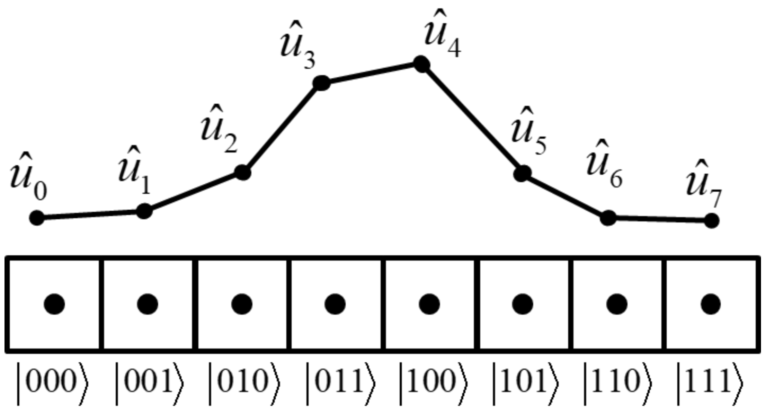


Fig. 1. Schematic representation of encoding information of 8 nodes in amplitudes of basis states for three qubits.

2.2.3. Quantum state preparation

Encoding information on a computational basis is straightforward; we only need to convert the values to their corresponding binary representations. However, implementing information in the amplitudes of basis states is not as easy as in the computational basis. The reason is that we have 2^n information and only *n* qubits. As a result, we need a strategy to find the appropriate rotations required to achieve the desired values in terms of amplitudes of the state vector. In this study, we used the divide-and-conquer technique [49] for quantum state preparation. This technique encodes the information in the amplitudes of the state vector using a sequence of controlled rotations. The goal is to calculate these rotation angles using a state tree where the normalized elements of the vector are located at the bottom. The next level will be calculated by summing the squares of each of the two elements, $\hat{u}_{i-j} = \sum_{n=i}^{j} \hat{u}_n^2$. This process will be continued until we obtain a value equal to one at the top level of the state tree. This process is illustrated for a simple case of encoding a vector with 4 elements in Fig. 2. Rotation angles can be calculated by dividing the elements in the lower level by the elements in the upper level as follows,

$$\alpha_0 = 2\sin^{-1}\left(\frac{\hat{u}_{2-3}}{\hat{u}_{0-3}}\right),\ \alpha_1 = 2\sin^{-1}\left(\frac{\hat{u}_1}{\hat{u}_{0-1}}\right),\ \alpha_2 = 2\sin^{-1}\left(\frac{\hat{u}_3}{\hat{u}_{2-3}}\right). \tag{6}$$

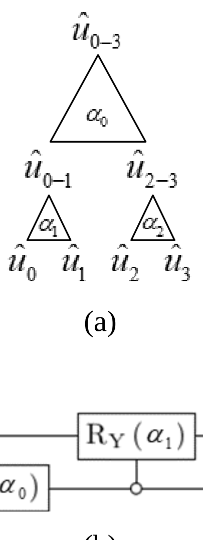


Fig. 2. (a) Representation of the divide-and-conquer strategy and (b) quantum circuit implementations for two qubits.

2.3. Quantum algorithm and circuit design for the linear advection equation

In Section 2.1, we showed that the LAE can be converted to a set of algebraic equations where the advection quantity can be updated as $\boldsymbol{u}^{n+1} = A\boldsymbol{u}^n$. We aim to employ a set of unitary gates to implement matrix A as a quantum circuit. The challenge is that matrix A is not unitary since $AA^\dagger \neq I$.

2.3.1. Implementing a non-unitary transformation using the Block-encoding technique

Block-encoding is a technique for implementing a non-unitary matrix A into a top left block of a larger unitary matrix U$_A$. As a result, we need to extend the state vector by adding extra ancilla qubits, $|\boldsymbol{v}\rangle = |\boldsymbol{u}\rangle|0\rangle = [\boldsymbol{u} \quad 0]^T$. Now, if we apply $U_A|\boldsymbol{v}\rangle$, we obtain $A|\boldsymbol{u}\rangle$ conditioned on post-selecting $|0\rangle$ on ancilla qubits as follows [50],

$$U_A = \begin{bmatrix} A & * \\ * & * \end{bmatrix};\; U_A|\boldsymbol{v}\rangle = U_A|\boldsymbol{u}\rangle|0\rangle = \begin{bmatrix} A\boldsymbol{u} \\ * \end{bmatrix} = A|\boldsymbol{u}\rangle|0\rangle + |*\rangle|1\rangle. \tag{7}$$

Here, the asterisk is a matrix block that has yet to be determined. While block-encoding dense matrices is challenging, sparse circulant matrices can be efficiently block-encoded. Hopefully, the LAE operator and also the operator for calculating the first and second derivatives with periodic boundary conditions form a circulant matrix as follows,

$$A = \begin{bmatrix} \alpha & \gamma & 0 & \cdots & \beta \\ \beta & \alpha & \gamma & \ddots & 0 \\ 0 & \beta & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \alpha & \gamma \\ \gamma & 0 & \cdots & \beta & \alpha \end{bmatrix}. \tag{8}$$

For an N by N matrix, we need $n=\log_2 N$ system qubits, $m=\log_2 s$ ancilla qubits to encode the row indices of non-zero elements, and one additional ancilla qubit to carry the value of non-zero elements. The structure of the block-encoding algorithm for embedding s-sparse circulant matrix A can be shown in Fig. 3.

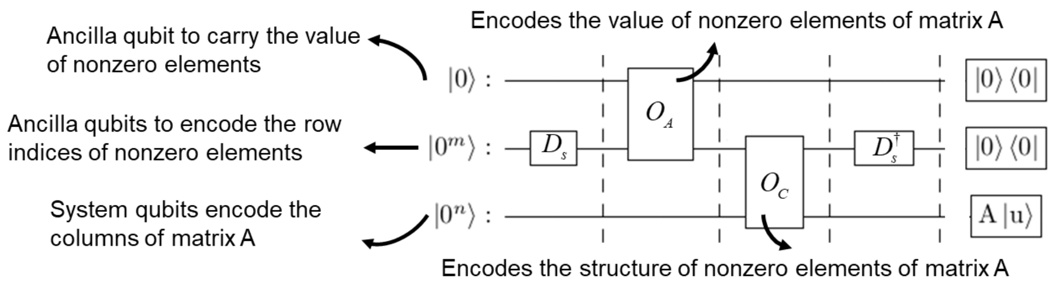

Ancilla qubit to carry the value of nonzero elements
Encodes the value of nonzero elements of matrix A
Ancilla qubits to encode the row indices of nonzero elements
System qubits encode the columns of matrix A
$|0\rangle$:
$|0^m\rangle$:
$|0^n\rangle$:
D_s
O_A
O_C
$D_s^\dagger$
$|0\rangle\langle 0|$
$|0\rangle\langle 0|$
$A|u\rangle$
Encodes the structure of nonzero elements of matrix A

Fig. 3. Schematic circuit representation of block-encoding of an s-sparse matrix.

The detailed circuit implementation for the linear advection equation is depicted in Fig.4. Information of $u(x,t)$ is encoded in the system register (qJ), where three qubits encode values corresponding to the computational nodes (N_x=8). The circuit starts with quantum state preparation to initialize the system qubits with $u_0(x)$. Operator D_s refers to the diffusion operator, which creates an equal superposition of all basis states in the second register (qL), which can be implemented using the Hadamard gates. For a circulant matrix with three non-zero elements (equation 8), we need only two qubits to create four basis states in register L. We define that L=0,1 and 2 correspond to the diagonal, sub-diagonal, and super-diagonal values of the matrix, respectively. Oracle O_A encodes the nonzero elements of matrix A on the first register (q0) using controlled R_Y-rotations (controlled on the values encoded in register L). Oracle O_C encodes the structure of the matrix on the system register using controlled left-shift (subdiagonal) and right-shift (super-diagonal) operations (controlled on the values encoded in register L). Finally, Oracle $D_s^\dagger$ uncomputes the diffusion operator. Rotation angles can be calculated as follows [50],

$$\theta_0 = 2\cos^{-1}(\alpha - 1),\ \theta_1 = 2\cos^{-1}(\beta),\ \theta_2 = 2\cos^{-1}(\gamma). \tag{9}$$

Details on the derivation of these operators, shift operators, and calculation of rotation angles, as well as block-encoding of more general matrices, can be found in the reference [50].

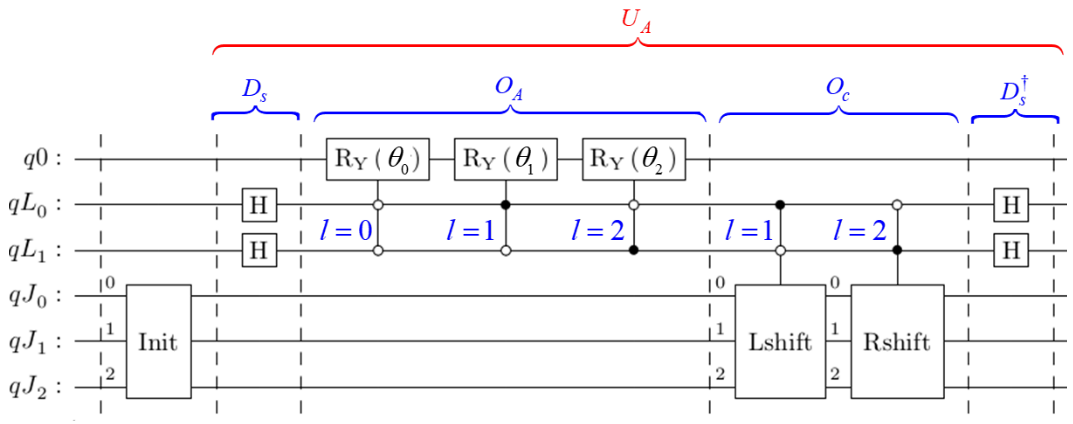


Fig. 4. Quantum circuit implementation for linear advection equation.

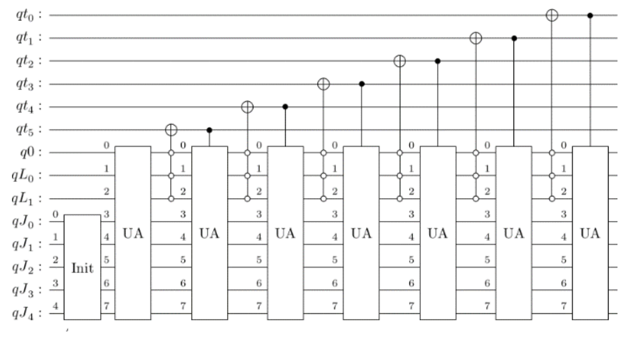
2.3.2. Time evolution strategy (multi-step simulation)

Implementing a multi-step simulation for unitary matrices is straightforward: we apply the unitary operator multiple times, depending on the number of iterations required. However, since obtaining the correct result for non-unitary matrices is conditional, special consideration is necessary for multi-step simulations involving these matrices. As shown in Section 2.3.1, by applying U$_A$ to the extended vector, we obtain $A|\boldsymbol{u}\rangle|0\rangle+|*\rangle|1\rangle$. If we again apply the block-encoded matrix to this state, the desired results will be mixed with garbage values as follows,

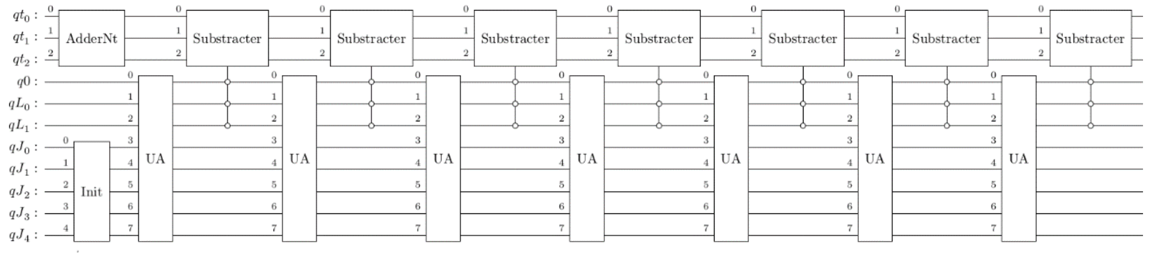
$$U_A\left[|\boldsymbol{u}_1\rangle|0\rangle+|*\rangle|1\rangle\right]=\begin{bmatrix} A & * \\ * & * \end{bmatrix}\begin{bmatrix} \boldsymbol{u}_1 \\ * \end{bmatrix}=\left[A|u_1\rangle+|*\rangle\right]|0\rangle+|*\rangle|1\rangle. \tag{10}$$

To address this issue, we introduce an additional register known as Feynman's clock [30], which encodes the time. After every iteration, a multi-controlled NOT gate (with the control state of zero), controlled by the ancilla qubits of the block-encoding, targets the corresponding qubit in the time register. As mentioned previously, the right values of block-encoding are conditioned on getting zero on ancilla qubits of block-encoding. By this multi-controlled NOT gate, we entangle the good state of block-encoding with the corresponding ancilla qubit in the time register. To apply the next block-encoding, we apply our unitary, controlled on getting one on the ancilla qubit in the time register, ensuring that the unitary operation is applied to the desired portion of the state vector. However, this approach requires *Nt* qubits in the time register for an *Nt*-step simulation.

An alternative approach involves using the compression gadget [51] method, which only needs $\log_2 N_t$ qubits in a counter register. For an N_t-step simulation, we initialize the binary form of the integer number of N_t in the computational basis of the counter register. After each unitary operation, we perform a conditional subtraction on the counter register. The counter register is reduced to zero when we apply the last unitary operation. We then post-select the outcome of measurements conditioned on getting zero on the counter register. Fig. 5 indicates the full circuit implementation for 8 iterations of the linear advection equation using both Feynman's clock and the compression gadget approach.



(a)



(b)

Fig. 5. Final quantum circuit for multi-step simulation of linear advection equation using (a) Feynman's clock and (b) the compression gadget time evolution strategies.

2.4. Numerical results of solving linear advection equation using quantum computing

Fig. 6 compares the numerical results obtained by the classical simulation and quantum simulator with the exact solution of the linear advection equation. The computational domain was initialized using an exponential function of $u_0(x) = u(x,0) = e^{-\left(\frac{x}{5}-2\right)^2}$. The computational domain consists of 32 nodes encoded using 5 system qubits. The solutions were obtained for four time steps, with a constant time step of Δt=1. As seen in Fig. 6, the solution at the final time step has some deviation from the exact solution and classical simulation. The reason is that with an increasing number of time steps, the probability of getting the good state (where the solution is encoded) becomes lower, and a huge number of measurements is needed to get an accurate result. For a practical simulation, an amplitude amplification algorithm can amplify the amplitude of good states compared to the bad states.

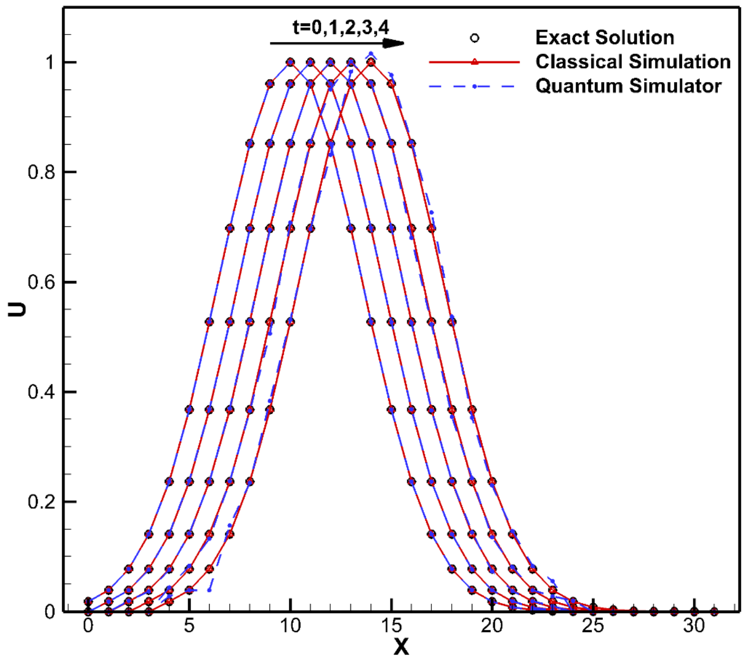


Fig. 6. Comparison of numerical results of solving linear advection equation obtained by quantum simulator compared with classical simulation and exact solution for four-time steps.

3. Pure quantum algorithm for solving nonlinear Burgers' equation

3.1. Governing equation and discretization

One of the main challenges when designing quantum algorithms for governing equations of fluid dynamics is calculating nonlinear terms arising in the transport of momentum. Consequently, Burgers' equation is an ideal candidate as a model equation for fluid dynamics due to its simplicity and nonlinearity. The viscous Burgers' and inviscid Burgers' equations can be expressed as follows,

$$\begin{aligned}&\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=\mu\frac{\partial^2 u}{\partial x^2},\\&\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=0,\end{aligned} \tag{11}$$

where u is the velocity vector, and μ is the viscosity. Employing FDM with an Euler scheme for time, the upwind scheme for the convective term, and the central difference scheme for the second derivative, Burgers' PDE can be discretized in space and time as follows,

$$\begin{aligned} u_i^{n+1} &= u_i^n - \Delta t u_i^n g_i^n + \mu \Delta t h_i^n, \\ g_i^n &= \left(\frac{u_i^n - u_{i-1}^n}{\Delta x} \right), \\ h_i^n &= \left(\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} \right). \end{aligned} \tag{12}$$

Here, g_i^n and h_i^n are the first and second derivatives of u at node i at time n.

To design a quantum algorithm for Burgers' equation, we can break the problem into four main parts, which are summarized in Fig. 7. The first challenge is the calculation of the nonlinear term, $u\dfrac{\partial u}{\partial x}$. Based on the no-cloning theorem, creating an identical copy of an arbitrary unknown quantum state is impossible. In other words, we cannot make a copy of the velocity, calculate its derivative, and finally multiply the resultant by the original velocity. To deal with this problem, we employ multiple copies of the velocity throughout the simulation.

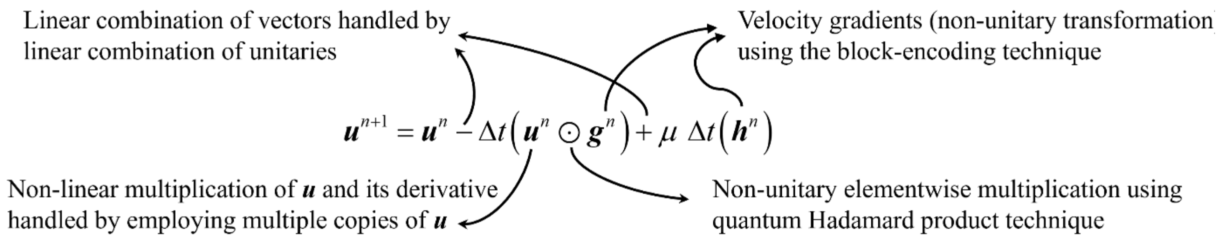

Linear combination of vectors handled by linear combination of unitaries
Velocity gradients (non-unitary transformation using the block-encoding technique
$u^{n+1} = u^n - \Delta t(u^n \odot g^n) + \mu \ \Delta t(h^n)$
Non-linear multiplication of u and its derivative handled by employing multiple copies of u
Non-unitary elementwise multiplication using quantum Hadamard product technique

Fig. 7. The main challenges to designing a quantum algorithm for Burgers' equation and their solutions.

The second challenge is that even with multiple copies of the velocity vector, we cannot directly perform the elementwise multiplication of two vectors (known as the Hadamard product) since it is not a unitary transformation. Section 3.3 will explain how we can employ the quantum Hadamard product (QHP) technique to perform this operation. The third challenge is calculating the first and second derivatives, which are non-unitary transformations (Section 3.2). The fourth challenge is creating a linear combination of all the terms in the discretized equation. This can be achieved using the linear combination of unitaries (LCU) method (Section 3.4).

3.2. Implementation of derivative operators using the block-encoding technique

To design a quantum algorithm for Burgers' equation, we need to calculate the first and second derivatives of the velocity using quantum algorithms. The calculation of the first and second derivatives as a matrix operation can be summarized as follows, respectively,

$$\begin{bmatrix} g_0^n \\ g_1^n \\ \vdots \\ g_{m-2}^n \\ g_{m-1}^n \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta x} & 0 & \cdots & 0 & \frac{-1}{\Delta x} \\ \frac{-1}{\Delta x} & \frac{1}{\Delta x} & 0 & \cdots & 0 \\ 0 & \frac{-1}{\Delta x} & \ddots & 0 & 0 \\ 0 & 0 & \ddots & \frac{1}{\Delta x} & 0 \\ 0 & 0 & 0 & \frac{-1}{\Delta x} & \frac{1}{\Delta x} \end{bmatrix} \begin{bmatrix} u_0^n \\ u_1^n \\ \vdots \\ u_{m-2}^n \\ u_{m-1}^n \end{bmatrix} = \frac{1}{\Delta x} \begin{bmatrix} 1 & 0 & \cdots & 0 & -1 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & \ddots & 0 & 0 \\ 0 & 0 & \ddots & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_0^n \\ u_0^n \\ \vdots \\ u_{m-2}^n \\ u_{m-1}^n \end{bmatrix}, \tag{13}$$

$$\begin{bmatrix} h_0^n \\ h_1^n \\ \vdots \\ h_{m-2}^n \\ h_{m-1}^n \end{bmatrix} = \begin{bmatrix} \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & 0 & \cdots & \frac{1}{\Delta x^2} \\ \frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & \ddots & 0 \\ 0 & \frac{1}{\Delta x^2} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} \\ \frac{1}{\Delta x^2} & 0 & \cdots & \frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} \end{bmatrix} \begin{bmatrix} u_0^n \\ u_1^n \\ \vdots \\ u_{m-2}^n \\ u_{m-1}^n \end{bmatrix} = \frac{-1}{\Delta x^2} \begin{bmatrix} 2 & -1 & 0 & \cdots & -1 \\ -1 & 2 & -1 & \ddots & 0 \\ 0 & -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 2 & -1 \\ -1 & 0 & \cdots & -1 & 2 \end{bmatrix} \begin{bmatrix} u_0^n \\ u_1^n \\ \vdots \\ u_{m-2}^n \\ u_{m-1}^n \end{bmatrix}. \tag{14}$$

While these operations are not unitary, both are banded sparse matrices that can be easily implemented as a quantum operator using the block-encoding technique. These matrices have the same structure as equation (8). As a result, they can be implemented using the circuit shown in Fig. 4. The only difference is the calculation of rotation angles, which are different from the linear advection equation. It is also worth mentioning that we use uniform spacing, which allows us to take the $\frac{1}{\Delta x}$ and $\frac{1}{\Delta x^2}$ out of the matrix. The reason is that their values can be larger than one, which makes it impossible to calculate the rotation angles. Moreover, it is easier to calculate rotation angles for the second derivative by taking $\frac{-1}{\Delta x^2}$ out of the matrix instead of $\frac{1}{\Delta x^2}$. As a result, the operator for the second derivative calculates $h_i = \frac{-1}{\Delta x^2}\left(-u_{i+1} + 2u_i - u_{i-1}\right)$. Rotation angles for the first and second derivatives can be calculated based on equation (9). Fig. 8 shows the quantum circuit for calculating the first and second derivatives.

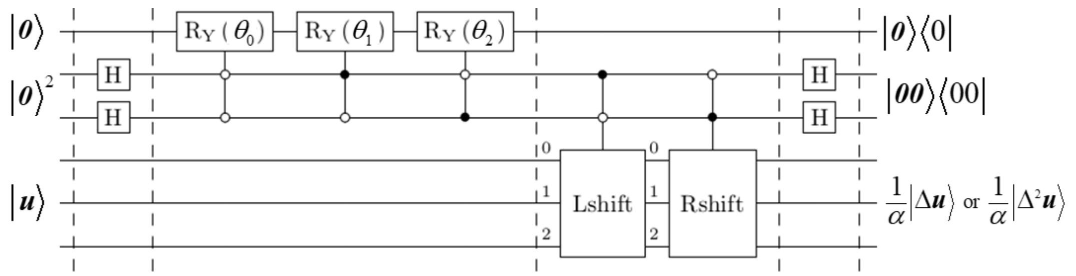

$|\boldsymbol{0}\rangle$
$|\boldsymbol{0}\rangle^2$
$|\boldsymbol{u}\rangle$
H
H
$\mathrm{R_Y}(\theta_0)$
$\mathrm{R_Y}(\theta_1)$
$\mathrm{R_Y}(\theta_2)$
Lshift
Rshift
0
1
2
$|\boldsymbol{0}\rangle\langle 0|$
$|\boldsymbol{00}\rangle\langle 00|$
$\frac{1}{\alpha}|\Delta\boldsymbol{u}\rangle$ or $\frac{1}{\alpha}|\Delta^2\boldsymbol{u}\rangle$

Fig. 8. Quantum circuit implementation for calculating the first and second derivatives using the block-encoding technique.

After applying the block-encoding, the derivative will be scaled by $1/\alpha$. Since we applied the Hadamard gate to the ancilla qubits four times, α equals four. The state vector for the first and second derivatives can be written as follows, respectively,

$$\begin{aligned}
&|\boldsymbol{u}\rangle = \frac{1}{\|u\|}\sum_{i=1}^{m} u_i |i\rangle, \\
&A_D|\boldsymbol{u}\rangle|0\rangle^2|0\rangle = \frac{1}{\alpha}\Big(D|\boldsymbol{u}\rangle|0\rangle^2|0\rangle + |*\rangle|\perp\rangle\Big), \\
&\frac{1}{\alpha}D|\boldsymbol{u}\rangle = \frac{1}{\alpha}|\Delta\boldsymbol{u}\rangle = \frac{1}{\|u\|}\frac{1}{\alpha}\sum_{i=1}^{m}(u_i - u_{i-1})|i\rangle,
\end{aligned} \tag{15}$$

and

$$\begin{aligned}
&A_{D^2}|\boldsymbol{u}\rangle|0\rangle^2|0\rangle = \frac{1}{\alpha}\Big(D^2|\boldsymbol{u}\rangle|0\rangle^2|0\rangle + |*\rangle|\perp\rangle\Big), \\
&\frac{1}{\alpha}D^2|\boldsymbol{u}\rangle = \frac{1}{\alpha}|\Delta^2\boldsymbol{u}\rangle = \frac{1}{\|u\|}\frac{1}{\alpha}\sum_{i=1}^{m}(-u_{i-1} + 2u_i - u_{i+1})|i\rangle.
\end{aligned} \tag{16}$$

3.3. Quantum Hadamard product (element-wise multiplication)

The next term we need to calculate is the elementwise multiplication of the velocity vector and its gradient. This operation can be written as follows,

$$\boldsymbol{u}\odot\boldsymbol{g} = \begin{bmatrix} u_0^n \\ u_1^n \\ \vdots \\ u_{m-2}^n \\ u_{m-1}^n \end{bmatrix} \odot \begin{bmatrix} g_0^n \\ g_1^n \\ \vdots \\ g_{m-2}^n \\ g_{m-1}^n \end{bmatrix} = \begin{bmatrix} u_0^n g_0^n \\ u_1^n g_1^n \\ \vdots \\ u_{m-2}^n g_{m-2}^n \\ u_{m-1}^n g_{m-1}^n \end{bmatrix}. \tag{17}$$

As mentioned, the elementwise multiplication of two vectors is not a unitary operation since it does not preserve the norm. This can be easily verified through a simple calculation as follows,

$$\begin{aligned} &\boldsymbol{u} = \begin{bmatrix} u_0 & u_1 & \cdots & u_n \end{bmatrix}^T \rightarrow \tilde{\boldsymbol{u}} = \tfrac{\boldsymbol{u}}{\|\boldsymbol{u}\|_2}; \quad \|\boldsymbol{u}\|_2 = \sqrt{\sum_i u_i^2} \rightarrow \|\tilde{\boldsymbol{u}}\|_2 = 1, \\ &\tilde{\boldsymbol{u}} \odot \tilde{\boldsymbol{u}} = \begin{bmatrix} \frac{u_0^2}{\sum_i u_i^2} & \frac{u_1^2}{\sum_i u_i^2} & \cdots & \frac{u_n^2}{\sum_i u_i^2} \end{bmatrix}^T \rightarrow \|\tilde{\boldsymbol{u}} \odot \tilde{\boldsymbol{u}}\|_2 = \frac{\sqrt{\sum_i u_i^4}}{\sum_i u_i^2} \neq 1. \end{aligned} \tag{18}$$

We employ the quantum Hadamard product technique to perform this operation. Let's assume that our vectors of $\boldsymbol{u}$ and $\boldsymbol{g}$ have two elements, which enables us to represent each of them with one qubit as follows,

$$\begin{aligned} &|\boldsymbol{u}\rangle = \left(u_0|0\rangle + u_1|1\rangle\right), \\ &|\boldsymbol{g}\rangle = \left(g_0|0\rangle + g_1|1\rangle\right). \end{aligned} \tag{19}$$

The quantum circuit for the quantum Hadamard product is shown in Fig. 9. Based on the postulates of quantum computing, the state of a multi-qubit system is the tensor product of state spaces of individual qubits. It can be easily seen that our desired vector of $|\boldsymbol{u}\rangle \odot |\boldsymbol{g}\rangle$ is already present in the tensor product of the two vectors. We need to rearrange them using a CNOT gate, such that we can access the desired vector as follows,

$$\begin{aligned} 0:&\ |\boldsymbol{u}\rangle \otimes |\boldsymbol{g}\rangle = u_0 g_0|00\rangle + u_0 g_1|01\rangle + u_1 g_0|10\rangle + u_1 g_1|11\rangle, \\ 1:&\ \xrightarrow{CNOT} u_0 g_0|00\rangle + u_1 g_1|01\rangle + u_1 g_0|10\rangle + u_0 g_1|11\rangle. \end{aligned} \tag{20}$$

If we post-select the results conditioned on getting zero on the second qubit (which is colored blue in Eq. (20)), we get an elementwise multiplication of two vectors in the first qubit.

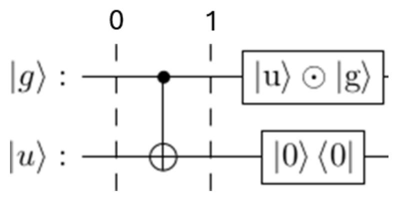


Fig. 9. Quantum circuit for quantum Hadamard product technique.

3.4. Linear combination of unitaries

The final step in implementing the quantum algorithm for solving Burgers' equation is to sum all the terms we calculated in the previous subsections. The general form of the linear combination of unitaries can be expressed as follows,

$$U = \sum_{i=0}^{2^m-1} \alpha_i U_i. \tag{21}$$

The value of α_i is the weight of the ith unitary, U_i. The general form of the circuit for implementing the LCU can be seen in Fig. 10. To create a linear combination of 2^m terms, we need m ancilla qubits. We apply a preparation oracle (PREP) on the ancilla qubits, followed by controlled unitaries conditioned on selecting from the ancilla qubits and targeting the system qubits (SELECT). Finally, we uncompute the ancilla qubits using PPREP† to create interference and make the summation.

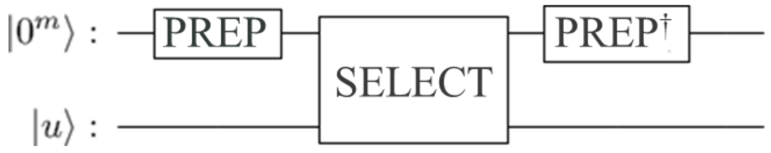


Fig. 10. Schematic of the quantum circuit for the linear combination of unitaries.

To make a uniform combination of unitaries, a preparation oracle can be equivalent to applying Hadamard gates to all ancilla qubits. Otherwise, R$_Y$ rotations can be used to create non-uniform combinations. For instance, assuming $U|0\rangle = |\psi\rangle$ and $V|0\rangle = |\phi\rangle$, the quantum circuit implementation of $|\psi\rangle - b|\phi\rangle$ can be seen in Fig. 11.

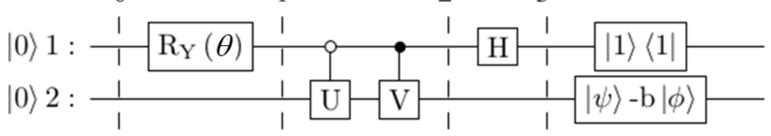


Fig. 11. Quantum circuit for calculating $|\psi\rangle - b|\phi\rangle$ using the linear combination of unitaries.

For the circuit shown in Fig. 11, the state vector can be calculated as follows,

$$\begin{aligned}
&0: \ |0\rangle|0\rangle \to 1: \ |0\rangle\left(\cos\theta|0\rangle + \sin\theta|1\rangle\right), \\
&2: \ \cos\theta|\psi\rangle|0\rangle + \sin\theta|\phi\rangle|1\rangle, \\
&3: \ \cos\theta|\psi\rangle\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right) + \sin\theta|\phi\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), \\
&3: \ \frac{1}{\sqrt{2}}\left(\cos\theta|\psi\rangle + \sin\theta|\phi\rangle\right)|0\rangle + \frac{1}{\sqrt{2}}\left(\cos\theta|\psi\rangle - \sin\theta|\phi\rangle\right)|1\rangle.
\end{aligned} \tag{22}$$

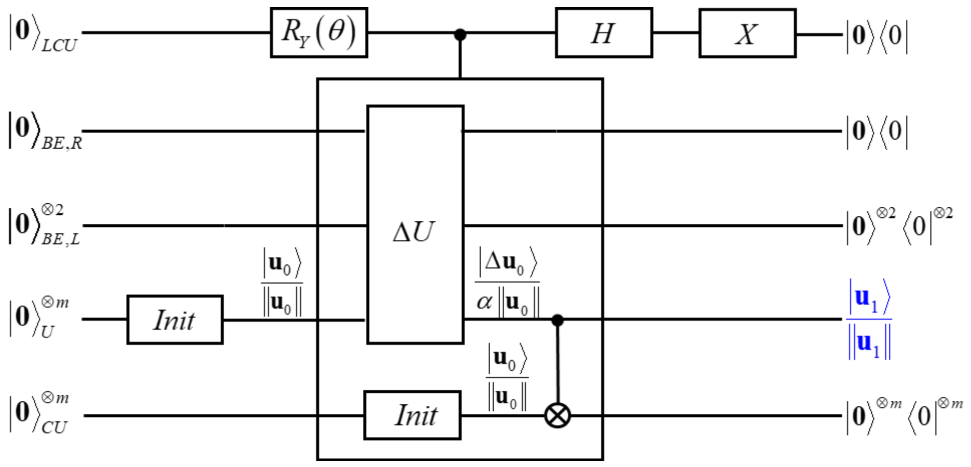
Measuring the state vector conditioned on getting $|1\rangle$ on the ancilla qubit guarantees that we will obtain the desired result. Moreover, the rotation angle can be calculated as follows,

$$\theta = \cos^{-1}\left(\frac{1}{\sqrt{1+b^2}}\right) \to \cos\theta|\psi\rangle - \sin\theta|\phi\rangle = \frac{1}{\sqrt{1+b^2}}\left(|\psi\rangle - b|\phi\rangle\right). \tag{23}$$

3.5. Circuit design for a single-step simulation

3.5.1. Quantum circuit for inviscid Burgers' equation

For the sake of simplicity, we first start with the circuit implementation of the inviscid Burgers' equation as shown in Fig. 12. This circuit includes 5 registers: From the top, the first register is the ancilla qubits for LCU, the second and third registers are ancilla qubits for block-encoding, the fourth and fifth are the main register and a copy of the main register, which encode the velocity and a copy of velocity, respectively. We only need log N_x qubits for the main register and its copy to simulate a computational domain with N_x spatial nodes or cells.



$|\mathbf{0}\rangle_{LCU}$
$|\mathbf{0}\rangle_{BE,R}$
$|\mathbf{0}\rangle^{\otimes 2}_{BE,L}$
$|\mathbf{0}\rangle^{\otimes m}_{U}$
$|\mathbf{0}\rangle^{\otimes m}_{CU}$
$R_Y(\theta)$
H
X
ΔU
Init
Init
$\frac{|\mathbf{u}_0\rangle}{\|\mathbf{u}_0\|}$
$\frac{|\Delta\mathbf{u}_0\rangle}{\alpha\|\mathbf{u}_0\|}$
$\frac{|\mathbf{u}_0\rangle}{\|\mathbf{u}_0\|}$
$|\mathbf{0}\rangle\langle 0|$
$|\mathbf{0}\rangle\langle 0|$
$|\mathbf{0}\rangle^{\otimes 2}\langle 0|^{\otimes 2}$
$\frac{|\mathbf{u}_1\rangle}{\|\mathbf{u}_1\|}$
$|\mathbf{0}\rangle^{\otimes m}\langle 0|^{\otimes m}$

Fig. 12. Quantum circuit for one iteration of the inviscid Burgers' equation.

The evolution of the state vector can be written as follows (the changes are color-coded; any change in the state vector is colored blue and rearranging the state vector is colored red),

$$
\begin{aligned}
&|0\rangle^{\otimes m}|0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|0\rangle \xrightarrow{Init\,\&\,R_Y(\theta)} \tfrac{1}{\|u_0\|}|0\rangle^{\otimes m}|u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle\left(\cos\theta|0\rangle+\sin\theta|1\rangle\right),\\
&\xrightarrow{Init\,\&\,\Delta U} \tfrac{\cos\theta}{\|u_0\|}|0\rangle^{\otimes m}|u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|0\rangle+\tfrac{\sin\theta}{\|u_0\|}\tfrac{1}{\|u_0\|}|u_0\rangle^{\otimes m}\tfrac{1}{\alpha}|\Delta u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|1\rangle,\\
&\xrightarrow{CNOT} \tfrac{\cos\theta}{\|u_0\|}|0\rangle^{\otimes m}|u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|0\rangle+\tfrac{\sin\theta}{\|u_0\|}\tfrac{1}{\|u_0\|}|0\rangle^{\otimes m}\tfrac{1}{\alpha}|u_0\Delta u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|1\rangle,\\
&\xrightarrow{H} \tfrac{\cos\theta}{\|u_0\|}|0\rangle^{\otimes m}|u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle\left(\tfrac{|0\rangle+|1\rangle}{\sqrt{2}}\right)+\tfrac{\sin\theta}{\|u_0\|}\tfrac{1}{\|u_0\|}|0\rangle^{\otimes m}\tfrac{1}{\alpha}|u_0\Delta u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle\left(\tfrac{|0\rangle-|1\rangle}{\sqrt{2}}\right),\\
&\xrightarrow{X} \tfrac{1}{\sqrt{2}\|u_0\|}\Big\{|0\rangle^{\otimes m}\Big[\cos\theta|u_0\rangle^{\otimes m}+\tfrac{1}{\alpha\|u_0\|}\sin\theta|u_0\Delta u_0\rangle^{\otimes m}\Big]|0\rangle^{\otimes 2}|0\rangle|1\rangle\\
&\qquad\qquad +|0\rangle^{\otimes m}\Big[\cos\theta|u_0\rangle^{\otimes m}-\tfrac{1}{\alpha\|u_0\|}\sin\theta|u_0\Delta u_0\rangle^{\otimes m}\Big]|0\rangle^{\otimes 2}|0\rangle|0\rangle\Big\}.
\end{aligned}
\tag{24}
$$

After applying the $R_Y(\theta)$ to the ancilla qubit in the LCU register, this qubit is in the state of $\cos\theta|0\rangle+\sin\theta|1\rangle$. This is followed by a controlled unitary targeting on all registers except the ancilla qubit of the LCU register, which is the control qubit. When the LCU qubit is in the state $|0\rangle$, it does nothing, and when it is in the state $|1\rangle$, this unitary will be applied. As a result, after applying the CNOT gate, the main register is in the state $|u_0\rangle$ when the LCU qubit is in state $|0\rangle$ and it is in the state $|u_0\Delta u_0\rangle$ when the LCU qubit is in the state $|1\rangle$. To take advantage of the quantum amplitude interference feature, we apply the Hadamard gate to the LCU qubit. As a result, the main register is in the state $\cos\theta|u_0\rangle-\frac{1}{\alpha\|u_0\|}\sin\theta|u_0\Delta u_0\rangle$ ($\cos\theta|u_0\rangle+\frac{1}{\alpha\|u_0\|}\sin\theta|u_0\Delta u_0\rangle$) when the LCU qubit is in state $|1\rangle$ ($|0\rangle$). Since we need to calculate $u_{n+1}=u_n-\frac{\Delta t}{\Delta x}u_n\Delta u_n$, the main register is in the desired state when the LCU qubit is in state $|1\rangle$. Finally, we apply an X gate to the LCU qubit since we prefer to get $|0\rangle$ on ancilla qubits.

Post-selecting the measurements conditioned on getting zero states on all ancilla qubits and also the register that holds a copy of the velocity guarantees that we get $\frac{1}{\|\mathbf{u}_1\|}|\mathbf{u}_1\rangle$ on the main register. The rotation angle for each iteration is only a function of Δt, Δx, α, and the scaling factor of velocity from the previous step, which can be calculated as follows,

$$\cos\theta_{n+1} = \frac{1/\alpha\left\|u_n\right\|}{\sqrt{\left(1/\alpha\left\|u_n\right\|\right)^2 + \left(\Delta t/\Delta x\right)^2}};\ \sin\theta_{n+1} = \frac{\Delta t/\Delta x}{\sqrt{\left(1/\alpha\left\|u_n\right\|\right)^2 + \left(\Delta t/\Delta x\right)^2}}. \tag{25}$$

Here, considering *n* equal to zero, calculates the rotation angle for the first iteration. Substituting the *sinθ* and *cosθ* in the final outcome of the state vector on the main register conditioned on getting zero on all other qubits (colored with red in equation 24) returns the scaled velocity vector $\frac{1}{\|\mathbf{u}_1\|}\left|\mathbf{u}_1\right\rangle$ as follows,

$$\frac{1}{\sqrt{2}\alpha}\frac{1}{\left\|u_0\right\|^2}\frac{1}{\sqrt{\left(1/\alpha\left\|u_0\right\|\right)^2 + \left(\Delta t/\Delta x\right)^2}}\left[\left|u_0\right\rangle^{\otimes m} - \frac{\Delta t}{\Delta x}\left|u_0\Delta u_0\right\rangle^{\otimes m}\right] = \frac{\left|u_1\right\rangle}{\left\|u_1\right\|}. \tag{26}$$

The scaling factor for each iteration can also be generalized as follows,

$$\left\|u_{n+1}\right\| = \sqrt{2}\alpha\left\|u_n\right\|^2\sqrt{\left(1/\alpha\left\|u_n\right\|\right)^2 + \left(\Delta t/\Delta x\right)^2}. \tag{27}$$

Similar to the rotation angle, the scaling factor for the velocity is also independent of the solution, which allows us to pre-calculate all the scaling factors and rotation angles. As a result, we do not need to measure in between to calculate the rotation angles classically. This is important since it allows us to design a pure quantum algorithm.

3.5.2. Quantum circuit for viscous Burgers' equation

The quantum circuit for the viscous Burgers' equation is also similar to the inviscid Burgers' equation except for an additional term for the second derivative of velocity, as shown in Fig. 13. As a result, we need to calculate the linear combination of three terms instead of two terms. For this reason, we need an extra ancillary qubit for the LCU register.

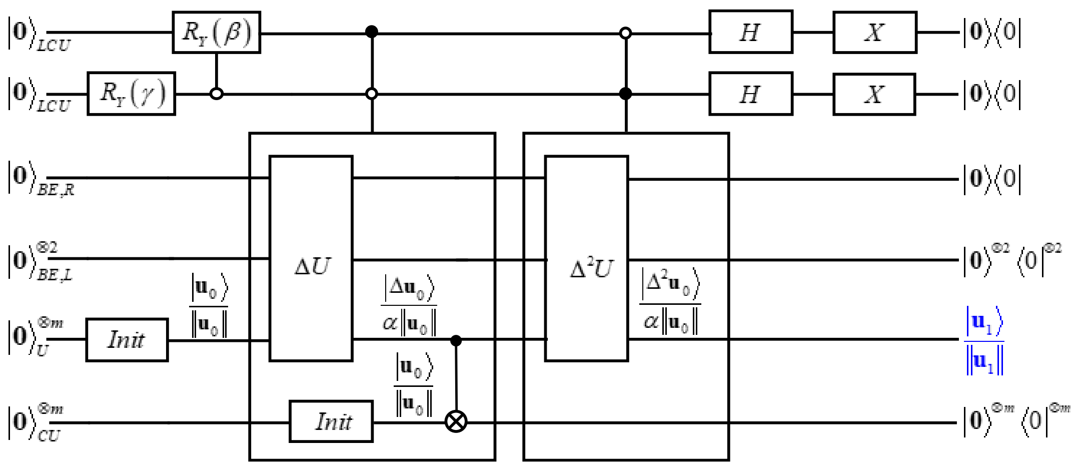

$|0\rangle_{LCU}$
$R_Y(\beta)$
H
X
$|0\rangle\langle 0|$
$|0\rangle_{LCU}$
$R_Y(\gamma)$
H
X
$|0\rangle\langle 0|$
$|0\rangle_{BE,R}$
$|0\rangle\langle 0|$
$|0\rangle^{\otimes 2}_{BE,L}$
ΔU
$\Delta^2 U$
$|0\rangle^{\otimes 2}\langle 0|^{\otimes 2}$
$\frac{|\mathbf{u}_0\rangle}{\|\mathbf{u}_0\|}$
$\frac{|\Delta\mathbf{u}_0\rangle}{\alpha\|\mathbf{u}_0\|}$
$\frac{|\Delta^2\mathbf{u}_0\rangle}{\alpha\|\mathbf{u}_0\|}$
$|0\rangle^{\otimes m}_{U}$
$Init$
$\frac{|\mathbf{u}_1\rangle}{\|\mathbf{u}_1\|}$
$\frac{|\mathbf{u}_0\rangle}{\|\mathbf{u}_0\|}$
$|0\rangle^{\otimes m}_{CU}$
$Init$
$|0\rangle^{\otimes m}\langle 0|^{\otimes m}$

Fig. 13. Quantum circuit for one iteration of Burgers' equation.

Similar to the inviscid Burgers' equation, we can write the evolution of the state vector for one iteration of solving viscous Burgers' equation as follows (the changes are color-coded; any change in the state vector is colored blue, and rearranging the state vector is colored red),

$$\begin{aligned}
&|0\rangle^{\otimes m}|0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|00\rangle \xrightarrow{Init} \tfrac{1}{\|u_0\|}|0\rangle^{\otimes m}|u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|00\rangle,\\
&\xrightarrow{R_Y(\alpha_0)\&R_Y(\alpha_1)} \tfrac{1}{\|u_0\|}|0\rangle^{\otimes m}|u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle\left(\overbrace{\cos\gamma\cos\beta}^{a}|00\rangle+\overbrace{\cos\gamma\sin\beta}^{b}|01\rangle+\overset{c}{\sin\gamma}|10\rangle\right),\\
&\xrightarrow{Init\,\&\,\Delta U\,\&\,CNOT\,\&\,\Delta^2 U} \tfrac{a}{\|u_0\|}|0\rangle^{\otimes m}|u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|00\rangle+\tfrac{b}{\|u_0\|}\tfrac{1}{\|u_0\|}|0\rangle^{\otimes m}\tfrac{1}{\alpha}|u_0\Delta u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|01\rangle\\
&\qquad+\tfrac{c}{\|u_0\|}|0\rangle^{\otimes m}\tfrac{1}{\alpha}|\Delta^2 u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle|10\rangle,\\
&\xrightarrow{H} \tfrac{a}{2\|u_0\|}|0\rangle^{\otimes m}|u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle\{|00\rangle+|01\rangle+|10\rangle+|11\rangle\}\\
&\qquad+\tfrac{b}{2\|u_0\|}\tfrac{1}{\|u_0\|}|0\rangle^{\otimes m}\tfrac{1}{\alpha}|u_0\Delta u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle\{|00\rangle-|01\rangle+|10\rangle-|11\rangle\}\\
&\qquad+\tfrac{c}{2\|u_0\|}|0\rangle^{\otimes m}\tfrac{1}{\alpha}|\Delta^2 u_0\rangle^{\otimes m}|0\rangle^{\otimes 2}|0\rangle\{|00\rangle+|01\rangle-|10\rangle-|11\rangle\},\\
&\xrightarrow{X} \tfrac{1}{2\|u_0\|}|0\rangle^{\otimes m}\left[a|u_0\rangle^{\otimes m}-\tfrac{1}{\alpha\|u_0\|}b|u_0\Delta u_0\rangle^{\otimes m}-\tfrac{1}{\alpha}c|\Delta^2 u_0\rangle^{\otimes m}\right]|0\rangle^{\otimes 2}|0\rangle|00\rangle.
\end{aligned}\tag{28}$$

Here, also, post-selecting the measurements conditioned on getting zero states on all ancilla qubits guarantees that we get $\frac{1}{\|\mathbf{u}_1\|}|\mathbf{u}_1\rangle$ on the main register. Rotation angles can be calculated by comparing the final outcome of the state vector on the main register with our goal of calculating $u_{n+1} = u_n - \frac{\Delta t}{\Delta x} u_n \Delta u_n - \frac{\mu \Delta t}{\Delta x^2}\left(-\Delta^2 u_n\right)$ as follows,

$$\begin{aligned} \cos\gamma_{n+1} &= \sqrt{\frac{\left(1/\alpha\|u_n\|\right)^2 + \left(\Delta t/\Delta x\right)^2}{\left(1/\alpha\|u_n\|\right)^2 + \left(\Delta t/\Delta x\right)^2 + \left(\mu\Delta t/\|u_n\|\Delta x^2\right)^2}}; \quad \cos\beta_{n+1} = \frac{1/\alpha\|u_n\|}{\sqrt{\left(1/\alpha\|u_n\|\right)^2 + \left(\Delta t/\Delta x\right)^2}}, \\ \sin\gamma_{n+1} &= \frac{\mu\Delta t/\|u_n\|\Delta x^2}{\sqrt{\left(1/\alpha\|u_n\|\right)^2 + \left(\Delta t/\Delta x\right)^2 + \left(\mu\Delta t/\|u_n\|\Delta x^2\right)^2}}; \quad \sin\beta_{n+1} = \frac{\Delta t/\Delta x}{\sqrt{\left(1/\alpha\|u_n\|\right)^2 + \left(\Delta t/\Delta x\right)^2}}. \end{aligned} \tag{29}$$

Substituting these values in the final outcome of the state vector on the main register (colored with red in equation 28) returns the scaled velocity vector $\frac{1}{\|\mathbf{u}_1\|}|\mathbf{u}_1\rangle$ as follows,

$$\frac{1}{2\|u_0\|^2\alpha}\frac{|0\rangle^{\otimes m}\left[|u_0\rangle^{\otimes m} - \frac{\Delta t}{\Delta x}|u_0\Delta u_0\rangle^{\otimes m} - \frac{2\mu\Delta t}{\Delta x^2}|\Delta^2 u_0\rangle^{\otimes m}\right]|0\rangle^{\otimes 2}|0\rangle|01\rangle}{\sqrt{\left(1/\alpha\|u_0\|\right)^2 + \left(\Delta t/\Delta x\right)^2 + \left(\mu\Delta t/\|u_0\|\Delta x^2\right)^2}} = \frac{|u_1\rangle}{\|u_1\|}, \tag{30}$$

where the scaling factors for each iteration also can be generalized as follows,

$$\|u_{n+1}\| = 2\alpha\|u_n\|^2\sqrt{\left(1/\alpha\|u_n\|\right)^2 + \left(\Delta t/\Delta x\right)^2 + \left(\mu\Delta t/\|u_n\|\Delta x^2\right)^2}. \tag{31}$$

3.6. Multiple copies and reusing qubits

As previously mentioned, we cannot make a copy of the velocity vector and calculate terms like u^2 or $u\frac{\partial u}{\partial x}$ due to the no-cloning theorem. As a result, we need to carry a copy of the velocity vector from the beginning of the simulation to calculate the nonlinear term using the QHP technique. To know how many copies we need to calculate a simulation with N_t number of iterations, we can simplify the problem to calculate $\left(u^2\right)^{N_t}$. This clarifies how the number of copies increases with N_t, as shown in Fig. 14.

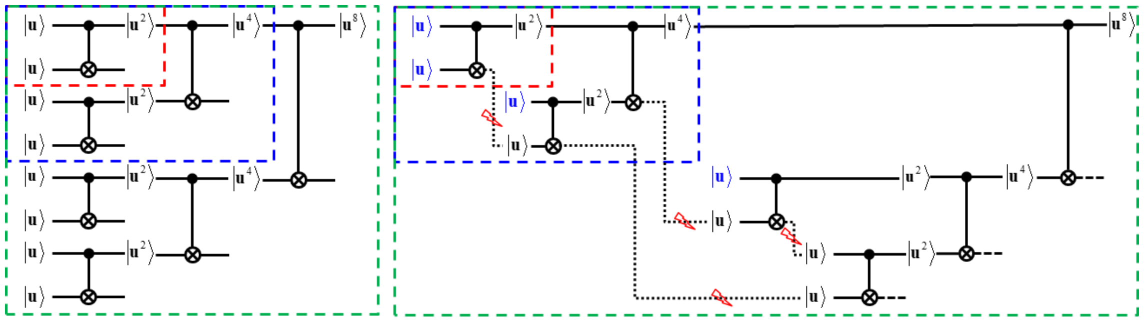
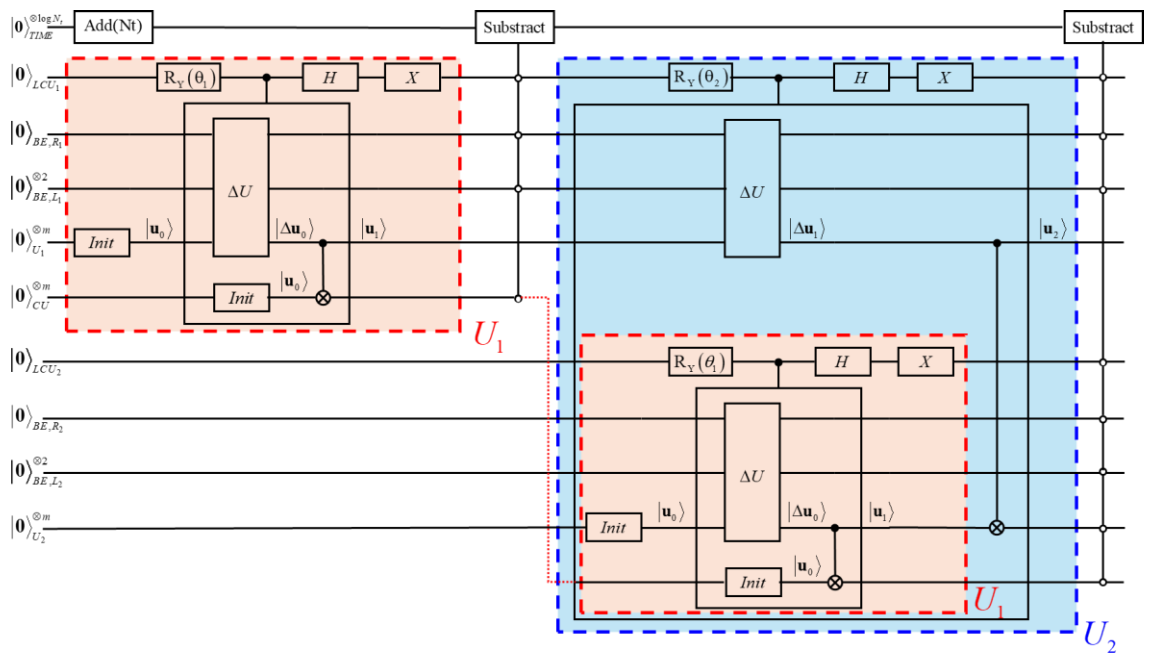


Fig. 14. Schematic representation of number copies required for multi-step calculation of $\left(u^2\right)^{N_t}$ without reusing qubits (left) and with reusing qubits (right).

It can be easily seen that for terms like u^2 (monomial of degree 2), the number of copies increases exponentially with the number of iterations, which can be expressed as 2^{N_t}. It is well known that this approach cannot be practical because, after only 30 iterations, we would need approximately one billion copies. However, if we reuse the copies from previous iterations, we can exponentially improve the dependency of the number of copies to time (N_t). If the velocity vector contains N_x nodes, we need log N_x qubits to encode the velocity in a quantum computer. The number of qubits required for conducting N_t steps of simulation is equal to $2^{N_t} \log N_x$ without reusing qubits and $(N_t + 1)\log N_x$ when reusing qubits. The next subsection will explain how we can reuse qubits to solve Burgers' equation.

3.7. Circuit design for multi-step simulation

Fig. 15 shows the quantum circuit for two steps for solving the inviscid Burgers' equation. As discussed in Section 2.3.2, we must add a time register for a multi-step simulation. This register will be initialized with the binary encoding of N_t in the computational basis of the time register. After each iteration, we do a conditional subtraction from the time register until this register reaches zero after the final step. Moreover, we must add a copy of all registers mentioned in the previous section for each simulation step.



$|0\rangle_{TIME}^{\otimes \log N_t}$
Add(Nt)
Substract
Substract
$|0\rangle_{LCU_1}$
$|0\rangle_{BE,R_1}$
$|0\rangle_{BE,L_1}^{\otimes 2}$
$|0\rangle_{U_1}^{\otimes m}$
$|0\rangle_{CU}^{\otimes m}$
$|0\rangle_{LCU_2}$
$|0\rangle_{BE,R_2}$
$|0\rangle_{BE,L_2}^{\otimes 2}$
$|0\rangle_{U_2}^{\otimes m}$
$R_Y(\theta_1)$
$R_Y(\theta_2)$
H
X
ΔU
$Init$
$|\mathbf{u}_0\rangle$
$|\Delta\mathbf{u}_0\rangle$
$|\mathbf{u}_1\rangle$
$|\Delta\mathbf{u}_1\rangle$
$|\mathbf{u}_2\rangle$
U_1
U_2

Fig. 15. Quantum circuit for two iterations of the inviscid Burgers' equation.

We designed the quantum algorithm such that it takes $|\boldsymbol{u}_n\rangle$ from the previous iteration and gives $|\boldsymbol{u}_{n+1}\rangle$ at the end of the controlled unitary in the main register (U). For calculating $|\boldsymbol{u}_{n+1}\rangle$, we need to have $|\boldsymbol{u}_n\rangle$ in the CU register to calculate the Hadamard product ($|\boldsymbol{u}_n\rangle_{CU} \odot |\Delta\boldsymbol{u}_n\rangle_U$). To add the terms that appear in the Burgers' equation, we use the LCU technique on the main register. All other qubits must be in the same state (here, zero state) to get the proper amplitude interference. As a result, we must prepare $|\boldsymbol{u}_n\rangle$ for the CU register inside the controlled unitary (U_n). Otherwise, we will get the wrong results at the end of the simulation. Consequently, the quantum circuit for multi-step simulation adopts a fractal structure, as shown in Fig. 16.

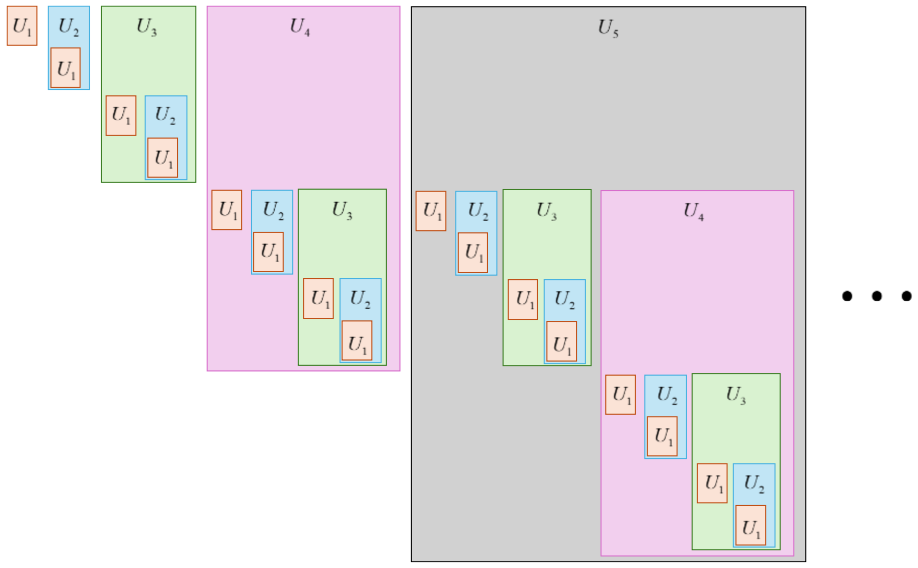


Fig. 16. Fractal structure of unitaries in the quantum circuit of the multi-step simulation of the Burgers' equation.

The red dotted line in Fig. 15 indicates that we are reusing qubits. If we reuse the qubits without any consideration, we will have wrong results since the good states are mixed with the bad ones. For instance, in Fig. 15, the CU register was reused in the second step of the simulation. From the QHP technique explained in Section 3.3, we know that if we post-select the results conditioned on getting zero on the copy of the main register (CU), we get an elementwise multiplication of two vectors in the main register (U). As a result, we need to add the CU register to the list of controlled qubits in the subtract gate to guarantee that we will apply the next operations on the right portion of the state vector.

3.8. The complexity of the quantum algorithm for space and time

The number of resources required to run the quantum algorithm for the Burgers' equation is growing logarithmically with space. In other words, we only need $\log N_x$ qubits for N_x spatial nodes. We need $2\log N_x$ qubits for encoding velocity and one copy of velocity for the first step. Also, we need 5 ancilla qubits to implement the quantum algorithm, including two qubits for LCU and three qubits for block-encoding. The number of qubits for the time register can be obtained as $\sum_{i=2}^{N_t}\left[\mathrm{Int}\left(\log_2(i)\right)+1\right]$. Without reusing the qubits, the number of copies of the main register increases exponentially with time. As a result, the number of qubits required for solving Burgers' equation without reusing qubits can be calculated as follows,

$$\left(2^{\mathrm{N_t}}\right)\log_2\left(\mathrm{N_x}\right)+5\mathrm{N_t}+\sum_{\mathrm{i}=2}^{\mathrm{N_t}}\left[\mathrm{Int}\left(\log_2\left(\mathrm{i}\right)\right)+1\right]. \tag{32}$$

However, we showed that we could reuse qubits by performing controlled subtraction (the controlled qubits should include the qubits we are going to reuse) from the time register. By reusing qubits from the previous steps, we could improve the exponential dependency of computational resources to linear dependency. As a result, the total number of qubits required for solving Burgers' equation on N_x points and for N_t steps can be calculated as follows,

$$\left(\mathrm{N_t}+1\right)\log_2\left(\mathrm{N_x}\right)+5\mathrm{N_t}+\sum_{\mathrm{i}=2}^{\mathrm{N_t}}\left[\mathrm{Int}\left(\log_2\left(\mathrm{i}\right)\right)+1\right] \tag{33}$$

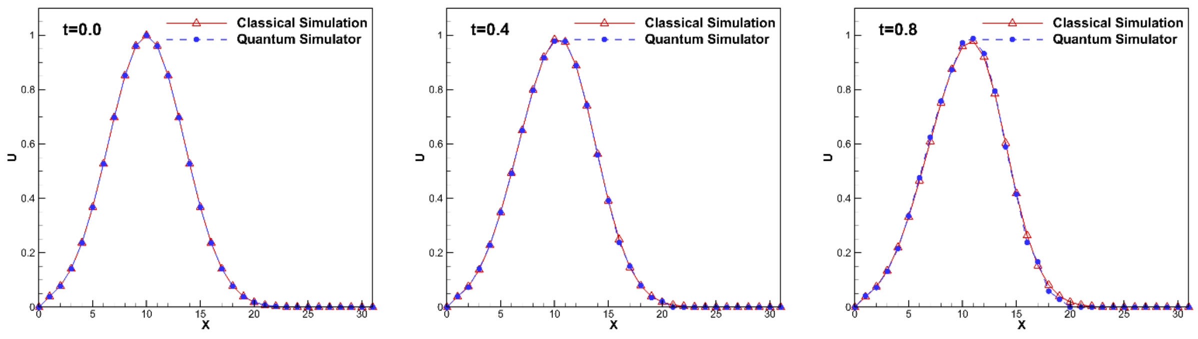
3.9. Limitations of executing quantum circuits on quantum computers and simulators

Despite the exponential improvement in the number of qubits in the current algorithm, we cannot execute this algorithm on quantum computers. The reason is that quantum computers in the noisy intermediate-scale quantum (NISQ) era have significant limitations that prevent the execution of complicated quantum algorithms. These systems are characterized by a relatively small number of qubits, which limits their computational capacity and makes them unable to handle the large-scale quantum circuits required for complex algorithms. Moreover, the qubits in NISQ devices are highly susceptible to errors and decoherence due to interactions with their environment, which results in a high error rate during computations.

As a result, we used quantum simulators to show that our quantum algorithm can solve Burgers' equation. However, the huge memory required to run quantum circuits on the quantum simulators limits the number of qubits for simulation. The memory required for a simulation with the N_{qubits} number of qubits is $2^{\max(0,\mathrm{N_{qubits}}-16)}\left[\mathrm{mb}\right]$. For instance, the memory requirement for running a quantum circuit with only 36 qubits is approximately 1000 GB. As a result, we could only execute the quantum circuit for 2 or 3 steps on the quantum simulator.

3.10. Numerical results of solving Burgers' equation using quantum computing

Fig. 17 compares the numerical results obtained from the classical simulation and the quantum simulator for the inviscid Burgers' equation. The computational domain was discretized to 32 nodes (Δx=1) and initialized using an exponential function of $u_0(x) = u(x,0) = e^{-\left(\frac{x}{5}-2\right)^2}$. The solutions were obtained for two time steps, with a constant time step of Δt=0.4.



(a) (b) (c)

Fig. 17. Comparison of numerical results of the inviscid Burgers' equation obtained with the quantum simulator compared with that from the classical simulation at (a) t=0, (b) t=0.4, (c) t=0.8.

We also executed the same quantum algorithm for a computational domain with 8 nodes to reduce the number of qubits, which enabled us to perform three-time steps of the inviscid Burgers' equation. The results of this simulation are shown in Fig. 18, where the computational domain was initialized using $u_0(x) = u(x,0) = e^{-(x-2)^2}$ and Δt was chosen to be 0.5.

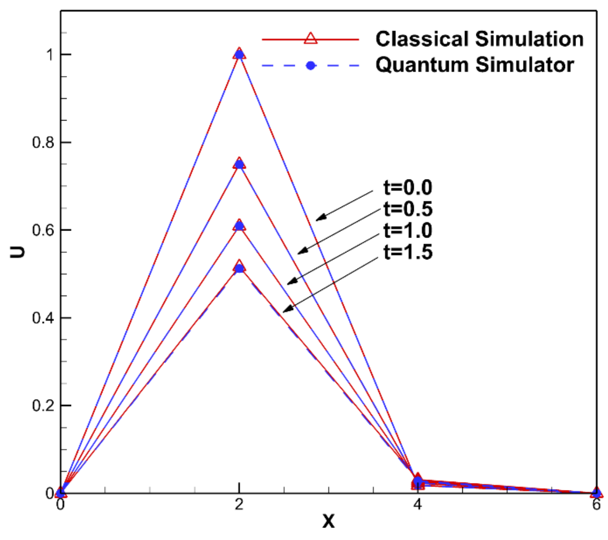


Fig. 18. Comparison of numerical results of the inviscid Burgers' equation obtained with the quantum simulator compared to the results from the classical simulation for *Nx*=8 and three-time steps.

The numerical results obtained by the quantum simulator were also compared with the classical simulation for Burgers' equation, shown in Fig. 19. For this problem, we discretized the domain to 16 nodes (Δ*x*=2), which can be encoded in four system qubits. The computational domain was initialized with $u_0(x) = u(x,0) = e^{-\left(\frac{x}{6}-2\right)^2}$ and the solutions were obtained for two time steps with Δt=0.5.

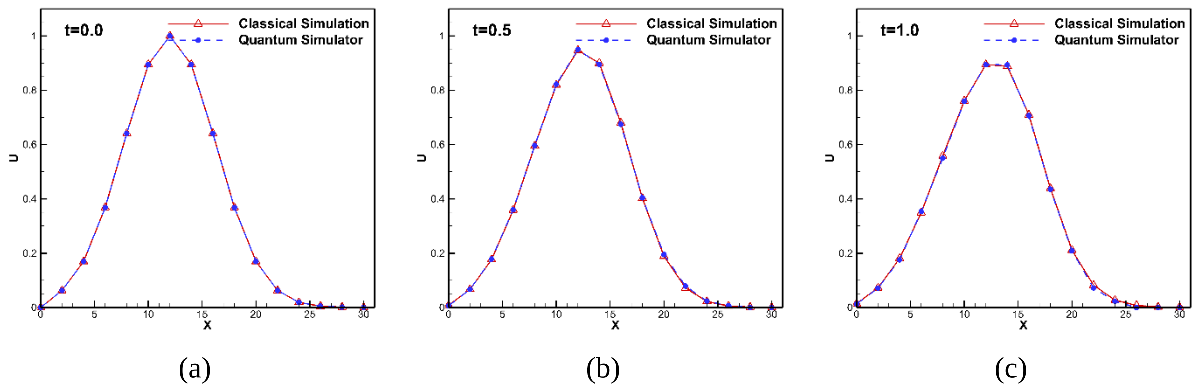


Fig. 19. Comparison of the numerical results of Burgers' equation obtained with the quantum simulator compared with the classical simulation at (a) t=0, (b) t=0.5, (c) t=1.0.

4. Conclusions

In this study, we first addressed several challenges that arise when trying to deal with nonlinearities using quantum algorithms, which are core problems related to solving nonlinear partial differential equations arising in fluid dynamics. After an extensive literature review and careful analysis, we established a strategy for designing an efficient quantum algorithm for nonlinear partial differential equations. We then developed a novel pure quantum algorithm for solving the nonlinear Burgers' equation as a first step to test the validity of such a strategy.

In the new method, we employed multiple copies of the state vector to calculate the nonlinear term encoded in amplitudes of the state vector. We showed that by reusing the qubits from the previous steps, the number of resources required for multi-step simulation scales linearly with time, which is a significant improvement over previous algorithms. Advanced quantum linear algebra techniques were employed to design a quantum circuit for the proposed quantum algorithm. The divide-and-conquer algorithm was used for quantum state preparation to initialize the computational domain. Non-unitary matrices of the first and second derivative operators were implemented using the block-encoding technique. The quantum Hadamard product technique was used to perform the non-unitary elementwise multiplication of two vectors. Finally, the linear combination of unitaries was employed to sum up all the terms that appear in the discretized Burgers' equation. The quantum circuit for Burgers' equation was executed on the quantum simulators. The results obtained with the quantum simulator showed excellent agreement with those obtained with classical simulations, demonstrating the feasibility of using the proposed quantum algorithm for solving the nonlinear Burgers' equation on future quantum computers.

The focus of this study was to develop a pure quantum algorithm to calculate the nonlinear terms in the governing equations of fluid dynamics. As a result, we used the upwind scheme for spatial discretization of the convective terms in a non-conservative form for the sake of simplicity. However, this approach is limited when shock waves form and the solution becomes discontinuous. For future work, we plan to develop a quantum algorithm for the Godunov scheme (challenging due to the structure of this scheme and the limitations imposed by quantum computing) to calculate the numerical fluxes in a finite volume framework. This will help us develop a quantum algorithm capable of solving shock structure problems using quantum computing. Additionally, given the current limitations of quantum computers—such as noise and a limited number of qubits—we aim to optimize our quantum circuits to solve Burgers' equation on real quantum computers.

Acknowledgment

This work was supported by the US AFOSR Grant No. FA2386-22-1-4052. D.A. also acknowledges partial support provided by COMPA grant No. RS-2024-00422330.

Data availability

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

References

[1] Sauro Succi, Wael Itani, Katepalli Sreenivasan, René Steijl, Quantum computing for fluids: Where do we stand?, Europhysics Letters 144 (2023) 10001.

[2] Alireza Nejadmalayeri, Alexei Vezolainen, Oleg V Vasilyev, Reynolds number scaling of coherent vortex simulation and stochastic coherent adaptive large eddy simulation, Physics of Fluids 25 (2013).

[3] Noah Linden, Ashley Montanaro, Changpeng Shao, Quantum vs. classical algorithms for solving the heat equation, Communications in Mathematical Physics 395 (2022) 601-641.

[4] YY Liu, Zhen Chen, Chang Shu, Siou-Chye Chew, Boo Cheong Khoo, Xiang Zhao, YD Cui, Application of a variational hybrid quantum-classical algorithm to heat conduction equation and analysis of time complexity, Physics of Fluids 34 (2022).

[5] C Sanavio, R Scatamacchia, C de Falco, S Succi, Three Carleman routes to the quantum simulation of classical fluids, Physics of Fluids 36 (2024).

[6] Wael Itani, Katepalli R Sreenivasan, Sauro Succi, Quantum algorithm for lattice Boltzmann (QALB) simulation of incompressible fluids with a nonlinear collision term, Physics of Fluids 36 (2024).

[7] Ljubomir Budinski, Quantum algorithm for the advection-diffusion equation simulated with the lattice Boltzmann method, Quantum Information Processing 20 (2021) 57.

[8] Budinski Ljubomir, Quantum algorithm for the Navier–Stokes equations by using the stream function-vorticity formulation and the lattice Boltzmann method, International Journal of Quantum Information 20 (2022) 2150039.

[9] Merel A Schalkers, Matthias Möller, On the importance of data encoding in quantum Boltzmann methods, arXiv preprint arXiv:2302.05305 (2023).

[10] Blaga N Todorova, René Steijl, Quantum algorithm for the collisionless Boltzmann equation, Journal of Computational Physics 409 (2020) 109347.

[11] Yudong Cao, Anargyros Papageorgiou, Iasonas Petras, Joseph Traub, Sabre Kais, Quantum algorithm and circuit design solving the Poisson equation, New Journal of Physics 15 (2013) 013021.

[12] Beimbet Daribayev, Aksultan Mukhanbet, Timur Imankulov, Implementation of the HHL algorithm for solving the Poisson equation on quantum simulators, Applied Sciences 13 (2023) 11491.

[13] Shengbin Wang, Zhimin Wang, Wendong Li, Lixin Fan, Zhiqiang Wei, Yongjian Gu, Quantum fast Poisson solver: the algorithm and complete and modular circuit design, Quantum Information Processing 19 (2020) 1-25.

[14] Peter Brearley, Sylvain Laizet, A quantum algorithm for solving the advection equation using Hamiltonian simulation, arXiv preprint arXiv:2312.09784 (2023).

[15] Julia Ingelmann, Sachin S Bharadwaj, Philipp Pfeffer, Katepalli R Sreenivasan, Jörg Schumacher, Two quantum algorithms for solving the one-dimensional advection-diffusion equation, arXiv preprint arXiv:2401.00326 (2023).

[16] Reuben Demirdjian, Daniel Gunlycke, Carolyn A Reynolds, James D Doyle, Sergio Tafur, Variational quantum solutions to the advection-diffusion equation for applications in fluid dynamics, Quantum Information Processing 21 (2022) 322.

[17] Jeffrey Yepez, An efficient quantum algorithm for the one-dimensional Burgers equation, arXiv preprint quant-ph/0210092 (2002).

[18] Frank Gaitan, Finding flows of a Navier–Stokes fluid through quantum computing. npj Quantum Information, 6 (1), 1-6, in, DOI, 2020.

[19] Frank Gaitan, Finding Solutions of the Navier-Stokes equations through quantum computing—Recent progress, a generalization, and next steps forward, Advanced Quantum Technologies 4 (2021) 2100055.

[20] Xiang Rao, Performance study of variational quantum linear solver with an improved ansatz for reservoir flow equations, Physics of Fluids 36 (2024).

[21] Michael A Nielsen, Isaac L Chuang, Quantum computation and quantum information, Cambridge University Press, 2010.

[22] Doyeol Ahn, Non-Markovian cost function for quantum error mitigation with Dirac Gamma matrices representation, Scientific Reports 13 (2023) 20069.

[23] J-H Bae, Paul M Alsing, Doyeol Ahn, Warner A Miller, Quantum circuit optimization using quantum Karnaugh map, Scientific Reports 10 (2020) 15651.

[24] Byeongyong Park, Doyeol Ahn, Reducing CNOT count in quantum Fourier transform for the linear nearest-neighbor architecture, Scientific Reports 13 (2023) 8638.

[25] Peter W Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35th annual symposium on foundations of computer science, IEEE, 1994.

[26] A Luis, J Peřina, Optimum phase-shift estimation and the quantum description of the phase difference, Physical Review A 54 (1996) 4564.

[27] Don Coppersmith, An approximate Fourier transform useful in quantum factoring, arXiv preprint quant-ph/0201067 (2002).

[28] Gilles Brassard, Peter Hoyer, Michele Mosca, Alain Tapp, Quantum amplitude amplification and estimation, Contemporary Mathematics 305 (2002) 53-74.

[29] Aram W Harrow, Avinatan Hassidim, Seth Lloyd, Quantum algorithm for linear systems of equations, Physical Review Letters 103 (2009) 150502.

[30] Dominic W Berry, High-order quantum algorithm for solving linear differential equations, Journal of Physics A: Mathematical and Theoretical 47 (2014) 105301.

[31] Dominic W Berry, Andrew M Childs, Aaron Ostrander, Guoming Wang, Quantum algorithm for linear differential equations with exponentially improved dependence on precision, Communications in Mathematical Physics 356 (2017) 1057-1081.

[32] Andrew M Childs, Jin-Peng Liu, Aaron Ostrander, High-precision quantum algorithms for partial differential equations, Quantum 5 (2021) 574.

[33] Sarah K Leyton, Tobias J Osborne, A quantum algorithm to solve nonlinear differential equations, arXiv preprint arXiv:0812.4423 (2008).

[34] Seth Lloyd, Giacomo De Palma, Can Gokler, Bobak Kiani, Zi-Wen Liu, Milad Marvian, Felix Tennie, Tim Palmer, Quantum algorithm for nonlinear differential equations, arXiv preprint arXiv:2011.06571 (2020).

[35] Zhaoyuan Meng, Yue Yang, Quantum computing of fluid dynamics using the hydrodynamic Schrödinger equation, Physical Review Research 5 (2023) 033182.

[36] Stefano Riva, Carolina Introini, Antonio Cammi, A finite element implementation of the incompressible Schrödinger flow method, Physics of Fluids 36 (2024).

[37] Bolesław Kacewicz, Almost optimal solution of initial-value problems by randomized and quantum algorithms, Journal of Complexity 22 (2006) 676-690.

[38] Furkan Oz, Rohit KSS Vuppala, Kursat Kara, Frank Gaitan, Solving Burgers' equation with quantum computing, Quantum Information Processing 21 (2022) 1-13.

[39] Biswajit Basu, Saravanan Gurusamy, Frank Gaitan, A quantum algorithm for computing dispersal of submarine volcanic tephra, Physics of Fluids 36 (2024).

[40] Jin-Peng Liu, Herman Øie Kolden, Hari K Krovi, Nuno F Loureiro, Konstantina Trivisa, Andrew M Childs, Efficient quantum algorithm for dissipative nonlinear differential equations, Proceedings of the National Academy of Sciences 118 (2021) e2026805118.

[41] René Steijl, Quantum circuit implementation of multi-dimensional non-linear lattice models, Applied Sciences 13 (2022) 529.

[42] Michael Lubasch, Jaewoo Joo, Pierre Moinier, Martin Kiffner, Dieter Jaksch, Variational quantum algorithms for nonlinear problems, Physical Review A 101 (2020) 010301.

[43] Rene Steijl, Quantum algorithms for nonlinear equations in fluid mechanics, Quantum Computing and Communications (2020).

[44] NTP Le, Hong Xiao, RS Myong, A triangular discontinuous Galerkin method for non-Newtonian implicit constitutive models of rarefied and microscale gases, Journal of Computational Physics 273 (2014) 160-184.

[45] RS Myong, Thermodynamically consistent hydrodynamic computational models for high-Knudsen-number gas flows, Physics of Fluids 11 (1999) 2788-2802.

[46] Jérémie Bec, Konstantin Khanin, Burgers turbulence, Physics Reports 447 (2007) 1-66.

[47] Omid Ejtehadi, Tapan K Mankodi, Ilyoup Sohn, Byoung Jae Kim, RS Myong, Gas-particle flows in a microscale shock tube and collection efficiency in the jet impingement on a permeable surface, Physics of Fluids 35 (2023).

[48] RS Myong, Analytical solutions of shock structure thickness and asymmetry in Navier–Stokes/Fourier framework, AIAA Journal 52 (2014) 1075-1081.

[49] Israel F Araujo, Daniel K Park, Francesco Petruccione, Adenilton J da Silva, A divide-and-conquer algorithm for quantum state preparation, Scientific Reports 11 (2021) 6329.

[50] Daan Camps, Lin Lin, Roel Van Beeumen, Chao Yang, Explicit quantum circuits for block encodings of certain sparse matrices, arXiv preprint arXiv:2203.10236 (2022).

[51] Guang Hao Low, Nathan Wiebe, Hamiltonian simulation in the interaction picture, arXiv preprint arXiv:1805.00675 (2018).