

Applicability of solving the one- and two-dimensional Poisson equations with the quantum Harrow-Hassidim-Lloyd algorithm

Luca Ghafourpour^{1,2,*} and Sylvain Laizet²

¹Department of Mathematics, *ETH Zürich*, Zürich, Switzerland

²Department of Aeronautics, *Imperial College London*, London, United Kingdom



(Received 8 October 2024; revised 22 May 2025; accepted 1 July 2025; published 13 August 2025)

This paper assesses the numerical accuracy of the Harrow-Hassidim-Lloyd (HHL) algorithm in solving a finite-difference approximation of the Poisson equation, with the Thomas algorithm used as a comparative benchmark. In the one-dimensional setting with homogeneous boundary conditions, the HHL algorithm exhibits numerical accuracy comparable to the Thomas algorithm; however, the HHL algorithm exhibits increased sensitivity to variations in input parameters for simulations with nonhomogeneous boundary conditions. While the current implementation of the HHL algorithm is limited to solving the one-dimensional Poisson equation, we propose a hybrid framework to extend the applicability of the HHL algorithm to solving the two-dimensional Poisson equation. Our results show that although both algorithms perform similarly for small systems of 8×8 mesh nodes, the HHL algorithm shows increased input-parameter sensitivity for systems of 16×16 mesh nodes, indicating that further development is needed to scale the hybrid approach effectively.

DOI: [10.1103/ymjf-6n8j](https://doi.org/10.1103/ymjf-6n8j)

I. INTRODUCTION

The Navier-Stokes equations are a system of partial differential equations (PDEs) that describe the motion of unsteady and viscous fluids, such as liquids and gases, by accounting for velocity, pressure, density, and external forces. Solving the incompressible Navier-Stokes equations requires first solving for the velocity field, followed by solving the Poisson equation for the pressure field. Notably, the Poisson equation contributes nearly 80% of the computational cost in solving the incompressible Navier-Stokes equations [1]. Traditional numerical methods, such as finite-difference, finite-element, and spectral methods, suffer from the curse of dimensionality. This challenge is compounded by the gradual departure from Moore's Law in recent years, which has slowed improvements in classical computational power [2]. While many well-known PDE solvers include compatibility with graphics processing units (GPUs) to leverage GPU acceleration as a short-term solution, quantum processing units (QPUs) offer a promising long-term alternative [3–5].

Quantum algorithms are a class of algorithms specific to QPUs that exploit the principles of superposition and

entanglement to offer improved scaling properties compared to classical algorithms [6]. Given their potential to significantly reduce computational costs, this paper explores the applicability of quantum algorithms to solving the Poisson equation.

The standard approach for solving linear PDEs involves discretizing them onto a mesh using techniques such as finite-difference methods. These methods convert differential equations into a linear system of algebraic equations, which approximate the solution to the PDE. In classical linear system problems, the goal is to find a vector $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$ for a given matrix A and vector $\mathbf{b}$. In contrast, a quantum linear system problem (QLSP) aims to find $|x\rangle$ such that $A|x\rangle = |b\rangle$. Thus, the key difference is that the output of a QLSP is a quantum state encoding the solution $\mathbf{x}$ rather than the solution vector itself.

The first breakthrough in solving a QLSP was made by Harrow, Hassidim, and Lloyd (HHL) in 2008 [7]. They presented a quantum linear system algorithm (QLSA) that solved the QLSP of a sparse $N \times N$ Hermitian matrix A with a resource complexity that scales as $\mathcal{O}(\kappa^2 \log_2(N)/\epsilon)$, where κ is the condition number of A and ϵ is an upper bound on the desired accuracy. The HHL algorithm achieved an exponential improvement in the dependency on the system size N compared to classical linear system algorithms (CLSAs), which scale at best as $\mathcal{O}(N)$; however, a key limitation of the HHL algorithm is its dependence on the quantum phase estimation (QPE) algorithm, which scales as $\mathcal{O}(1/\epsilon)$. The QPE algorithm, along with a controlled rotation subroutine, are used within

*Contact author: lghafourpour@ethz.ch

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

the HHL algorithm to compute A^{-1} . In 2017, Childs, Kothari, and Somma addressed this limitation by presenting a QLSA that omits the QPE algorithm [8]. Instead, it represents the matrix inverse as a linear combination of unitary matrices. The resource complexity of this approach scales as $\mathcal{O}(\kappa \log_2(N) \text{polylog}_2(1/\epsilon))$, achieving an exponential improvement in precision and a linear dependence on κ . Further advancements were made by Subasi, Somma, and Orsucci in 2018, who proposed a QLSA that reduced the scaling to $\mathcal{O}(\kappa \log_2(N)/\epsilon)$ [9]. Unlike previous methods relying on gate-based quantum computing models, their approach uses adiabatic quantum computing (AQC). In AQC, computations are performed by evolving a Hamiltonian that represents a physical system from an initial state, which encodes the problem, to a final state, which encodes the solution [10].

The previously introduced algorithms are classed as fault tolerant and assume noiseless qubits; however, the current era of quantum computing is characterized by significant noise from sources such as environmental interactions, imprecise control of qubits, and errors in quantum gate operations. The noise level also increases with the number of qubits, which has restricted the use of fault-tolerant QLSAs for solving large QLSPs. Despite this, current efforts have been exploring ways to use existing hardware during what has been called the noisy intermediate-scale quantum (NISQ) era [6]. This has given rise to a new class of algorithms called variational quantum algorithms (VQAs), which have lower hardware requirements and greater resilience to quantum noise [11]. The framework of a VQA is similar to that of machine learning, where a parameterized quantum circuit with adjustable parameters is tuned using classical optimization techniques to minimize a loss function. By taking measurements from the quantum circuit, the optimizer iteratively adjusts the parameters until the algorithm converges to a solution. Current efforts are focused on defining the framework for VQAs, testing their resilience to noise when solving large QLSPs, and applying VQAs to solving PDEs such as the quantum harmonic oscillator [12] and the time-independent one-dimensional nonlinear Schrödinger equation [13]. Impressively, some implementations have successfully solved QLSPs of size $2^{50} \times 2^{50}$ [14].

Since the publication of the HHL algorithm, efforts have been made to optimize the design of the HHL circuit specifically for solving the QLSP that arises from the finite-difference approximation of the one-dimensional Poisson equation [15,16]; however, these earlier works primarily focused on algorithmic design and complexity estimation, and they lacked experimental results to assess their numerical accuracy. While theoretical advancements in QLSAs have led to proposed strategies for applying these algorithms to solve the Poisson equation [17,18], these proposals similarly lack numerical simulations. In contrast,

the development of VQAs for the Poisson equation has led to successful simulations on both quantum simulators and real quantum hardware [19–21]. As a result, there remains a gap in understanding the numerical performance of fault-tolerant QLSAs in solving the Poisson equation.

The preceding paragraphs highlight the significant progress made in the theoretical development of quantum algorithms for solving PDEs; however, much of this progress remains largely theoretical, and there is no definitive strategy for translating these concepts into quantum circuit designs, thereby limiting their practical applicability. While references have been made to authors who have proposed strategies to apply their fault-tolerant QLSAs to solve the Poisson equation, these proposals also lack numerical simulations of their algorithms. Although Ref. [22] also explores using the HHL algorithm to solve a finite-difference approximation of the Poisson equation, that study is limited to a single, unspecified source function and Dirichlet boundary conditions. Additionally, the discussion focuses on computational and execution-time comparisons across different quantum simulators, providing no new insights into algorithmic behavior. Hence, the present study aims to address this gap through extensive numerical simulations with the HHL algorithm to analyze its applicability in solving a finite-difference approximation of the Poisson equation with different source functions, boundary conditions, and mesh refinements. In this work, we use a QPE algorithm that assumes A is a Toeplitz symmetric tridiagonal (TST) matrix, limiting its application to the one-dimensional Poisson equation. To overcome this limitation, we introduce a hybrid QLSA, which uses a line-Jacobi scheme to extend the algorithm's applicability to the two-dimensional Poisson equation. To the best of our knowledge, this is the first work to leverage line schemes in extending the capability of QLSAs for solving higher-dimensional PDEs.

To comprehensively address this research question, the remainder of this paper is structured as follows. Section II provides an overview of the HHL algorithm used in this research, highlighting any specific deviations from the original formulation made in this work. Section III presents the Poisson problem in one and two dimensions, explains how our QLSA extends to the two-dimensional case, and concludes with a comparison of the resource complexities of both formulations. Section IV discusses the results from the numerical simulations, including a sensitivity study of the input parameters. The paper concludes with suggestions for future research directions and potential improvements.

II. THE HHL ALGORITHM

The QLSA used in this study is the HHL algorithm [7], implemented using the circuit design proposed by Cao *et al.* [23], as illustrated in Fig. 1. The circuit consists of

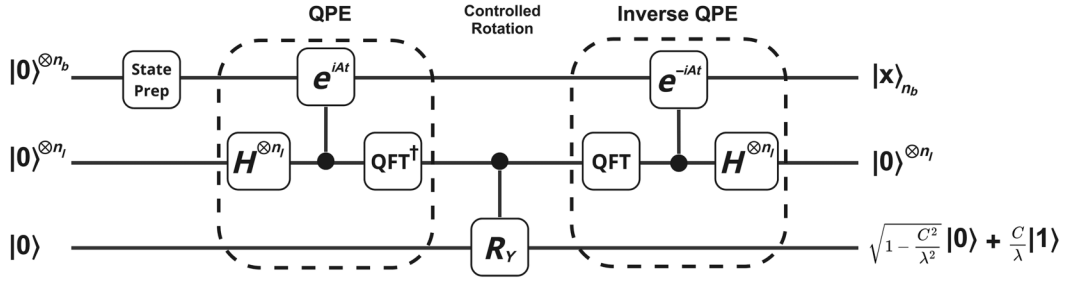


FIG. 1. Quantum circuit design for the HHL algorithm introduced in Ref. [23].

three qubit registers, each assumed noiseless: n_b qubits in the input b -register, n_l qubits in the clock l -register, and a single ancillary qubit in the a -register. The circuit is composed of five subroutines: state preparation, QPE, controlled rotation, inverse QPE, and measurement. Section II A provides a detailed explanation of the function of each subroutine.

A. Overview of the HHL algorithm

Let $A \in \mathbb{C}^{N \times N}$ be a Hermitian matrix and $|b\rangle, |x\rangle \in (\mathbb{C}^2)^{\otimes N}$. The HHL algorithm finds the quantum state $|x\rangle$ that encodes the solution to the QLSP $A|x\rangle = |b\rangle$, where $|x\rangle$ is proportional to the solution vector $\mathbf{x}$. The algorithm is derived using the following line of reasoning. Since A is assumed to be Hermitian, it admits an eigendecomposition of the form

$$A = \sum_{i=0}^{N-1} \lambda_i |u_i\rangle \langle u_i|, \quad \lambda_i \in \mathbb{R},$$

where $\{|u_i\rangle\}$ is an independent orthonormal eigenbasis of A and λ_i are the respective eigenvalues. Moreover, since any arbitrary vector ϕ can be expressed in the eigenspace of A as $\phi = \sum_{i=0}^{N-1} \phi_i |u_i\rangle$,

$$|x\rangle = A^{-1}|b\rangle = \sum_{i=0}^{N-1} \frac{1}{\lambda_i} |u_i\rangle \langle u_i| \sum_{j=0}^{N-1} \beta_j |u_j\rangle = \sum_{i=0}^{N-1} \frac{\beta_i}{\lambda_i} |u_i\rangle.$$

Initially, all registers of the HHL circuit are initialized to the $|0\rangle$ state. The algorithm proceeds as follows:

(1) Encode $\mathbf{b}$ in a quantum state $|b\rangle$ and load it into the b -register. This achieves the mapping

$$|0\rangle|0\rangle^{\otimes n_l}|0\rangle^{\otimes n_b} \mapsto |0\rangle|0\rangle^{\otimes n_l}|b\rangle.$$

(2) Execute the QPE algorithm with unitary operator $U = e^{iAt}$, control register l , and target register b . This encodes the eigenvalues of A in the l -register and achieves

the mapping

$$\begin{aligned} |0\rangle \text{QPE}(e^{iAt}, |0\rangle^{\otimes n_l}|b\rangle) &\mapsto |0\rangle|\lambda\rangle|b\rangle \\ &= |0\rangle \sum_{i=0}^{N-1} b_i |\lambda_i\rangle_{n_l} |u_i\rangle_{n_b}. \end{aligned}$$

(3) Perform controlled rotations on the ancillary qubit to invert the encoded eigenvalues. This achieves the mapping

$$\begin{aligned} |0\rangle \sum_{i=0}^{N-1} b_i |\lambda_i\rangle |u_i\rangle &\mapsto \sum_{i=0}^{N-1} b_i \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right) |\lambda_i\rangle_{n_l} |u_i\rangle_{n_b}, \end{aligned}$$

where C is a normalization constant.

(4) Apply the inverse QPE algorithm with unitary $U = e^{-iAt}$ to transform the encoded eigenvalues in the l -register back to the $|0\rangle$ state. This achieves the mapping

$$\begin{aligned} \sum_{i=0}^{N-1} b_i \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right) |\lambda_i\rangle_{n_l} |u_i\rangle_{n_b} &\mapsto \sum_{i=0}^{N-1} b_i \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right) |0\rangle_{n_l} |u_i\rangle_{n_b}. \end{aligned}$$

(5) Measure the ancillary qubit in the computational basis. If the measurement outcome is $|1\rangle$, the register is in the quantum state

$$\begin{aligned} &\left(\sqrt{\frac{1}{\sum_{i=0}^{N-1} |b_i|^2 / |\lambda_i|^2}} \right) |1\rangle |0\rangle_{n_l} \sum_{i=0}^{N-1} \frac{b_i}{\lambda_i} |u_i\rangle_{n_b} \\ &= \left(\sqrt{\frac{1}{\sum_{i=0}^{N-1} |b_i|^2 / |\lambda_i|^2}} \right) |1\rangle |0\rangle_{n_l} |x\rangle_{n_b}. \end{aligned}$$

The solution to the QLSP is encoded in the quantum state $|x\rangle$, stored within the b -register.

The following subsections introduce the special treatments undertaken in this work. Since the Hamiltonian simulation and controlled rotation algorithms have been used in previous works, detailed discussions of these algorithms have been moved to Appendices A and C. For the controlled rotation algorithm, our method uses uniformly controlled rotation gates, $R_Y(2\theta_j)$, with rotation angles $\theta_j = \arcsin(C/\lambda_j)$. Additionally, as some understanding of the Hamiltonian simulation algorithm is necessary for interpreting the simulation results in Sec. IV, a high-level overview is provided in Sec. II C.

B. State preparation

The first step of the HHL algorithm is to encode the vector $\mathbf{b} \in \mathbb{R}^N$ into a quantum state stored in the b -register. This process relies on a series of quantum gates to map the classical data into a quantum state in a Hilbert space. While a plethora of quantum state encoding techniques exist, basis encoding, angle encoding, and amplitude encoding are the most commonly used. Table I summarizes the number of qubits required to encode $\mathbf{b}$ using each of the aforementioned techniques.

In this work, we use amplitude encoding to prepare the b -register as it requires the fewest qubits to encode $\mathbf{b}$. This technique encodes classical information as the amplitudes of the computational basis states. Given a vector $\mathbf{b} \in \mathbb{R}^N$ to be encoded, it is first normalized as $\hat{\mathbf{b}} = \mathbf{b}/|\mathbf{b}|$ and subsequently encoded as the amplitudes of an n -qubit quantum state

$$|b\rangle = \sum_{i=1}^N \hat{b}_i |i\rangle,$$

where $N = 2^n$ and $|i\rangle$ represents the i th computational basis state of $\mathbb{C}^n$.

Amplitude encoding requires special consideration when N is not a power of 2, as an n -qubit system produces exactly 2^n amplitudes. In cases where N is not a power of 2, the quantum state must be padded with additional auxiliary amplitudes, ensuring that it can be encoded by an n -qubit system.

C. Hamiltonian simulation

At the core of the QPE algorithm lies the simulation of the time evolution operator e^{iAt} , referred to as Hamiltonian

TABLE I. Numbers of qubits required to encode a vector of length N using basis, amplitude, and angle encoding.

Encoding technique	Number of qubits
Basis	$\sum_{i=1}^N \lceil \log_2(b_i) \rceil$
Amplitude	$\lceil \log_2(N) \rceil$
Angle	N

simulation. This process models the dynamics of a quantum state within a closed quantum system. In this work, we use the circuit developed by Vazquez *et al.* [24], optimized for TST system matrices. This focus is motivated by the fact that TST matrices are commonly encountered in various engineering applications; however, for general tridiagonal matrices that are not necessarily symmetric or Toeplitz, alternative Hamiltonian simulation algorithms offer greater flexibility [25]. The subsequent discussion provides a high-level overview of the Hamiltonian simulation algorithm for TST matrices as presented in Ref. [24]. For a comprehensive explanation, including circuit diagrams, we refer the reader to Appendix A.

Consider a general $N \times N$ TST matrix of the form

$$A = \begin{pmatrix} a & b & & & \\ b & a & b & & \\ & \ddots & \ddots & \ddots & \\ & & b & a & b \\ & & & b & a \end{pmatrix}, \quad a, b \in \mathbb{R}.$$

To compute e^{iAt} , Ref. [24] decomposes A into k -local Hamiltonians such that $A = \sum_i A_i$, where each A_i is a 1-sparse matrix. A valid decomposition of A is given by

$$A_1 = \begin{pmatrix} a & 0 & & 0 \\ 0 & \ddots & \ddots & \\ & \ddots & & 0 \\ 0 & & 0 & a \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & b & & \\ b & 0 & & \\ & & \ddots & \\ & & & 0 & b \\ & & & b & 0 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} 0 & & & & \\ & 0 & b & & \\ & b & 0 & & \\ & & & \ddots & \\ & & & & 0 & b \\ & & & & b & 0 \\ & & & & & 0 \end{pmatrix}.$$

The authors combine the contributions of each $e^{iA_i t}$ using a Lie-Trotter-Suzuki approximation,

$$e^{iAt} = e^{i(A_1+A_2+A_3)t} = \lim_{m \rightarrow \infty} e^{iA_1 t/m} e^{-iA_2 t/2m} (e^{iA_2 t/m} e^{iA_3 t/m})^m e^{iA_2 t/2m}, \quad (1)$$

where each operator of Eq. (1) can be simulated using a simple quantum circuit, as detailed in Appendix A.

The QPE algorithm computes the controlled e^{iAt} operator using a sequence of controlled $e^{iA(t/2^j)}$ operators, each approximated using the Lie-Trotter-Suzuki approximation in Eq. (1), where $j = 0, 1, \dots, n_l - 1$ denotes the index of the control qubit in the l -register. For each $e^{iA(t/2^j)}$ operator,

a finite number of Trotter steps m_j is used to approximate the evolution. To bound the error in computing $e^{iA(tj)}$ to a specified ϵ , the number of Trotter steps should be $m_j = \mathcal{O}(\sqrt{2j}/\epsilon)$.

D. Postselection algorithm

The final step of the HHL algorithm is to extract information about the solution to the QLSP, which is encoded in the amplitudes of a quantum state. To achieve this, a measurement function $F(x) = \langle x|M|x \rangle$ is used, which represents the expectation of a known measurable Hermitian operator M acting on x . While this approach preserves the potential for exponential speedup over CLSAs, it does not allow for a direct numerical comparison between the solutions produced by classical and quantum algorithms. This limitation arises because the measurement function provides information about the solution rather than the full solution vector itself.

To facilitate direct numerical comparisons between the solutions generated by classical and quantum algorithms, we conduct state-vector simulations of the HHL circuit. State-vector simulations represent the full quantum state as a vector and track its evolution and computations at each step of the algorithm. This approach provides a detailed insight into the algorithm's behavior and enables the extraction of each element of the solution vector; however, the computational resources required to store and manipulate this state vector scale exponentially with the number of qubits, thereby diminishing any observable speedup from using a QLSA. Nonetheless, the present work prioritizes insight into algorithmic behavior over considerations of computational complexity and speedup advantages. The following details our procedure for extracting the solution to the QLSA.

Consider the HHL circuit in Fig. 1, which uses a total of M qubits. Such a circuit manipulates a quantum-state vector $|\psi\rangle = |q_a q_{l_{n_l-1}} \dots q_{l_0} q_{b_{n_b-1}} \dots q_{b_0}\rangle \in (\mathbb{C}^2)^{\otimes M}$. If the ancillary qubit is measured in the $|1\rangle$ state following the inverse QPE algorithm, the final quantum state $|\psi\rangle = |1, 0 \dots 0, q_{b_{n_b-1}} \dots q_{b_0}\rangle$.

The state $|\psi\rangle$ is expressed in the standard basis of $(\mathbb{C}^2)^{\otimes M}$, with each basis vector indexed by an M -bit binary string, resulting in a total of 2^M basis vectors. Since the states of the ancillary and l -register qubits are known, $|\psi\rangle$ has nonzero amplitudes only in the 2^{n_b} basis vectors whose binary string has the first $n_a + n_l$ bits fixed as $10 \dots 0$. The solution $|x\rangle$ is then recovered by extracting these specific amplitudes.

The solution state $|x\rangle$ obtained from the HHL algorithm is a quantum state proportional to the true solution vector $\mathbf{x}$. This can be expressed using the proportionality equation $c|x\rangle = \mathbf{x}$ for some constant c . Given that A , $|x\rangle$, and $\mathbf{b}$ are known, c is calculated using the relationship

TABLE II. Qubit requirements for each register in the HHL algorithm circuit.

Register	Number of qubits
b -register	$\log_2(N)$
l -register	$\max(\log_2(N) + 1, \lceil \log_2(\kappa(A) + 1) \rceil) (+1 \text{ for sign})$
MCMT-register	$\log_2(N) - 1$
a -register	1

$cA|x\rangle = \mathbf{b}$. Once c is determined, $\mathbf{x}$ is computed using the proportionality equation.

E. Circuit complexity and qubit requirements

We briefly summarize the quantum resources required to apply the HHL algorithm to the QLSP, where A is assumed TST. Specifically, we quantify the qubit requirements for each register and provide asymptotic gate counts for the primary subroutines as functions of the number of qubits and the target precision ϵ .

To encode $\mathbf{b} \in \mathbb{R}^N$ using amplitude encoding, as detailed in Sec. II B, the b -register requires $\log_2(N)$ qubits. Here, N is chosen to be a power of 2 to avoid padding the quantum state with auxiliary amplitudes.

The l -register must be able to represent the full eigenspectrum of the matrix A using n_l binary bits. After normalizing by the smallest eigenvalue, the eigenspectrum spans integer levels 0 to $\kappa(A)$, where $\kappa(A) = |\lambda_{\max}(A)|/|\lambda_{\min}(A)|$ denotes the condition number of A . To uniquely encode all $\kappa(A) + 1$ distinct levels, a minimum of $\lceil \log_2(\kappa(A) + 1) \rceil$ qubits are needed; however, to ensure compatibility with the Hamiltonian simulation algorithm, the width of the l -register must be at least $n_b + 1$. In practice, it is thus taken to be $\max(n_b + 1, \lceil \log_2(\kappa(A) + 1) \rceil)$. An additional sign qubit is appended if any eigenvalues of A are known to be negative.

The Hamiltonian simulation algorithm used in this work [24] uses a series of multicontrolled multitarget (MCMT) gates. An efficient construction of each MCMT gate necessitates an auxiliary register, referred to here as the MCMT-register, with a width of $n_b - 1$ qubits. This register is initialized to and subsequently returned in the state $|0\rangle^{\otimes(n_b-1)}$.

Finally, a single ancilla qubit, stored in the a -register, is used to perform the controlled rotation algorithm. A summary of the qubit requirements for each register in the HHL circuit is provided in Table II.

Having established the qubit requirements for the HHL algorithm, we next examine the circuit complexity. Table III summarizes the asymptotic gate complexity of its primary subroutines as a function of the number of qubits and the target precision ϵ of the Hamiltonian simulation algorithm.

TABLE III. Asymptotic gate scaling of primary HHL subroutines.

Subroutine	Complexity
Quantum Fourier transform [26]	$\mathcal{O}(n_l^2)$
Hamiltonian simulation	$\mathcal{O}(\sqrt{2^{n_l}}/\epsilon n_b^2)$
Controlled rotation [27]	$\mathcal{O}(2^{n_l})$

A proof of the resource complexity for the Hamiltonian simulation algorithm can be found in Appendix B 3.

III. APPLICATION TO THE POISSON EQUATION

Let Ω denote a d -dimensional finite domain with boundary $\partial\Omega$. The Poisson equation for u with Dirichlet boundary conditions is given by

$$\begin{aligned} \nabla^2 u(\mathbf{x}) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}) &= g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \end{aligned} \quad (2)$$

where f is a smooth source function. Since the Poisson equation is elliptic, boundary conditions must be specified for the entire $\partial\Omega$ to ensure that the problem is well-posed.

The following subsections detail our discretization approach for the one- and two-dimensional Poisson equations. We then introduce the technique used to extend the HHL algorithm to solve the two-dimensional Poisson equation. As the system matrices arising from the one- and two-dimensional discretizations differ, the section concludes with a summary of the corresponding resource complexities for each problem.

A. Problem definition in one dimension

Consider the one-dimensional Poisson equation on a unit line,

$$\begin{aligned} \frac{d^2 u}{dx^2} &= f(x), \quad x \in [0, 1], \\ u(0) &= \alpha, \quad u(1) = \beta, \end{aligned} \quad (3)$$

where $\alpha, \beta \in \mathbb{R}$. To solve Eq. (3) using finite differences, the domain is discretized into a uniform mesh of equally spaced nodes with spacing Δx . The derivative is approximated using a second-order, centered-in-space finite-difference scheme, transforming the one-dimensional Poisson equation into the following linear system of equations:

$$\begin{aligned} u_0 &= \alpha, \\ u_{i+1} - 2u_i + u_{i-1} &= (\Delta x^2)f_i, \quad i = 1, 2, \dots, N-2 \\ u_{N-1} &= \beta, \end{aligned} \quad (4)$$

where the evaluation of a variable at a location x_i is expressed with a subscript i . Since the system matrix of

Eq. (4) is not TST, it is incompatible with the Hamiltonian simulation algorithm of Ref. [24]. By incorporating the boundary conditions into the system of equations, the resulting system matrix becomes TST and thus falls within the framework of the proposed QLSA:

$$\begin{aligned} u_2 - 2u_1 &= (\Delta x^2)f_1 - \alpha, \\ u_{i+1} - 2u_i + u_{i-1} &= (\Delta x^2)f_i, \quad i = 2, 3, \dots, N-3, \\ -2u_{N-2} + u_{N-3} &= (\Delta x^2)f_{N-2} - \beta. \end{aligned} \quad (5)$$

B. Extension to two dimensions

In two dimensions, the Poisson equation over the unit square is defined as

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= f(x, y), \quad (x, y) \in [0, 1]^2, \\ u(x, 0) &= g_{y_0}(x), \quad u(x, 1) = g_{y_1}(x), \\ u(0, y) &= g_{x_0}(y), \quad u(1, y) = g_{x_1}(y), \end{aligned} \quad (6)$$

where each boundary condition function $g : [0, 1] \rightarrow \mathbb{R}$. To solve Eq. (6), we adopt a similar discretization strategy as in the one-dimensional setting. The computational domain is discretized into a uniform mesh of equally spaced nodes with spacing $\Delta x = \Delta y = \Delta h$. The spatial derivatives in the x and y directions are approximated using second-order, centered-in-space finite-difference schemes, leading to the following linear system of equations:

$$\begin{aligned} u_{i+1,j} + u_{i-1,j} - 4u_{i,j} + u_{i,j+1} + u_{i,j-1} &= (\Delta h^2)f_{i,j}, \\ i, j &= 0, \dots, N-1. \end{aligned} \quad (7)$$

The resulting system is characterized by a block-banded matrix with bandwidth N , given by

$$A = \frac{1}{(\Delta h^2)} \begin{bmatrix} T & I_N & 0 & \cdots & 0 \\ I_N & T & I_N & \ddots & \vdots \\ 0 & I_N & T & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & I_N \\ 0 & \cdots & 0 & I_N & T \end{bmatrix}, \quad (8)$$

where T is an $N \times N$ TST matrix with main-diagonal value -4 and off-diagonal value 1 , and I_N is the $N \times N$ identity matrix; however, this block-banded structure is incompatible with the Hamiltonian simulation framework of Ref. [24], which requires the system matrix to be TST.

In classical numerical analysis, iterative methods are generally preferred for solving large, sparse linear systems such as Eq. (7). A common iterative approach is the point-Jacobi scheme, where the solution at each grid point is

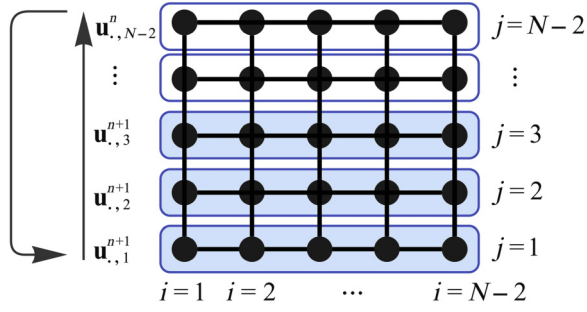


FIG. 2. Stencil of the line-Jacobi iterative scheme on a two-dimensional mesh.

updated based on the values of its neighbors from the previous iteration; however, applying the point-Jacobi scheme to the two-dimensional Poisson equation does not result in a TST system matrix and, as such, is incompatible with the Hamiltonian simulation framework of Ref. [24].

To overcome this, we use a line-Jacobi scheme. In this approach, instead of updating individual mesh nodes, we simultaneously update all nodes along a specific line based on the values from the previous iteration. By choosing to update along rows, fixing the j index, the update rule for the j th row becomes

$$u_{i+1,j}^{n+1} - 4u_{i,j}^{n+1} + u_{i-1,j}^{n+1} = (\Delta h^2)f_{i,j} - (u_{i,j-1}^n + u_{i,j+1}^n),$$

$$i = 0, 1, \dots, N - 1. \quad (9)$$

Fixing a row j and applying Dirichlet boundary conditions at $i = 0$ and $i = N - 1$ results in a linear system of equations for the updated solution vector along that row, $\mathbf{u}_j^{n+1}$, with a TST system matrix. This realization makes the approach compatible with the Hamiltonian simulation framework of Ref. [24].

Consequently, the original two-dimensional problem can be decomposed into a sequence of $N - 2$ one-dimensional subproblems. Each subproblem corresponds to updating a single row j using values from adjacent rows $j - 1$ and $j + 1$ from the previous iteration. This process is illustrated in Fig. 2, where each highlighted region represents the solution of a TST linear system of equations. To obtain the updated solution over the entire mesh, $\mathbf{u}^{n+1}$, we sequentially solve the system defined by Eq. (9) for each interior row $j = 1, 2, \dots, N - 2$, while rows $j = 0$ and $N - 1$ enforce Dirichlet boundary conditions. This full sweep constitutes one iteration of the line-Jacobi method, which is repeated until the solution converges to the desired error tolerance.

The line-Jacobi scheme retains the parallelization capabilities of the point-Jacobi scheme while providing improved convergence rates and better error damping for high-wavenumber solution components. A proof of this claim is provided in Appendix D.

TABLE IV. Qubit requirements for the HHL circuit applied to the one-dimensional Poisson equation.

Register	Number of qubits
b -register	$\log_2(N)$
l -register	$\lceil \log_2(\kappa(A) + 1) \rceil$ (+1 for sign)
MCMT-register	$\log_2(N) - 1$
a -register	1

The proposed extension should be considered a hybrid QLSA since it integrates classical methodologies into the *quantum world*. It relies on the strong assumption that the solution obtained from the HHL algorithm can be extracted as a classical vector and subsequently used to reinitialize the quantum circuit with the intermediate solution at each iterative step.

C. Circuit complexity and qubit requirements to the one-dimensional and two-dimensional Poisson equation

In Sec. II E, we analyzed the complexity and quantum resource requirements of the HHL algorithm for a general $N \times N$ linear system with a TST system matrix. While the system matrices arising from the finite-difference discretization of the one- and two-dimensional Poisson equations share similarities, notably differing in their main diagonal elements, these distinctions lead to variations in the algorithm's resource requirements. The subsequent analysis focuses on the one- and two-dimensional Poisson problems. The scaling derivations for the condition number and l -register width are provided in Appendices B 1 and B 2, respectively.

For the system matrix corresponding to the one-dimensional Poisson problem, as given in Eq. (5), we have $\kappa(A) = \mathcal{O}(N^2)$. The resulting qubit requirements are summarized in Table IV, and the asymptotic gate complexity of the primary HHL subroutines is provided in Table V.

For the system matrix arising from the two-dimensional Poisson problem, as shown in Eq. (9), we have $\kappa(A) = \mathcal{O}(1)$, with $\lim_{N \rightarrow \infty} \kappa(A) = 3^-$. The corresponding qubit requirements are presented in Table VI, and the asymptotic gate complexity of the primary HHL subroutines is shown in Table VII.

TABLE V. Asymptotic gate scaling of primary HHL subroutines for the one-dimensional Poisson equation.

Subroutine	Complexity
Quantum Fourier transform	$\mathcal{O}(\log_2^2(N))$
Hamiltonian simulation	$\mathcal{O}\left(\frac{N}{\sqrt{\epsilon}} \log_2^2(N)\right)$
Controlled rotation	$\mathcal{O}(N^2)$

TABLE VI. Qubit requirements for the HHL circuit applied to the two-dimensional Poisson equation.

Register	Number of qubits
b -register	$\log_2(N)$
l -register	$\log_2(N) + 1$ (+1 for sign)
MCMT-register	$\log_2(N) - 1$
a -register	1

Comparing the two settings, the HHL algorithm exhibits less favorable quantum resource scaling for the one-dimensional Poisson problem than for the two-dimensional case. This arises from differences in how the condition number of each system matrix scales with problem size. Consequently, the one-dimensional problem requires a larger l -register, and both the controlled rotation and Hamiltonian simulation subroutines scale more poorly with system size. As a result, simulations may suffer from greater error accumulation, particularly when modeled on classical hardware, which may reduce the overall accuracy of the solution.

IV. RESULTS

To evaluate the numerical accuracy of the HHL algorithm introduced in Sec. II, we perform a series of simulations comparing its performance with that of the Thomas algorithm, outlined in Algorithm 1. The Thomas algorithm is a well-established classical method for solving TST linear systems and serves as a benchmark in this context.

To ensure consistent and fair numerical comparisons between the two algorithms, all simulations were conducted on the same computer with an Intel Core i7-7820HK CPU operating at 2.90 GHz (4 cores, 8 logical processors) with 32 GB of RAM. The quantum circuits were constructed using QISKIT version 0.44 [28], and simulations were performed with QISKIT AER version 0.12.2 using the state-vector backend.

The following sections will first present the results from simulations solving the one-dimensional Poisson equation, and this will be followed by those for the two-dimensional Poisson equation. An empirical analysis of the HHL algorithm's time and qubit complexities will not be

TABLE VII. Asymptotic gate scaling of primary HHL subroutines for the two-dimensional Poisson equation.

Subroutine	Complexity
Quantum Fourier transform	$\mathcal{O}(\log_2^2(N))$
Hamiltonian simulation	$\mathcal{O}\left(\sqrt{\frac{N}{\epsilon}} \log_2^2(N)\right)$
Controlled rotation	$\mathcal{O}(N)$

 ALGORITHM I. Thomas algorithm for solving a tridiagonal linear system of equations $Ax = d$.

Input: Vectors $a, b, c \in \mathbb{R}^N$ representing the subdiagonal, diagonal, and superdiagonal of the system matrix, and vector $d \in \mathbb{R}^N$ for the right-hand side.

Output: Solution vector $x \in \mathbb{R}^N$.

```

for  $i \leftarrow 2$  to  $N$  do
     $m \leftarrow a_{i-1}/b_{i-1}$ 
     $b_i \leftarrow b_i - m \cdot c_{i-1}$ 
     $d_i \leftarrow d_i - m \cdot d_{i-1}$ 
end for
 $x_n \leftarrow d_n/b_n$ 
for  $i \leftarrow N-1$  to  $1$  do
     $x_i \leftarrow (d_i - c_i \cdot x_{i+1})/b_i$ 
end for
    
```

included in this paper, as we prioritize insights into algorithmic behavior over computational complexity. For each simulation, the parameter ϵ was tuned until the solution produced by the HHL algorithm closely approximated the analytical solution. Recall that ϵ influences the number of Trotter steps in the Hamiltonian simulation algorithm, which in turn influences the accuracy of the computation of e^{iAt} and, consequently, the solution generated by the HHL algorithm.

For notational simplicity, N will be used to denote the size of the $N \times N$ system matrix A . As outlined in Sec. II B, N is always taken to be a power of 2 to simplify state preparation and avoid unnecessary padding.

A. One-dimensional homogeneous boundary-value problem

The first set of simulations solves Eq. (4) with boundary conditions $u(0) = u(1) = 0$. The source functions considered are

- (a) $f_S(x) = \sin(\pi x)$,
- (b) $f_L(x) = 10x$,
- (c) $f_H(x) = 2H(x - 0.5) - 1$,

where $H(x)$ is the Heaviside step function. Since Eq. (4) has an analytical solution for these source functions and boundary conditions, both the Thomas and HHL algorithms will be evaluated by comparing their results against this analytical solution.

Figure 3 demonstrates that both the Thomas and HHL algorithms produce solutions that closely align with the analytical solution for $f(x) = f_H$. With $N = 8$, the coarse mesh limits the accuracy of the representation; however, with $N = 16$, both algorithms capture important solution features, such as the location and magnitude of the

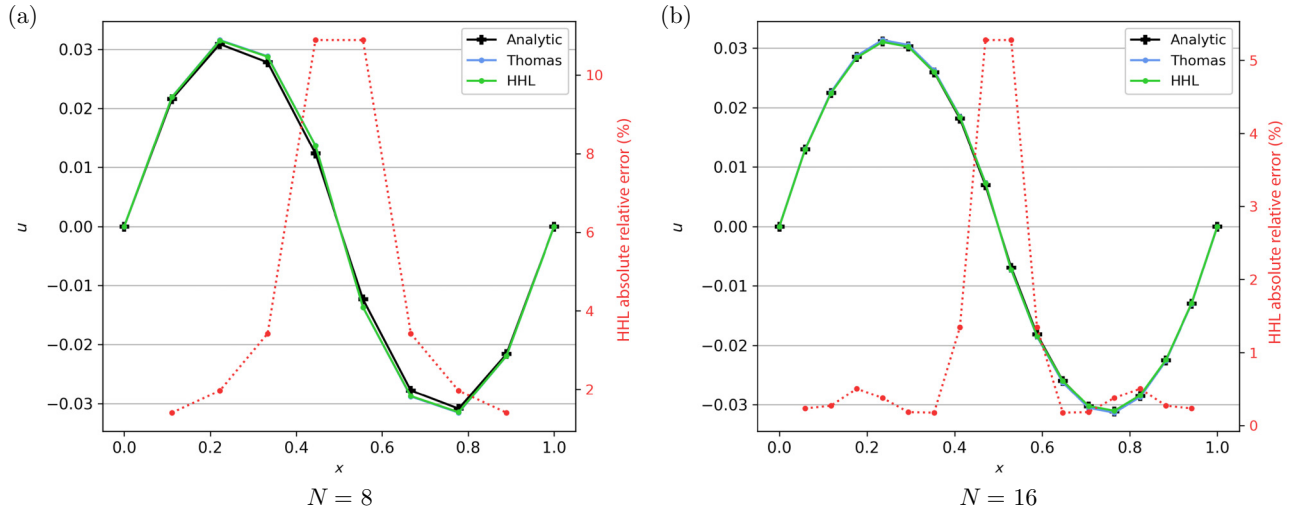


FIG. 3. Simulation results with $f(x) = f_H$, $\epsilon = 0.01$, and $u(0) = u(1) = 0$ for (a) $N = 8$ and (b) $N = 16$.

extrema, more accurately. Excluding the two central nodes, where the analytical solution values approach zero, the relative error between the HHL algorithm and the analytical solution remains below 1.3%.

A comparison between Figs. 3 and 4 shows that the HHL algorithm produces less accurate results for $f(x) = f_L$ than for $f(x) = f_H$. While the HHL algorithm shows close agreement with both the Thomas algorithm and the analytical solution for $N = 8$ and $\epsilon = 0.01$, it struggles to accurately capture the location and magnitude of the minima when the mesh is refined to $N = 16$ with $\epsilon = 0.01$. In contrast, the Thomas algorithm remains accurate within machine precision. Decreasing ϵ to 0.001 improves the HHL result, as shown in Fig. 4(c), where the relative error at the minima has reduced from 38% to 4.3%. Similar reductions in ϵ were necessary when refining the mesh from $N = 8$ to $N = 16$ with $f(x) = f_S$, and from $N = 16$ to $N = 32$ with $f(x) = f_H$, to ensure an accurate representation of the solution.

These results demonstrate that the HHL algorithm is capable of effectively solving the one-dimensional Poisson equation with homogeneous boundary conditions; however, its dependency on ϵ to accurately capture important solution features as the mesh is refined warrants further discussion. This will be addressed in Sec. IV D.

B. One-dimensional nonhomogeneous boundary-value problem

The next set of simulations considers the one-dimensional Poisson equation with nonhomogeneous boundary conditions of the type $|u(0)| = |u(1)|$ and $|u(0)| = 0$, $|u(1)| \neq 0$. The same source functions used in the homogeneous-boundary-value-problem simulations are applied here. We introduce a new term, ϵ_{hom} , to denote

the value of ϵ used in the corresponding homogeneous boundary-value problem with the same source function and system size.

Consistent trends in the HHL algorithm's behavior are more difficult to characterize under nonhomogeneous boundary conditions. In simulations where $|u(0)| = |u(1)|$, Figs. 5(b) and 5(d) suggest that choosing $\epsilon \leq \epsilon_{\text{hom}}$ yields a relative error profile of similar magnitude to that observed in the homogeneous setting.

In contrast, HHL simulations with $u(0) = 0$ and $u(1) \neq 0$ are highly sensitive to the choice of ϵ , with this sensitivity increasing as the mesh is refined. Figure 5(a) shows that for $f(x) = f_H$ with $N = 16$, the relative error remains below 3.5%, indicating reasonable accuracy; however, Fig. 5(c) demonstrates that refining the mesh to $N = 32$ results in a significant decline in accuracy. Although an ϵ value of 0.0038 minimizes the relative error profile for $N = 32$, the average relative error remains high at 43%, and the solution struggles to capture the right boundary condition properly. Small deviations from this optimal ϵ result in catastrophic reductions in accuracy. Increasing ϵ slightly to 0.0042 results in a 220% average relative error, while decreasing it to 0.0035 yields an error of 1575%. Similar sensitivity was observed in other simulations, where even minor perturbations from the optimal ϵ significantly impacted the HHL algorithm's numerical accuracy.

While choosing ϵ similar to ϵ_{hom} provides a practical approach for nonhomogeneous boundary-value problems with $|u(0)| = |u(1)|$, the choice of ϵ becomes arbitrary if $|u(0)| \neq |u(1)|$, requiring a trial-and-error process to achieve an acceptable relative error. This sensitivity to the imposed boundary conditions necessitates an additional study into their impact on the accuracy of the HHL algorithm. This will be discussed in Sec. IV C.

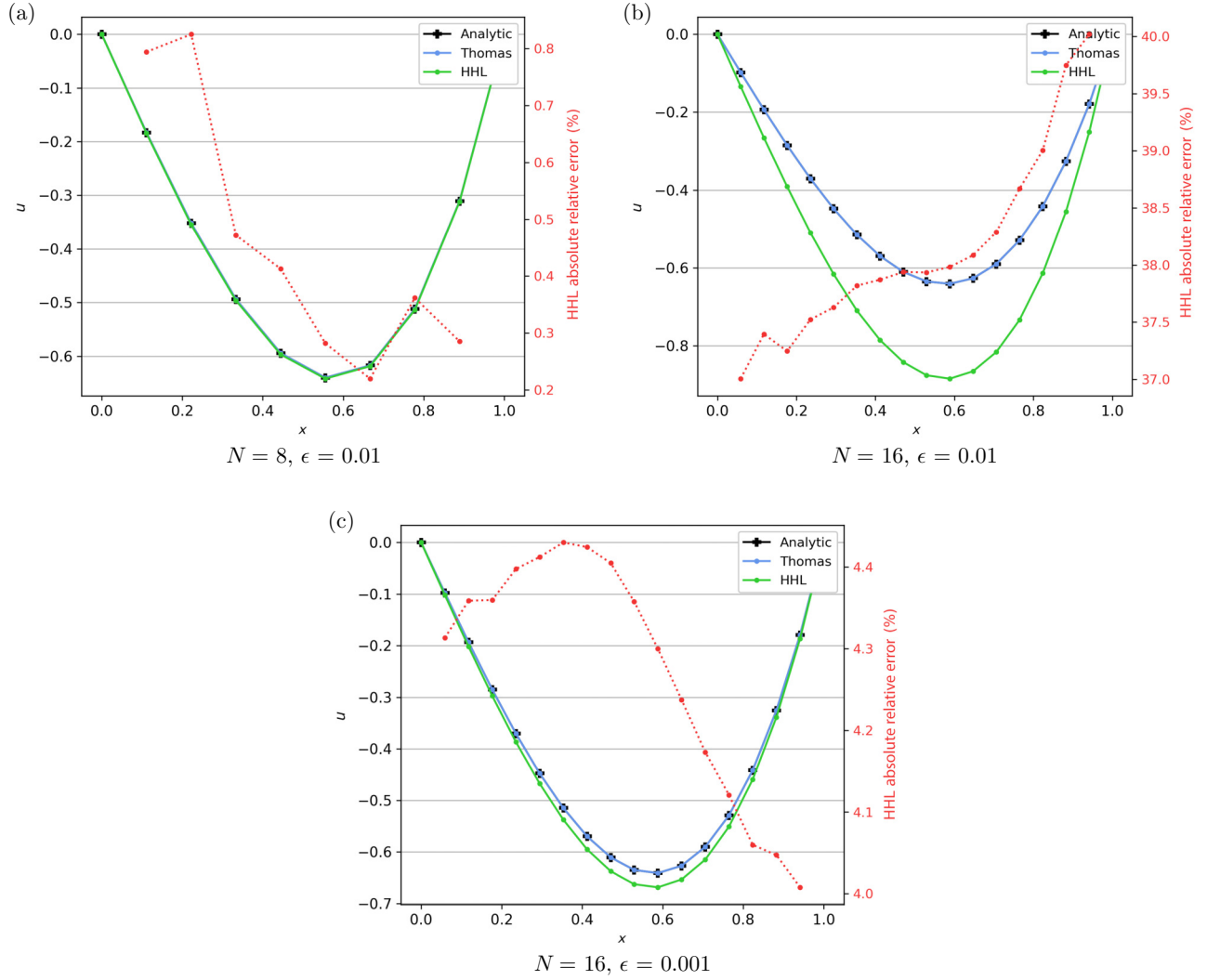


FIG. 4. Simulation results with $f(x) = f_L$ and $u(0) = u(1) = 0$ for: (a) $N = 8, \epsilon = 0.01$; (b) $N = 16, \epsilon = 0.01$; (c) $N = 16, \epsilon = 0.001$.

C. One-dimensional boundary-condition sensitivity analysis

To study the effect of the imposed boundary conditions on the numerical accuracy of the HHL algorithm, a series of simulations were conducted with ϵ fixed at ϵ_{hom} , $u(0)$ kept constant, and $u(1)$ incrementally varied. To mitigate the effect of near-zero values in the analytical solution, the maximum and average absolute errors were recorded for each simulation rather than the relative errors.

Figure 6 shows a clear trend for the study with $f(x) = f_H$. In Fig. 6(a), where $u(0) = 0$, both the maximum and average absolute errors are minimized when the boundary conditions are equal and homogeneous. As the absolute difference between the boundary conditions increases, both error metrics increase accordingly. Interestingly, in both Figs. 6(b) and 6(c), where $u(0) \neq 0$, the maximum absolute error is minimized when $u(1) = -u(0)$. For $u(0) =$

-0.05 [Fig. 6(b)], the average absolute error is also minimized at this point. In contrast, for $u(0) = 0.3$ [Fig. 6(c)], the minimum average error is attained for values of $u(1)$ close to $-u(0)$. Both error metrics increase as the absolute difference between $u(1)$ and $-u(0)$ grows. A similar trend was observed for $f(x) = f_L$ (not shown here for brevity).

The boundary-condition study with $f(x) = f_S$ reveals a notable deviation from the expected trend. While most boundary-condition studies were consistent with the behavior shown in Fig. 6, a notable difference is observed in Fig. 7, corresponding to $u(0) = 0.05$. The maximum-error profile exhibits a global minimum at $u(1) = -u(0)$ and a local minimum at $u(1) = u(0)$, with the error increasing as $u(1)$ is further increased. The average absolute error attains a minimum for values of $u(1)$ close to $-u(0)$, with an inflection point at $u(1) = u(0)$, followed by a subsequent increase in error for larger values of $u(1)$.

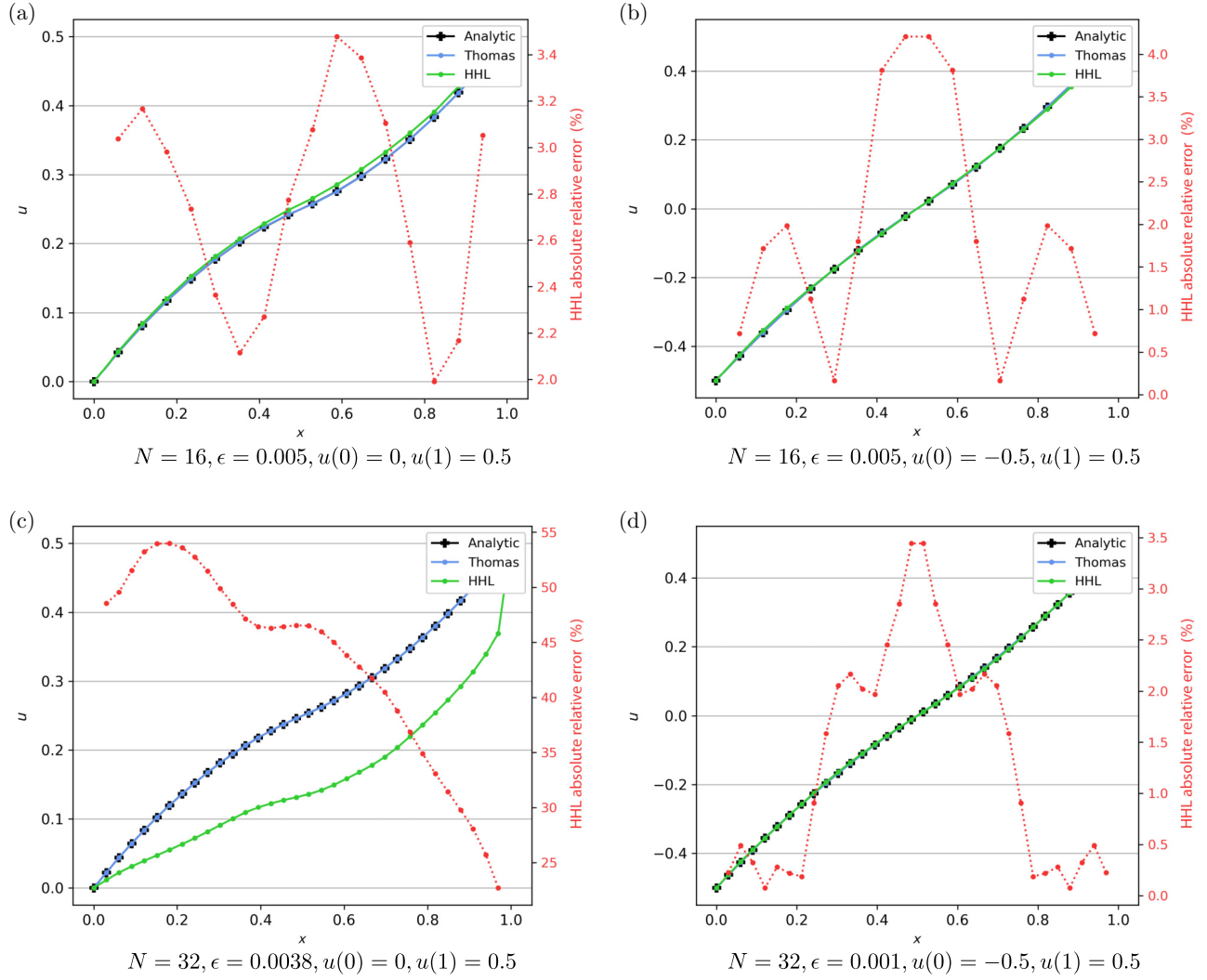


FIG. 5. Simulation results with $f(x) = f_H$: (a) $N = 16, \epsilon = 0.005, u(0) = 0, u(1) = 0.5$; (b) $N = 16, \epsilon = 0.005, u(0) = -0.5, u(1) = 0.5$; (c) $N = 32, \epsilon = 0.0038, u(0) = 0, u(1) = 0.5$; (d) $N = 32, \epsilon = 0.001, u(0) = -0.5, u(1) = 0.5$.

This study into the impact of the imposed boundary conditions on the numerical accuracy of the HHL algorithm reveals two key observed behaviors. First, the maximum absolute error was observed to be minimized when $u(1) = -u(0)$, while the average absolute error attained its minimum for values of $u(1)$ close to $-u(0)$. Second, both error metrics were observed to increase with the absolute difference between $u(1)$ and $-u(0)$.

D. One-dimensional ϵ sensitivity analysis

To investigate the impact of ϵ on the numerical accuracy of the HHL algorithm, a series of simulations were conducted in which the boundary conditions were fixed and ϵ was incrementally varied. Following the approach used in the one-dimensional boundary-condition sensitivity analysis, the maximum and average absolute errors

were recorded for each simulation rather than the relative errors.

Figure 8 presents the results of the ϵ sensitivity study for $f(x) = f_L$. With homogeneous boundary conditions, Fig. 8(a) demonstrates that reducing ϵ consistently lowers the maximum and average absolute errors; however, Figs. 8(b)–8(d) show that for nonhomogeneous boundary-value problems, reducing ϵ initially decreases both error metrics, reaching a minimum at $\epsilon = \epsilon_{\min}$, beyond which, further reductions lead to a rise in error followed by a plateau. The magnitude of this increase correlates with the difference between $u(0)$ and $u(1)$, where larger differences lead to a more significant error increase before plateauing. This high sensitivity to ϵ when $u(0) = 0$ and $u(1) \neq 0$ is consistent with the findings in Sec. IV B.

We initially hypothesized that the error increase in the ϵ studies with nonhomogeneous boundary conditions was due to error accumulation from the Lie-Trotter-Suzuki

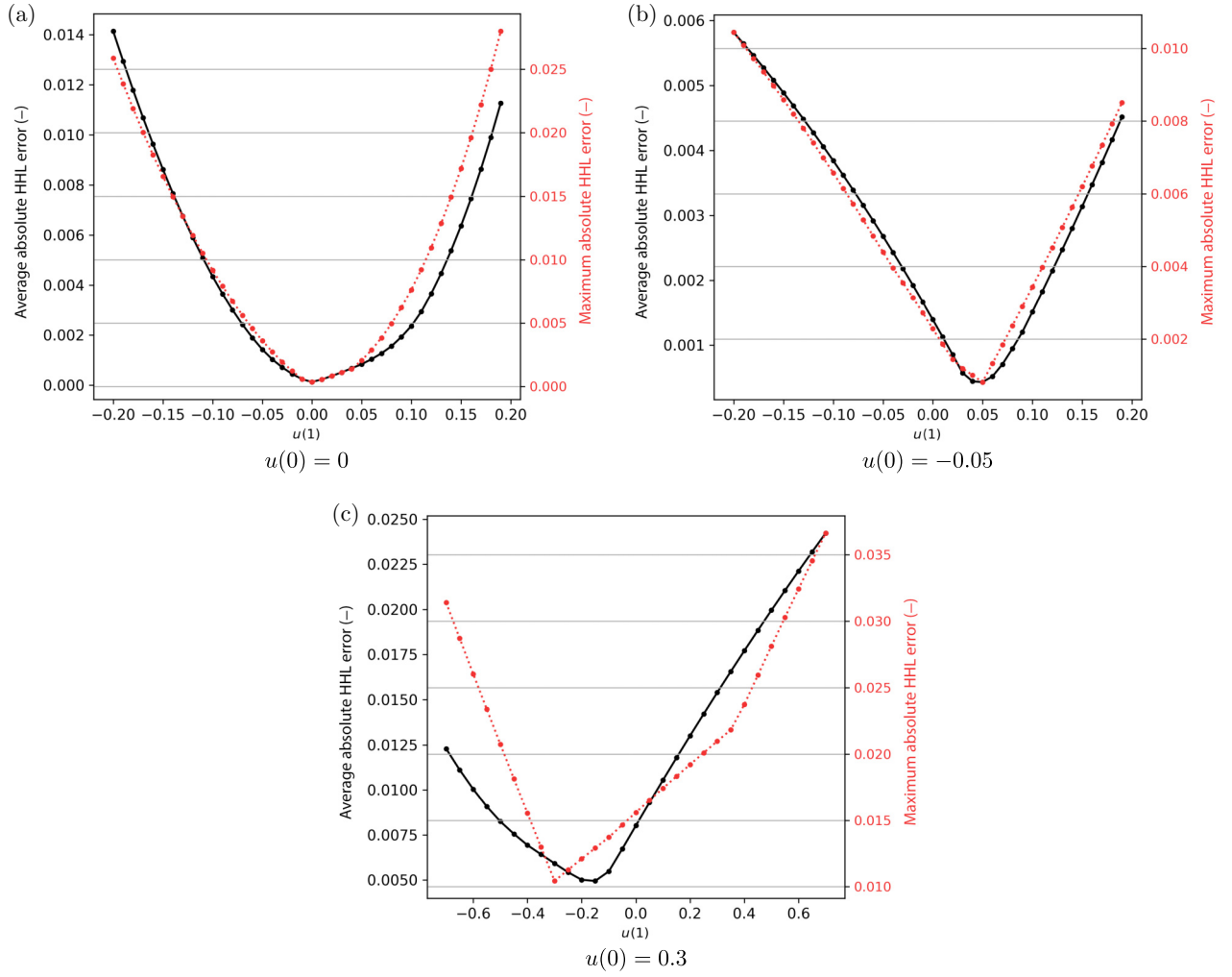


FIG. 6. Average and maximum absolute errors between the solution generated by the HHL algorithm and the analytical solution as a function of $u(1)$, with $f(x) = f_H$, $N = 16$, and $\epsilon = 0.01$: (a) $u(0) = 0$; (b) $u(0) = -0.05$; (c) $u(0) = 0.3$.

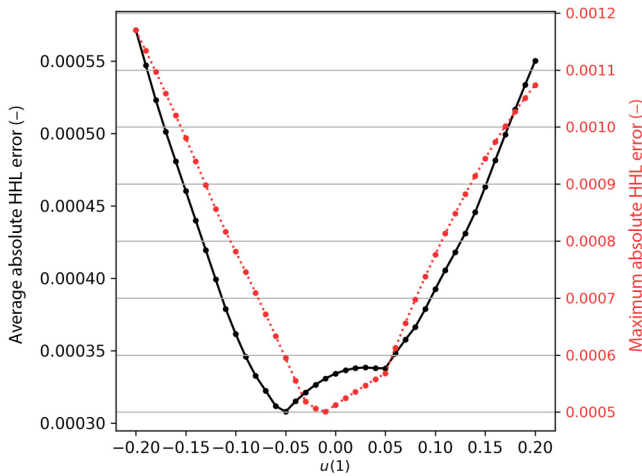


FIG. 7. Average and maximum relative errors between the solution generated by the HHL algorithm and the analytical solution as a function of $u(1)$, with $f(x) = f_S$, $u(0) = 0.05$, $N = 16$, and $\epsilon = 0.001$.

approximation. For very small ϵ values, the larger number of gate operations could introduce numerical errors that negate the advantage of more Trotter steps; however, this is contradicted by the consistent error reduction in the homogeneous-boundary-condition study. A similar pattern was observed with $f(x) = f_S$, but this is omitted here for brevity.

With $f(x) = f_H$ and homogeneous boundary conditions, Fig. 9 shows a deviation from the previously observed behavior. Initially, reducing ϵ lowers both the maximum and average absolute errors; however, similar to the non-homogeneous setting, these errors reach a minimum at $\epsilon = \epsilon_{\min}$ before increasing with further reductions in ϵ . Interestingly, after this increase in error, the errors subsequently decrease, following a consistent downward trend as ϵ is reduced. In contrast, the ϵ study with $f(x) = f_H$ and nonhomogeneous boundary conditions produced results consistent with those observed for $f(x) = f_S$ and $f(x) = f_L$ in Fig. 8, and this has therefore been omitted for brevity.

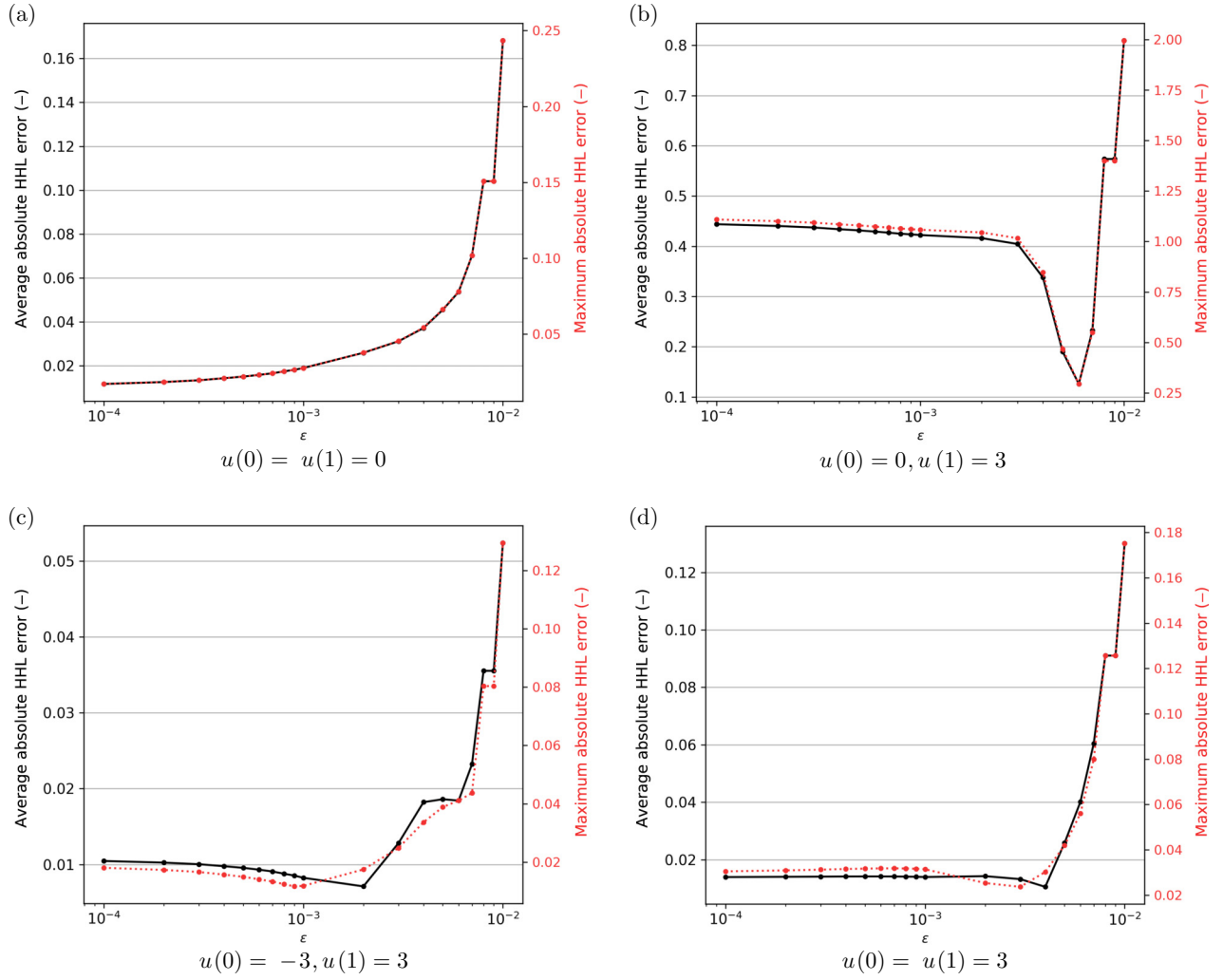


FIG. 8. Average and maximum absolute errors between the solution generated by the HHL algorithm and the analytical solution as a function of ϵ , with $f(x) = f_L$ and $N = 16$: (a) $u(0) = u(1) = 0$; (b) $u(0) = 0, u(1) = 3$; (c) $u(0) = -3, u(1) = 3$; (d) $u(0) = u(1) = 3$.

This study into the impact of ϵ on the HHL algorithm reveals two key observations. First, for homogeneous boundary-value problems, reducing ϵ consistently decreased both the maximum and average absolute errors; however, for nonhomogeneous boundary-value problems, selecting the optimal ϵ required a trial-and-error approach to identify $\epsilon_{\min}$. Second, in nonhomogeneous boundary-value problems where $|u(0)| \neq |u(1)|$, the HHL algorithm exhibited increased sensitivity to ϵ near $\epsilon_{\min}$.

E. Two-dimensional homogeneous boundary-value problem

The following subsections present the results from simulations solving the two-dimensional Poisson equation using our hybrid QLSA. For brevity, we will continue to refer to the extended Thomas and HHL algorithms by their original names.

The first set of simulations solve Eq. (7) with $u(x, 0) = u(x, 1) = u(0, y) = u(1, y) = 0$. The source functions considered are

- (a) $f_L(x, y) = 10x$,
- (b) $f_H(x, y) = 4 - 8H(x - 0.5, y)$,
- (c) $f_S(x, y) = 10 \sin(2\pi x) \cos(2\pi y)$,

where $H(x, y) = H(x)$. The analytical solution is approximated using the Thomas algorithm on a refined mesh with spacing $\Delta h = 1/153$. This choice of Δh allows for the computation of node-wise errors for system matrices with $N = 8$ and $N = 16$ without requiring interpolation between nodes. Both the Thomas and HHL algorithms are considered converged when the iteration error, defined as $\max(|\mathbf{u}^{n+1} - \mathbf{u}^n|)$, falls below 1×10^{-8} , a threshold

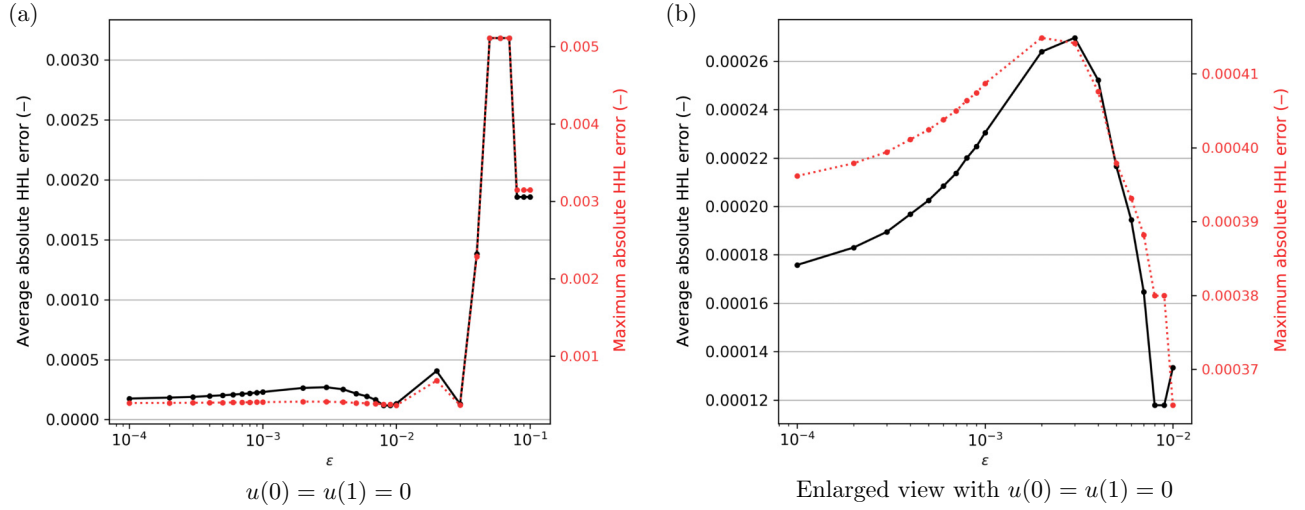


FIG. 9. Average and maximum absolute errors between the solution generated by the HHL algorithm and the analytical solution as a function of ϵ , with homogeneous boundary conditions, $f(x) = f_H$, and $N = 16$: (a) $u(0) = u(1) = 0$; (b) enlarged view with $u(0) = u(1) = 0$.

that ensures a balance between accuracy and reasonable simulation runtimes.

The numerical accuracy of the HHL algorithm in solving the two-dimensional Poisson equation with $f(x, y) = f_S$ closely matches that of the Thomas algorithm, with both methods accurately predicting the locations of the extrema. The large relative errors are attributed to the coarse mesh discretization used. Similar results were observed for $f(x, y) = f_H$ and $f(x, y) = f_L$ with $N = 8$.

Based on the analysis of the HHL algorithm in solving the one-dimensional Poisson equation, it was expected that refining the mesh to $N = 16$ would require either keeping ϵ fixed or reducing it to ensure accurate results. While this strategy proved effective for simulations with $f(x, y) = f_L$, with $f(x, y) = f_S$, it either failed to capture key features of the analytical solution [Fig. 11(a)] or introduced spurious oscillations [Fig. 11(b)], resulting in oscillating residuals and nonconvergence. The emergence of oscillations was surprising since, in Appendix D, it was demonstrated that the line-Jacobi scheme dampens high-frequency components of numerical solutions.

A comparison between Figs. 11 and 12 shows that increasing ϵ to 0.5 significantly improves the alignment between the solutions generated by the HHL and Thomas algorithms, with both methods correctly identifying the location of the extrema. The Thomas algorithm closely matches the analytical solution, maintaining a relative error below 4%. In contrast, the HHL algorithm amplifies the extrema and produces steeper gradients near the zero set of the source function, defined by the union of the lines $x = 0, 0.5$, and 1 , and $y = 0.25$ and 0.75 . This behavior results in higher relative errors for the HHL algorithm compared to the Thomas algorithm.

The final set of simulations considered $f(x, y) = f_H$ with $N = 16$. Initial simulations demonstrated worse performance, with ϵ values smaller than those used for $N = 8$ producing spurious oscillations and larger values resulting in the HHL algorithm failing to capture solution features accurately. Figure 13(b) demonstrates that using the HHL algorithm with $\epsilon = 0.03$ produces a solution that does not match the expected analytical behavior in Fig. 13(a). Interestingly, Fig. 13(c) shows that the HHL algorithm is able to capture the general solution features within the first 30 iterations with $\epsilon = 0.03$; however, small-scale high-frequency oscillations emerge over the next 10 iterations, leading to a degradation of the solution similar to that shown in Fig. 13(b).

The final set of simulations considered $f(x, y) = f_H$ with $N = 16$. Initial results showed worse performance. Values of ϵ smaller than those used for $N = 8$ led to the onset of spurious oscillations, while larger values of ϵ resulted in the HHL algorithm failing to capture important solution features. Figure 13(b) shows that with $\epsilon = 0.03$, the HHL is unable to reproduce the expected analytical behavior in Fig. 13(a). Interestingly, Fig. 13(c) demonstrates that during the first 30 iterations, the algorithm captures the general shape of the solution; however, over subsequent iterations, small-scale high-frequency oscillations emerge and permeate the domain, ultimately degrading the solution in a manner consistent with what is shown in Fig. 13(b).

To improve the performance of the HHL algorithm, an initial approach was to warm-start the algorithm using the solution obtained after five iterations of the Thomas algorithm; however, this resulted in a similar degradation of solution quality. A possible explanation for this is that the discontinuity in f_H aligns with the axis of

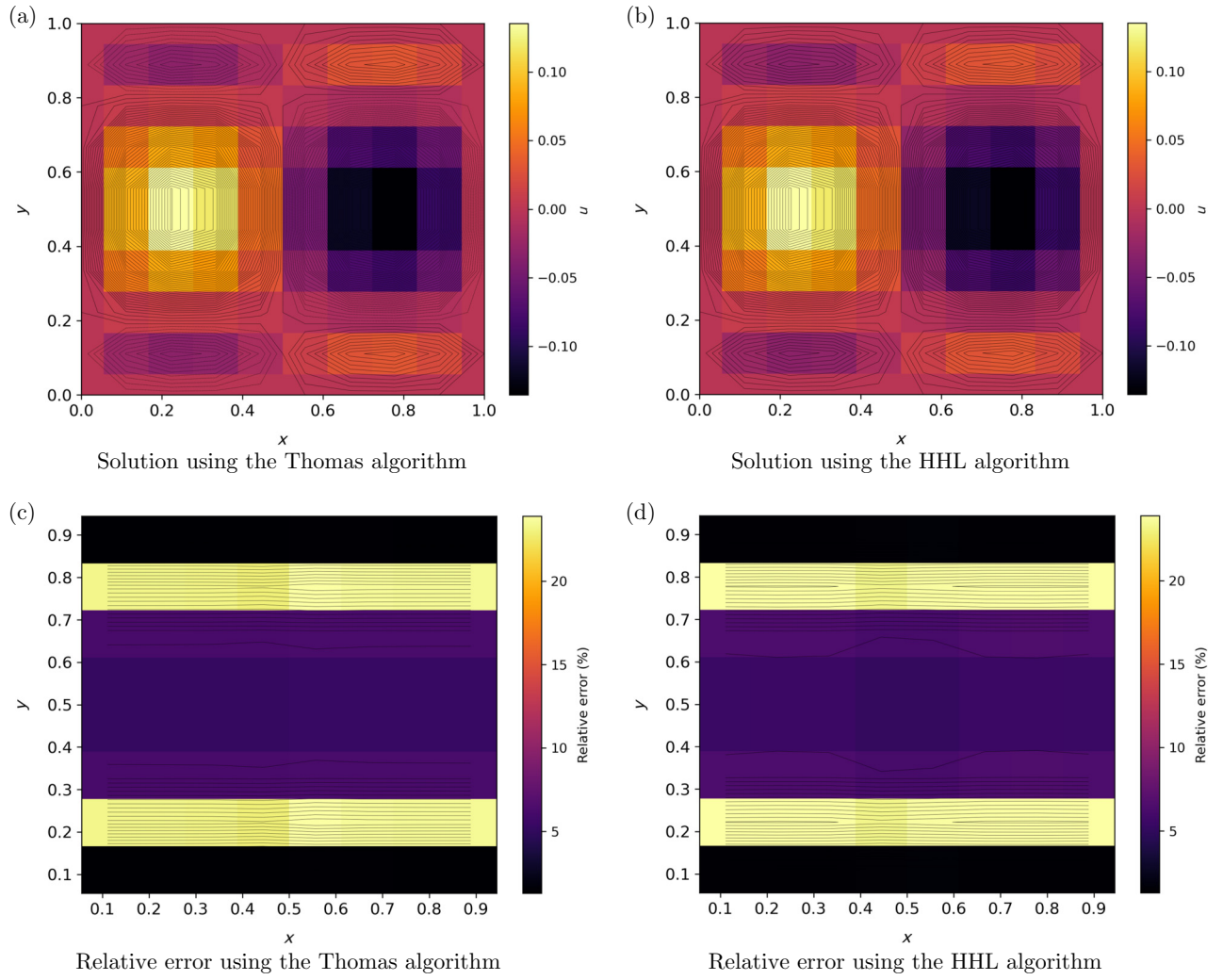


FIG. 10. Simulation results with $f(x, y) = f_S$, $N = 8$, and $\epsilon = 0.01$. (a) Solution using the Thomas algorithm. (b) Solution using the HHL algorithm. (c) Relative error using the Thomas algorithm. (d) Relative error using the HHL algorithm.

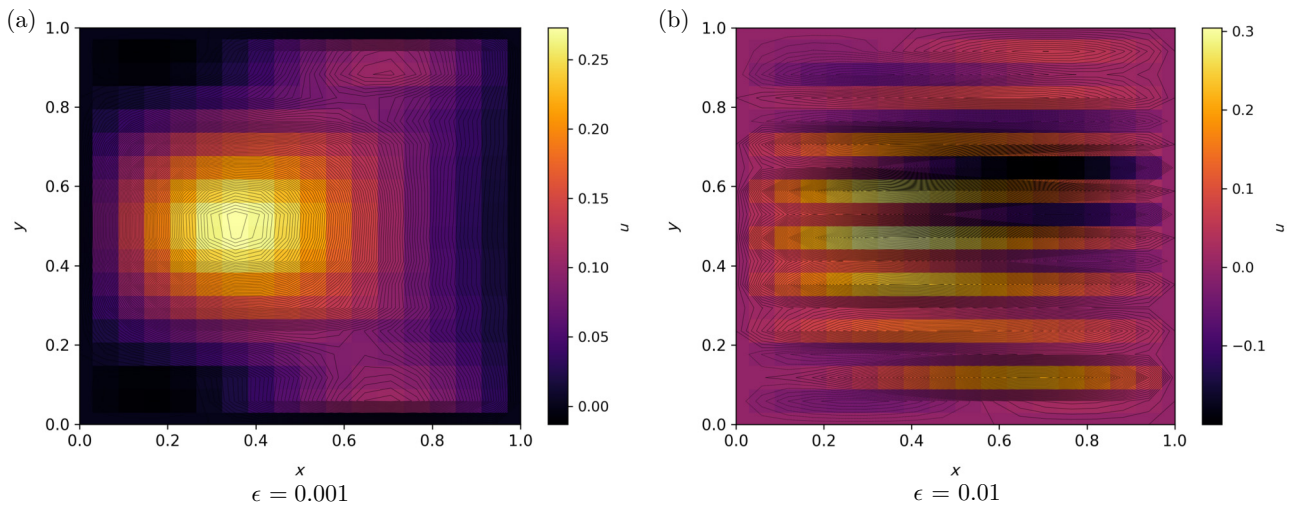


FIG. 11. Simulation results from the HHL algorithm with $f(x, y) = f_S$ and $N = 16$ for (a) $\epsilon = 0.001$ and (b) $\epsilon = 0.01$.

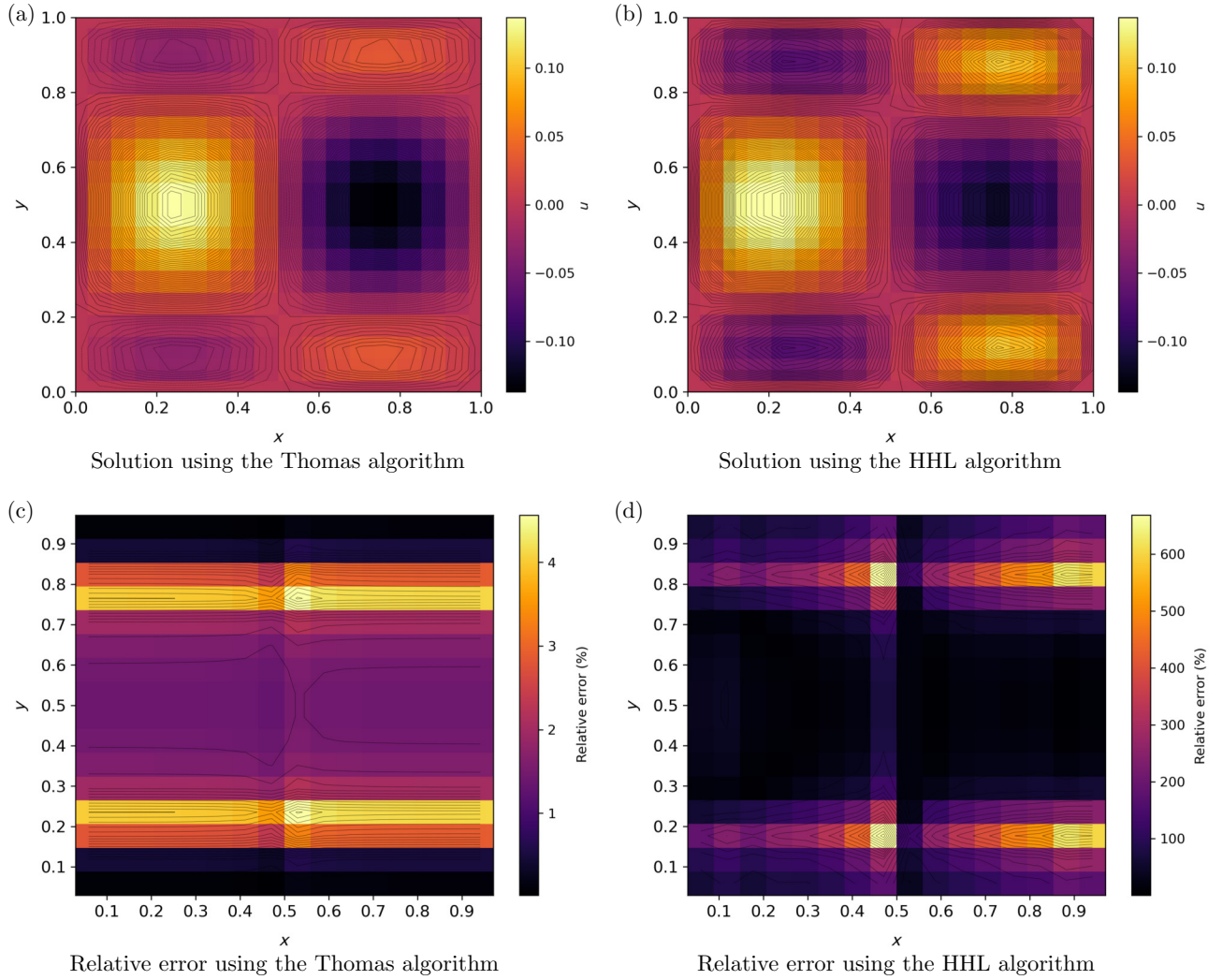


FIG. 12. Simulation results with $f(x,y) = f_S$, $N = 16$, and $\epsilon = 0.5$. (a) Solution using the Thomas algorithm. (b) Solution using the HHL algorithm. (c) Relative error using the Thomas algorithm. (d) Relative error using the HHL algorithm.

the line-Jacobi scheme. While the assumption of f being smooth may hold for coarser meshes, it becomes invalid as the mesh is refined, potentially leading to the observed issues. To test this hypothesis, the source function was modified to $H(x,y) = H(y)$, changing the axis of the discontinuity to the y axis, thereby making it orthogonal to the axis of the line-Jacobi scheme. This adjustment improved the algorithm's ability to capture key solution features, such as the location of extrema and the overall solution shape, although very small-scale high-frequency oscillations remained [Fig. 13(d)]. Warm-starting the HHL algorithm using the solution obtained after five iterations of the Thomas algorithm was ineffective in dampening these oscillations. Despite these modifications, Fig. 13(d) shows that the HHL algorithm still overpredicts the magnitude of the extrema relative to the analytical solution.

Evaluating the HHL algorithm's ability to solve the two-dimensional Poisson equation with homogeneous

boundary conditions reveals inherent challenges. For smaller problem sizes, the algorithm performs comparably to the Thomas algorithm; however, simulations with $N = 16$ showed that the HHL algorithm struggles to accurately capture key solution features, such as the magnitude and location of extrema, while also amplifying gradients. As the problem size increases, the HHL algorithm becomes increasingly sensitive to ϵ , making the tuning process complex and unpredictable.

F. Two-dimensional nonhomogeneous boundary-value problem

Given the challenges encountered when solving the two-dimensional Poisson equation with homogeneous boundary conditions using the HHL algorithm for $N = 16$, the following analysis is restricted to system matrices of size $N = 8$. Moreover, to reduce the impact of near-zero values

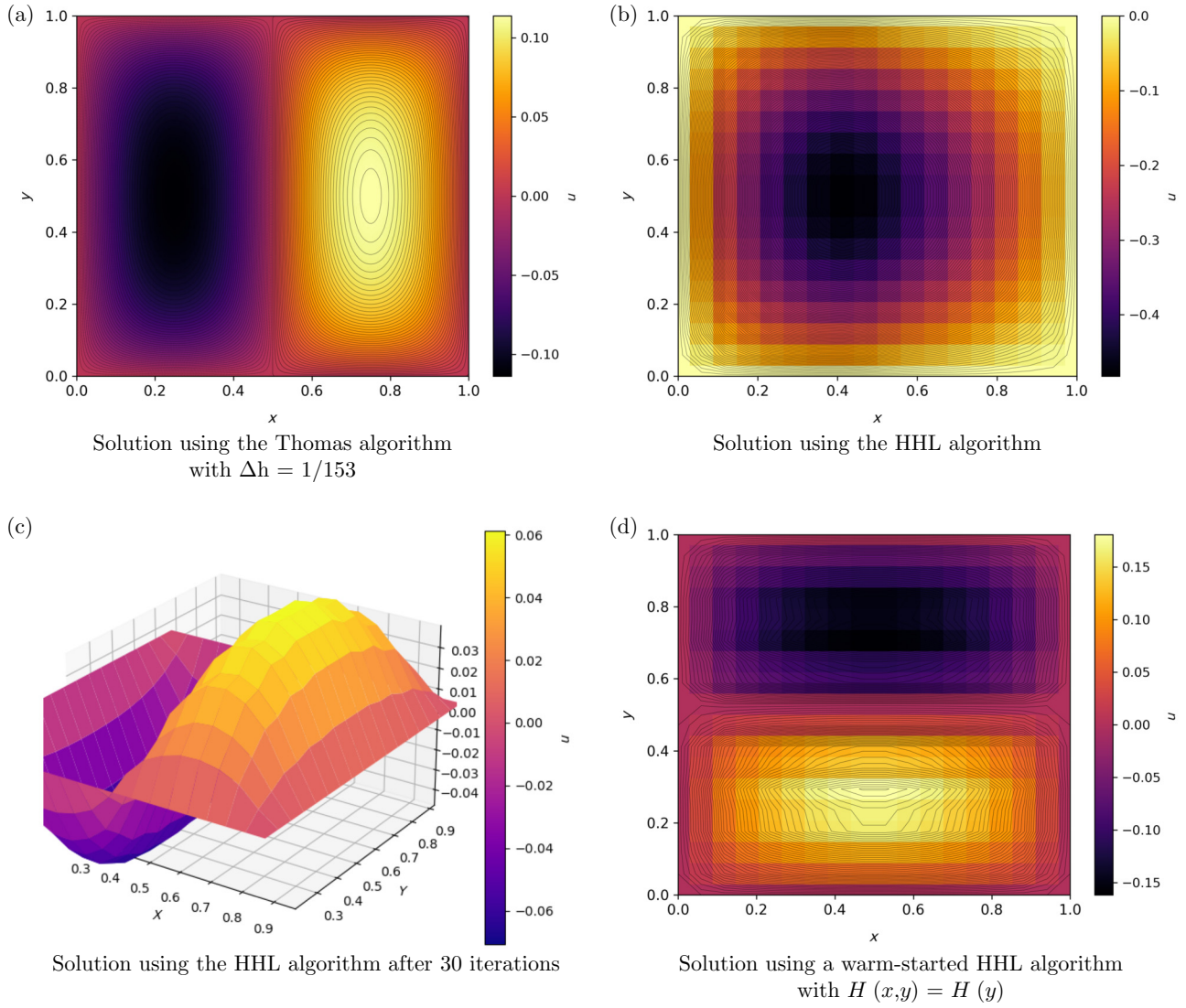


FIG. 13. Simulation results with $f(x,y) = f_H$, $N = 16$, and $\epsilon = 0.03$. (a) Solution using the Thomas algorithm with $\Delta h = 1/153$. (b) Solution using the HHL algorithm. (c) Solution using the HHL algorithm after 30 iterations. (d) Solution using a warm-started HHL algorithm with $H(x,y) = H(y)$.

in the analytical solution, which would inflate the relative errors and complicate interpretation, the absolute error is reported instead of the relative error.

Figure 14 shows the solution to the two-dimensional Poisson equation with $f(x,y) = f_H$ and the following boundary conditions:

- (a) $u(0,y) = 0.5$,
- (b) $u(1,y) = \sin(\pi y)$,
- (c) $u(x,0) = (x-0.5)(x-1)$,
- (d) $u(x,1) = 0.5(x-1)$.

The solutions obtained from both the HHL and Thomas algorithms show good agreement, though both exhibit regions with significantly large absolute errors. This behavior is anticipated when solving two-dimensional

PDEs with complex boundary conditions on a coarse mesh. While the two methods demonstrated similar levels of agreement across a large number of tested simulations, a notable exception is presented in Fig. 15. This specific simulation, with $f(x,y) = f_S$ and the following asymmetric boundary conditions, illustrates an instance where the HHL algorithm did not converge:

- (a) $u(0,y) = 0.08(1-y)$,
- (b) $u(1,y) = 0.08y$,
- (c) $u(x,0) = 0.08(1-x)$,
- (d) $u(x,1) = 0.08x$.

To investigate the cause of the nonconvergence, additional simulations were performed in which the strength of the source function was halved and the strength of

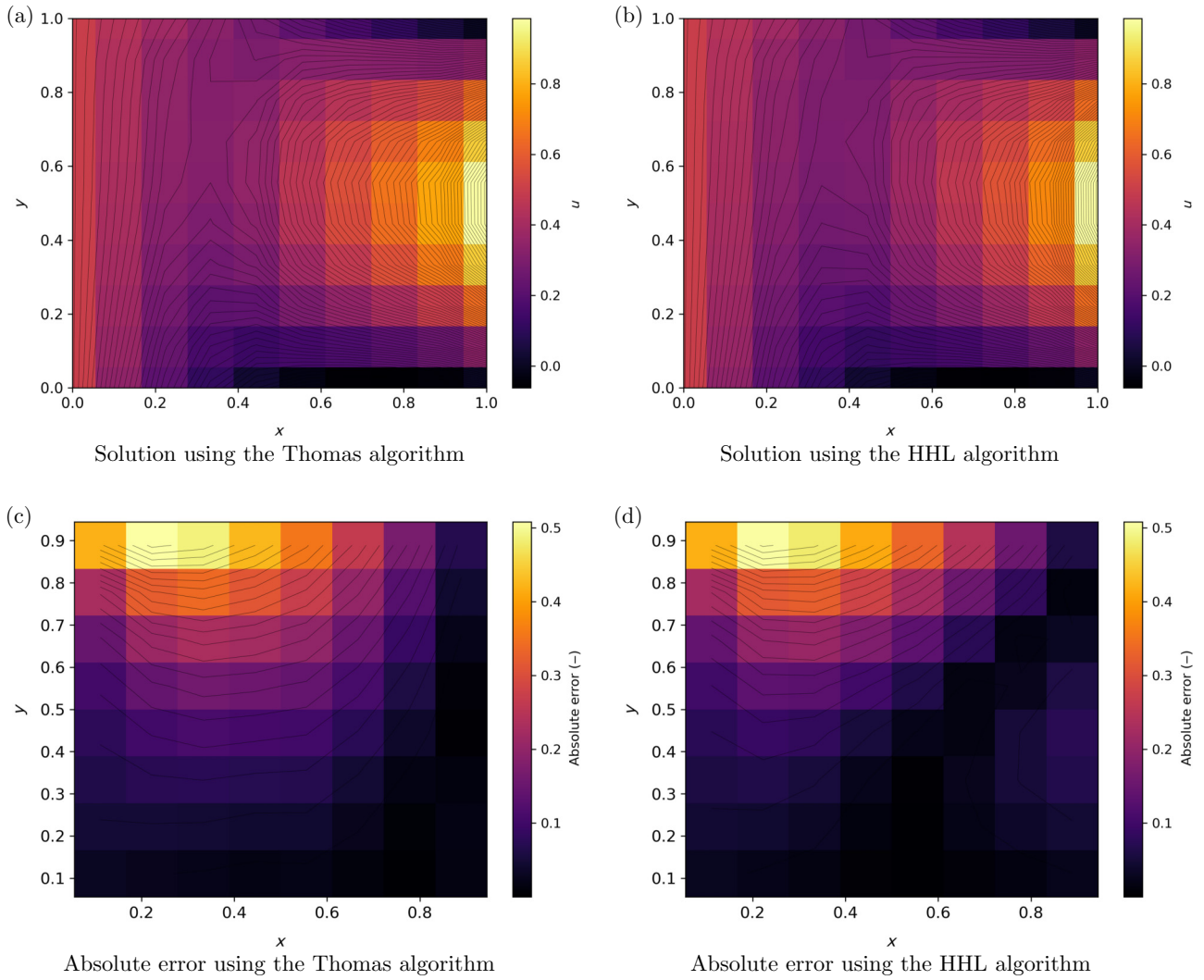


FIG. 14. Simulation results with $f(x, y) = f_H$, $N = 8$, and $\epsilon = 0.01$. (a) Solution using the Thomas algorithm. (b) Solution using the HHL algorithm. (c) Absolute error using the Thomas algorithm. (d) Absolute error using the HHL algorithm.

the boundary conditions was reduced to 0.03. These adjustments enabled the HHL algorithm to converge successfully, producing relative errors comparable to those obtained with the Thomas algorithm. To further evaluate whether the asymmetry in boundary conditions was responsible for the convergence issue, another simulation was performed with $f(x, y) = f_S$ and the following symmetric boundary conditions, matching the amplitude used in the case that failed to converge:

- (a) $u(0, y) = 0.08y$,
- (b) $u(1, y) = 0.08y$,
- (c) $u(x, 0) = 0$,
- (d) $u(x, 1) = 0.08$.

This simulation was motivated by the boundary-condition study in Sec. IV C, which showed that the

numerical accuracy of the HHL algorithm in solving the one-dimensional Poisson equation decreased as the absolute difference between the left- and right-hand boundary conditions grew. As shown in Fig. 16, imposing symmetric boundary conditions enabled the HHL algorithm to converge successfully. Moreover, the solution produced by the HHL algorithm achieves a lower absolute error than the Thomas algorithm across the entire domain.

Overall, the numerical accuracy of the HHL algorithm in solving the two-dimensional Poisson equation with nonhomogeneous boundary conditions is comparable to that of the Thomas algorithm for $N = 8$ and symmetric boundary conditions; however, highly asymmetric boundary conditions lead to poor convergence behavior in the HHL algorithm. This is analogous to the reduced numerical accuracy observed when solving the one-dimensional

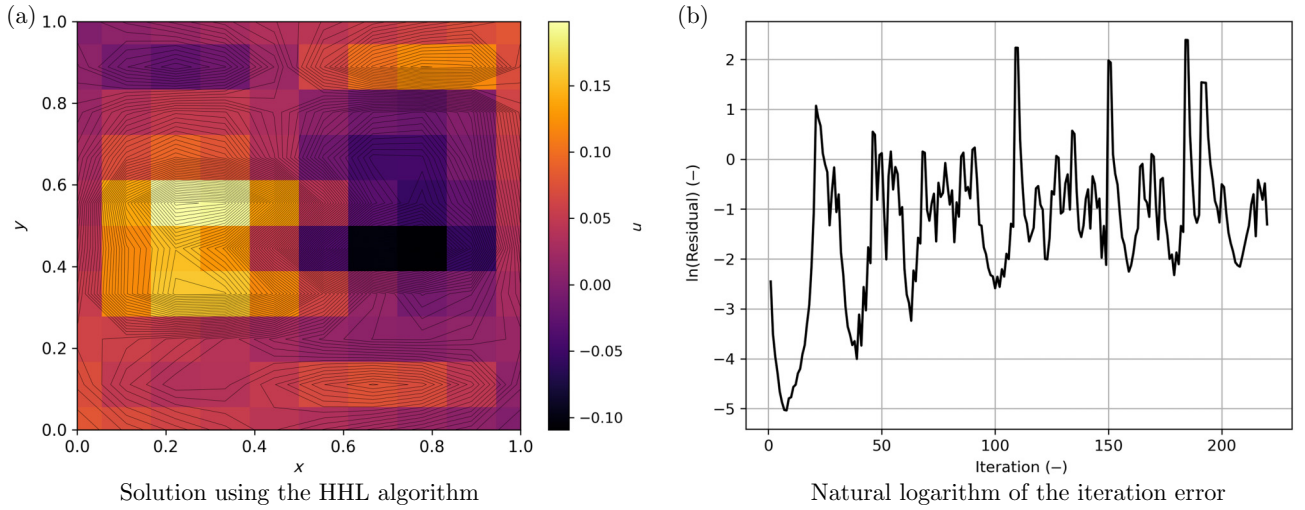


FIG. 15. Simulation results with $f(x,y) = f_S$, $N = 8$, $\epsilon = 0.01$, and asymmetric boundary conditions. (a) Solution using the HHL algorithm. (b) Natural logarithm of the iteration error.

nonhomogeneous boundary-value problem with $|u(0)| \neq |u(1)|$.

V. CONCLUSION

In light of the gradual departure from Moore's Law in recent years, this paper has explored the applicability of the HHL algorithm to solving the Poisson equation. The motivation stems from fluid dynamics, where solving the Poisson equation for the pressure field accounts for nearly 80% of the total computational cost in solving the incompressible Navier-Stokes equations.

When applied to the one-dimensional Poisson equation with homogeneous boundary conditions, the HHL

algorithm captured important solution features, including the location and magnitude of extrema, with low relative errors; however, its performance in solving nonhomogeneous boundary-value problems was highly sensitive to boundary conditions and ϵ , posing a potential challenge to tuning the algorithm accurately when no prior knowledge of the solution is available. Extending the HHL algorithm to the two-dimensional Poisson equation with our hybrid QLSA presented additional challenges. While it performed similarly to the Thomas algorithm on a coarse mesh, refining the mesh led to issues such as spurious oscillations, inaccurate capture of solution features, and nonconverging residuals.

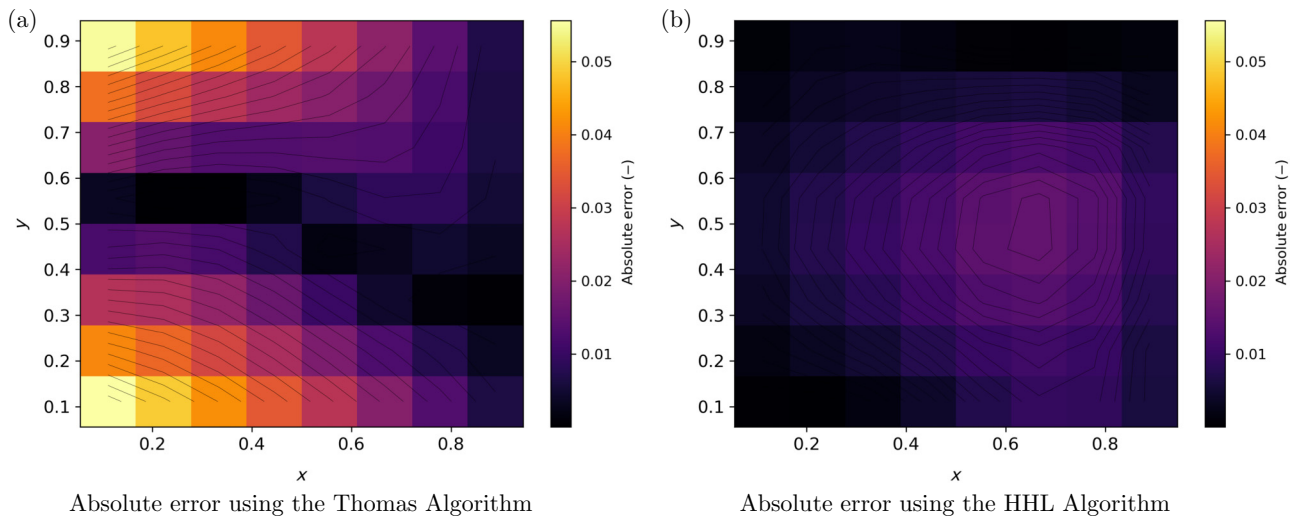


FIG. 16. Simulation results with $f(x,y) = f_S$, $N = 8$, $\epsilon = 0.01$, and symmetric boundary conditions. (a) Absolute error using the Thomas algorithm. (b) Absolute error using the HHL Algorithm.

Given these observations, we recommend three directions for future work. Our hybrid QLSA extended the HHL algorithm to solve the two-dimensional Poisson equation using a line-Jacobi scheme to decompose the problem into a series of one-dimensional problems with TST system matrices. This strategy allowed the use of the Hamiltonian simulation algorithm from Ref. [24], assuming the HHL algorithm could be prepared, run, and reinitialized with an intermediate solution; however, in practice, this assumption is not valid. Therefore, one potential direction would be to extend the Hamiltonian simulation algorithm from Ref. [24] to handle direct simulation of block-banded matrices similar to Eq. (8). This would support the direct simulation of finite-difference approximations of the two-dimensional Poisson equation with Dirichlet boundary conditions.

Another direction for future work is identifying a measurement function $F(x)$ relevant to fluid dynamics. We used state-vector simulations to compare the accuracy of the HHL and Thomas algorithms; however, the computational resources for state-vector simulations scale exponentially with the number of qubits, diminishing any speedup. To leverage the exponential scaling benefits of the HHL algorithm, extracting information using a suitable $F(x)$ is essential. For cases where only solution statistics are needed, the HHL algorithm holds promise; however, since $F(x)$ outputs a single value rather than the full solution vector, our iterative hybrid QLSA becomes impractical for two-dimensional problems, suggesting a need to move beyond the hybrid approach.

The final direction for future work is motivated by the current need to tune ϵ using prior knowledge of the solution to the Poisson equation. Developing methods to determine the optimal ϵ based on the source function and boundary conditions would enhance the algorithm's practical applicability. Machine learning approaches, such as using convolutional neural networks to improve initial guesses for iterative solvers, have shown promise [29]. A similar approach could be explored to learn an appropriate ϵ using only the source term, boundary conditions, and discretization information.

ACKNOWLEDGMENTS

The authors would like to thank Enrico Fontana, Almudena Carrera Vázquez, and Adrien Vandenbroucq for their invaluable insights.

DATA AVAILABILITY

The data are not publicly available. The data are available from the authors upon reasonable request.

APPENDIX A: HAMILTONIAN SIMULATION ALGORITHM

The core of the QPE algorithm is the computation of e^{iAt} , known as Hamiltonian simulation, which models the time

evolution of a quantum state's dynamics within a closed quantum system. While Sec. II C provides a high-level description that is sufficient for interpreting the simulation results, the following provides a more detailed explanation of the Hamiltonian simulation algorithm for TST matrices presented in Ref. [24]. Consider the following TST matrix:

$$A = \begin{pmatrix} a & b & & & \\ b & a & b & & \\ & \ddots & \ddots & \ddots & \\ & & b & a & b \\ & & & b & a \end{pmatrix}, \quad a, b \in \mathbb{R}. \quad (\text{A1})$$

The algorithm to compute e^{iAt} consists of three steps:

- (a) Find a decomposition of A into k -local Hamiltonians such that $A = \sum_i A_i$, where A_i is a 1-sparse matrix.
- (b) Simulate $e^{iA_i t}$.
- (c) Use a Lie-Trotter-Suzuki approximation to combine the actions of each $e^{iA_i t}$ to simulate e^{iAt} .

1. Decomposition into 1-sparse local Hamiltonians

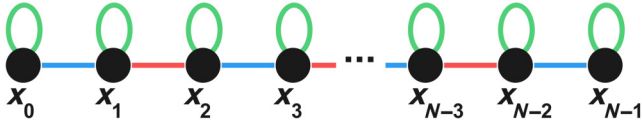
The following definitions will prove useful:

Definition A1. A graph G represents a pair $G = (V, E)$, where V is a set of vertices and E is a set of unordered pairs of vertices, each pair representing an edge.

Definition A2. Let $G = (V, E)$ be a graph with $|V| = N$. The adjacency matrix of the graph G , denoted $A(G)$, is an $N \times N$ symmetric matrix, defined as $a_{ij} = 1$ if $(v_i, v_j) \in E$, otherwise $a_{ij} = 0$.

Definition A3. Let $G = (V, E)$ be a graph with $|V| = N$. A k -edge coloring of G is the mapping $c : E \mapsto \{1, 2, \dots, k\}$, where $c(e)$ is the color of edge e . Furthermore, let $M_i = \{e \in E : c(e) = i\}$ be the set of edges with color i . If $M_1, M_2, \dots, M_k$ are disjoint matchings, then the k -edge coloring is said to be “proper.”

An implication of Definition A2 is that a matrix $A \in \mathbb{R}^{N \times N}$ can be interpreted as the adjacency matrix of a graph G with vertex set $V = 1, 2, \dots, N$ and edges between vertices if $a_{ij} \neq 0$. Given Definition A3, a k -edge coloring of G produces a decomposition of A into k color-specific adjacency matrices A_c , where the entries of A_c correspond to the entries of $A(G)$ with color c . Figure 17 demonstrates that a proper k -edge coloring for the graph corresponding to the TST adjacency matrix in Eq. (A1) requires three

FIG. 17. A proper k -edge coloring of an $N \times N$ TST matrix.

colors. The corresponding adjacency matrices are

$$A_1 = \begin{pmatrix} a & 0 & & 0 \\ 0 & \ddots & \ddots & \\ & \ddots & \ddots & 0 \\ 0 & & 0 & a \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & b & & \\ b & 0 & & \\ & & \ddots & \\ & & & 0 & b \\ & & & b & 0 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} 0 & & & & \\ & 0 & b & & \\ & b & 0 & & \\ & & & \ddots & \\ & & & & 0 & b \\ & & & & b & 0 \\ & & & & & 0 \end{pmatrix},$$

where A_1 is the adjacency matrix for the green edges, A_2 for the blue edges, and A_3 for the red edges. Since each A_c is 1-sparse, they can be efficiently simulated, as demonstrated in the next subsection.

2. Lie-Trotter-Suzuki approximation of e^{iAt}

A Lie-Trotter-Suzuki approximation is used to compute e^{iAt} [30].

Definition A4. Let $A_1, A_2, \dots, A_n$ be Hermitian operators. Then, for any $t \in \mathbb{R}$,

$$\lim_{m \rightarrow \infty} (e^{iA_1 t/m} e^{iA_2 t/m} \dots e^{iA_n t/m})^m = e^{i(A_1 + A_2 + \dots + A_n)t}.$$

Each computation of the product is called a Trotter step, and the total number of Trotter steps is denoted m . In the case where $A = A_1 + A_2 + A_3$,

$$e^{iAt} = e^{i(A_1 + A_2 + A_3)t} = \lim_{m \rightarrow \infty} (e^{iA_1 t/m} e^{iA_2 t/m} e^{iA_3 t/m})^m. \quad (\text{A2})$$

Since A_1 commutes with both A_2 and A_3 , $e^{iA_1 t/m}$ can be taken outside the Trotter step. Further manipulation of Eq. (A2) results in the final approximation:

$$\begin{aligned} e^{iAt} &= \lim_{m \rightarrow \infty} (e^{iA_1 t/m} e^{iA_2 t/m} e^{iA_3 t/m})^m \\ &= \lim_{m \rightarrow \infty} e^{iA_1 t} (e^{iA_2 t/m} e^{iA_3 t/m})^m \\ &= \lim_{m \rightarrow \infty} e^{iA_1 t} (e^{iA_2 t/2m} e^{iA_3 t/m} e^{iA_2 t/2m})^m \\ &= \lim_{m \rightarrow \infty} e^{iA_1 t} e^{-iA_2 t/2m} (e^{iA_2 t/m} e^{iA_3 t/m})^m e^{iA_2 t/2m}. \quad (\text{A3}) \end{aligned}$$

The QPE algorithm computes the controlled e^{iAt} operator using a sequence of controlled $e^{iA(t2^j)}$ operators, each approximated using the Lie-Trotter-Suzuki approximation in Eq. (A3), where $j = 0, 1, \dots, n_l - 1$ denotes the index of the control qubit in the l -register. For each $e^{iA(t2^j)}$ operator, a finite number of Trotter steps m_j is used to approximate the evolution. The necessary m_j to bound the error in computing $e^{iA(t2^j)}$ to a specified value ϵ can be determined as follows:

$$m_j = \lceil \sqrt{2^j} \cdot m_0 \rceil, \quad \text{where } m_0 = \left\lceil \sqrt{\frac{3(t|b|)^3}{\epsilon}} \right\rceil$$

and $t = \left(\frac{2\pi \Delta}{\lambda_{\min} 2^\delta} \right).$ (A4)

Here, t denotes the evolution time of the Hamiltonian simulation, $\lambda_{\min}$ is a lower bound on the minimum eigenvalue of A , and Δ is a scaling factor used to exactly represent $\lambda_{\min}$ on n_l binary bits. The additional variable δ is equal to 0 if all eigenvalues of A are assumed to be positive; otherwise, it is equal to 1. The user-tunable parameter ϵ directly influences the number of Trotter steps in the Hamiltonian simulation, which controls the error in the Lie-Trotter-Suzuki approximation. Figures 18–21 present the circuit diagrams for the j th controlled computation of $e^{iA(t2^j)}$ in the Hamiltonian simulation algorithm.

APPENDIX B: SUPPORTING DERIVATIONS FOR QUANTUM RESOURCE ESTIMATES

1. Derivation of asymptotic condition number scaling for the one- and two-dimensional Poisson problems

For a general $N \times N$ TST matrix,

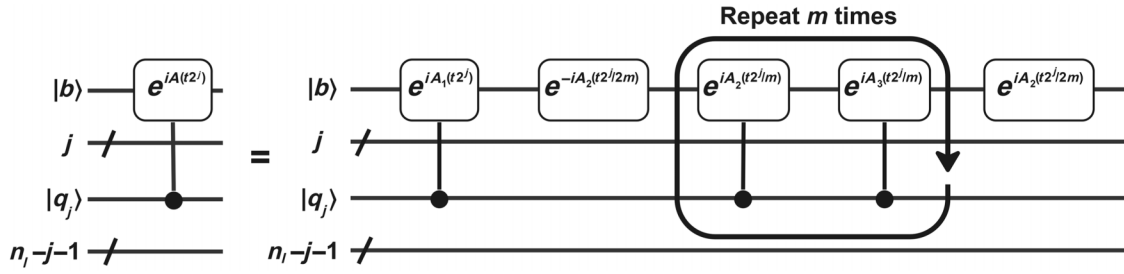
$$A = \begin{pmatrix} a & b & & \\ b & a & b & \\ & \ddots & \ddots & \ddots \\ & & b & a & b \\ & & & b & a \end{pmatrix}, \quad a, b \in \mathbb{R},$$

the eigenvalues admit a closed-form expression [31],

$$\lambda_j = a - 2b \cos\left(\frac{j\pi}{N+1}\right), \quad 1 \leq j \leq N. \quad (\text{B1})$$

Moreover, recall that the condition number of such a TST matrix is defined as $\kappa(A) = |\lambda_{\max}(A)|/|\lambda_{\min}(A)|$.

The TST matrix arising from the one-dimensional Poisson problem, given in Eq. (4), with $a = -2$ and $b = 1$,


 FIG. 18. Circuit diagram for the j th controlled computation of $e^{iA(t2^j)}$.

results in the following set of absolute eigenvalues:

$$|\lambda_j| = 2 + 2 \cos\left(\frac{j\pi}{N+1}\right), \quad 1 \leq j \leq N.$$

By the monotonicity of $\cos(\theta)$ on $[0, \pi]$,

$$\begin{aligned} |\lambda_{\max}| &= 2 + 2 \cos\left(\frac{\pi}{N+1}\right), \\ |\lambda_{\min}| &= 2 + 2 \cos\left(\frac{N\pi}{N+1}\right) = 2 - 2 \cos\left(\frac{\pi}{N+1}\right). \end{aligned}$$

For $\theta \ll 1$, corresponding to when N is large, $\cos(\theta) = 1 - \frac{1}{2}\theta^2 + \mathcal{O}(\theta^4)$. Setting $\theta = \pi/(N+1)$,

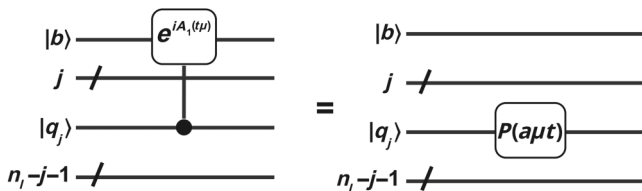
$$\begin{aligned} |\lambda_{\max}| &= 4 - \left(\frac{\pi}{N+1}\right)^2 + \mathcal{O}(N^{-4}), \\ |\lambda_{\min}| &= \left(\frac{\pi}{N+1}\right)^2 + \mathcal{O}(N^{-4}). \end{aligned}$$

Hence,

$$\kappa(A) = \frac{4 - \left(\frac{\pi}{N+1}\right)^2}{\left(\frac{\pi}{N+1}\right)^2} = \mathcal{O}(N^2).$$

Thus, the condition number of the system matrix of the one-dimensional Poisson problem grows quadratically with the grid size of the mesh; $\kappa(A) \sim (4/\pi^2)(N+1)^2$.

The TST matrix arising from the two-dimensional Poisson problem under a line-Jacobi approximation, given in


 FIG. 19. Circuit diagram for the controlled computation of $e^{iA_1(t\mu)}$.

Eq. (9), with $a = -4$ and $b = 1$, results in the following set of absolute eigenvalues:

$$|\lambda_j| = 4 + 2 \cos\left(\frac{j\pi}{N+1}\right), \quad 1 \leq j \leq N.$$

By the monotonicity of $\cos(\theta)$ on $[0, \pi]$,

$$\begin{aligned} |\lambda_{\max}| &= 4 + 2 \cos\left(\frac{\pi}{N+1}\right), \\ |\lambda_{\min}| &= 4 + 2 \cos\left(\frac{N\pi}{N+1}\right) = 4 - 2 \cos\left(\frac{\pi}{N+1}\right). \end{aligned}$$

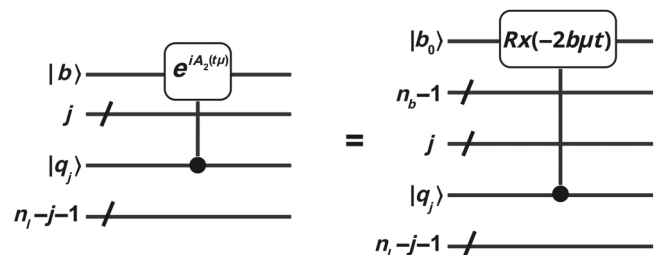
For $\theta \ll 1$, corresponding to when N is large, we can once more apply the Taylor approximation for $\cos(\theta)$. Setting $\theta = \pi/(N+1)$,

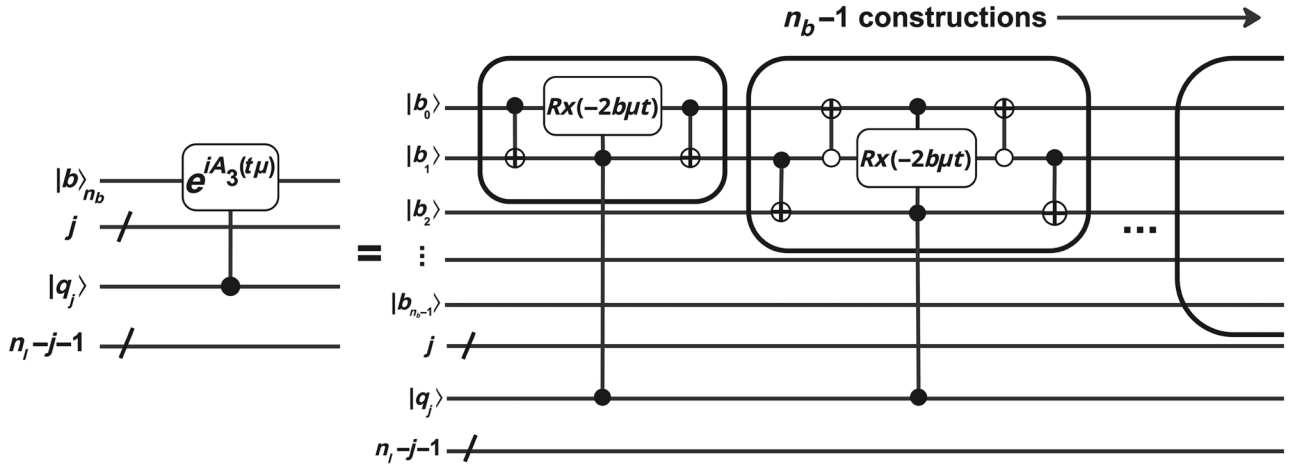
$$\begin{aligned} |\lambda_{\max}| &= 6 - \left(\frac{\pi}{N+1}\right)^2 + \mathcal{O}(N^{-4}), \\ |\lambda_{\min}| &= 2 + \left(\frac{\pi}{N+1}\right)^2 + \mathcal{O}(N^{-4}). \end{aligned}$$

Hence,

$$\kappa(A) = \frac{6 - \left(\frac{\pi}{N+1}\right)^2}{2 + \left(\frac{\pi}{N+1}\right)^2} = \mathcal{O}(1),$$

where $\lim_{N \rightarrow \infty} \kappa(A) = 3^-$. Thus, the condition number of the system matrix of the two-dimensional Poisson problem quickly approaches a constant value of 3 from below.


 FIG. 20. Circuit diagram for the controlled computation of $e^{iA_2(t\mu)}$.

FIG. 21. Circuit diagram for the controlled computation of $e^{iA_3(t\mu)}$.

2. Width of the l -register for the one- and two-dimensional Poisson problems

In Sec. II E, the l -register of the HHL circuit, excluding the additional qubit to encode the sign of the eigenvalues, was derived to have the following width:

$$n_l = \max(n_b + 1, \lceil \log_2(\kappa(A) + 1) \rceil),$$

where $n_b = \log_2(N)$.

For the system matrix of the one-dimensional Poisson problem, it was shown in Appendix B 1 that $\kappa(A) \sim (4/\pi^2)(N+1)^2$. Consequently, $\log_2(\kappa(A) + 1) > \log_2(N) + 1$, and therefore, $n_l = \lceil \log_2(\kappa(A) + 1) \rceil$.

For the system matrix of the two-dimensional Poisson problem, it was also shown in Appendix B 1 that $\lim_{N \rightarrow \infty} \kappa(A) = 3^-$, and as such, $\lceil \log_2(\kappa(A) + 1) \rceil \leq 2$. Since $\log_2(N) + 1 \geq 2$ for $N \geq 2$, we have $\log_2(N) + 1 > \lceil \log_2(\kappa(A) + 1) \rceil$, giving, $n_l = \log_2(N) + 1$.

3. General circuit complexity of the Hamiltonian simulation subroutine

The complexity of a quantum circuit is measured in the scaling of the number of controlled NOT (CNOT) gates and single-qubit gates (SQGs).

The Hamiltonian simulation algorithm requires computing a series of controlled $e^{iA(t2^j)}$ operators, where $0 \leq j \leq n_l - 1$. In Fig. 18, the Lie-Trotter-Suzuki decomposition for computing the controlled $e^{iA(t2^j)}$ operator involves a controlled $e^{iA_1(t2^j)}$ operator, an $e^{iA_2(t2^j)}$ and $e^{-iA_2(t2^j)}$ operator, and m_j controlled $e^{iA_2(t2^j)}$ and $e^{iA_3(t2^j)}$ operators.

The computation of the controlled $e^{iA_1(t2^j)}$ operator requires $1 \times$ SQG. The computation of the $e^{iA_2(t2^j)}$ operator requires $1 \times$ SQG, and the controlled operator requires $1 \times$ CNOT and $3 \times$ SQGs. Similarly, $e^{iA_2(t2^j)}$ requires $1 \times$ SQG.

The computation of the controlled $e^{iA_3(t2^j)}$ operator consists of ν constructions, where $1 \leq \nu \leq n_b - 1$. The ν th construction requires $2\nu \times$ CNOTs, $4(\nu - 1) \times$ SQGs, and $1 \times$ MCMT, where MCMT denotes a multicontrolled multitarget gate. The MCMT gate of the ν th construction can be decomposed into $2 \times$ CNOTs, $3 \times$ SQGs, and $2\nu \times$ Toffoli gates, where the Toffoli gates can be further decomposed into $6 \times$ CNOTs and $9 \times$ SQGs. Hence, the computation of the controlled $e^{iA_3(t2^j)}$ operator requires the following number of gates:

$$\begin{aligned} \text{Num. of gates} &= \sum_{\nu=1}^{n_b-1} (14\nu + 2) \times \text{CNOTs} \\ &\quad + (22\nu - 1) \times \text{SQGs} \\ &= (7n_b^2 - 5n_b - 2) \times \text{CNOTs} \\ &\quad + (11n_b^2 - 12n_b + 1) \times \text{SQGs} \\ &=: \phi. \end{aligned}$$

Combining the number of CNOTs and SQGs for each controlled $e^{iA(t2^j)}$ operator results in the following gate complexity for the full Hamiltonian simulation algorithm:

$$\begin{aligned} \text{Num. of gates} &= \sum_{j=0}^{n_l-1} 3 \times \text{SQGs} + m_j \phi \\ &= 3n_l \times \text{SQGs} + m_0 \phi \sum_{j=0}^{n_l-1} \sqrt{2^j} \\ &= 3n_l \times \text{SQGs} + m_0 \phi (1 + \sqrt{2})(2^{\frac{n_l}{2}} - 1). \end{aligned}$$

Since $\phi = \mathcal{O}(n_b^2)$ and $m_0 = \mathcal{O}(1/\sqrt{\epsilon})$, the final complexity of the Hamiltonian simulation algorithm for both CNOT gates and SQGs is $\mathcal{O}(\sqrt{2^{n_l}}/\epsilon n_b^2)$.

APPENDIX C: CONTROLLED ROTATION ALGORITHM

To compute the inverse of a Hermitian matrix A , recall that Hermitian matrices admit an eigendecomposition of the form

$$A = \sum_{i=0}^{N-1} \lambda_i |u_i\rangle\langle u_i|, \quad \lambda_i \in \mathbb{R},$$

where $\{|u_i\rangle\}$ is an independent orthonormal eigenbasis of A and λ_i are the respective eigenvalues. Given this decomposition, the inverse of A can be expressed as

$$A^{-1} = \sum_{i=0}^{N-1} \frac{1}{\lambda_i} |u_i\rangle\langle u_i|, \quad \lambda_i \in \mathbb{R}.$$

To compute the reciprocal of the eigenvalues, the HHL algorithm uses a controlled rotation algorithm. The algorithm should construct a circuit that can produce the following mapping:

$$|0\rangle \sum_{i=0}^{N-1} b_i |\lambda_i\rangle |u_i\rangle \mapsto \sum_{i=0}^{N-1} b_i \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right) |\lambda_i\rangle |u_i\rangle.$$

Consider the action of a controlled $R_Y(2\theta)$ gate, as shown in Fig. 22. This operation results in the mapping

$$|0\rangle |\lambda\rangle \mapsto (\cos(\theta) |0\rangle + \sin(\theta) |1\rangle) |\lambda\rangle.$$

If we select the rotation angle $\theta = \arcsin(C/\lambda)$, the action of the controlled $R_Y(2\theta)$ gate results in the mapping

$$\begin{aligned} |0\rangle |\lambda\rangle &\mapsto (\cos(\arcsin(C/\lambda)) |0\rangle + \sin(\arcsin(C/\lambda)) |1\rangle) |\lambda\rangle \\ &= \left(\sqrt{1 - \frac{C^2}{\lambda^2}} |0\rangle + \frac{C}{\lambda} |1\rangle \right) |\lambda\rangle. \end{aligned}$$

This is precisely the mapping required by the HHL algorithm, and thus, it provides a strategy for achieving the necessary quantum state. The strategy involves applying successive $R_Y(2\theta_j)$ rotations on the ancillary qubit, conditioned on each of the l -register qubits. Here, the rotation angles $\theta_j = \arcsin(C/\lambda_j)$, where λ_j is the eigenvalue encoded within the control qubit.

Suppose the complete set of eigenvalues of A is known *a priori*. Then, the corresponding rotation angles can be calculated, and a circuit can be constructed that applies a series of controlled R_Y gates with the appropriate rotation angles; however, in practice, the full eigenspectrum is typically unknown, and only upper and lower bounds on the eigenvalues are available.

Consider a continuous representation of the function $f(x) = \arcsin(C/x)$. The quantum phase estimation algorithm encodes an n_l -bit binary representation of the

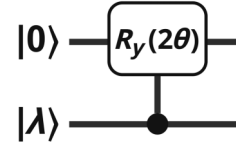


FIG. 22. Quantum circuit of a controlled $R_Y(2\theta)$ gate, where the control qubit encodes an eigenvalue and the target qubit is initially in the $|0\rangle$ state.

eigenvalues of A , denoted $\tilde{\lambda}_j$, within the l -register. Since an n_l -bit binary sequence can represent up to 2^{n_l} different sequences, we can instead consider $f : \{0, 1\}^{n_l} \mapsto \mathbb{R}$.

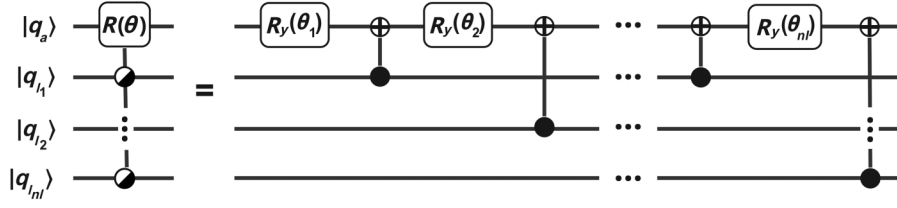
An initial approach to constructing a circuit that represents f assumes limited prior knowledge about the eigenspectrum of A . The known parameters include the minimum eigenvalue of A , $\lambda_{\min}$, a scaling constant Δ that ensures $\lambda_{\min}$ is exactly represented on n_l qubits, and the condition number of A , $\kappa(A)$. For a TST matrix, $\kappa(A) = |\lambda_{\max}(A)|/|\lambda_{\min}(A)|$. Intuitively, $\kappa(A)$ provides insight into the range between the minimum and maximum eigenvalues. Given the exact n_l -bit representation of the minimum eigenvalue, the number of l -register qubits, and the condition number of A , the set of values the given quantum circuit can represent will encompass the full eigenspectrum of A under an n_l -bit representation.

With the complete set of representable eigenvalues, the corresponding rotation angles can be computed. Given the set of representable rotation angles Θ , a crude approach to constructing a circuit that achieves the required mapping is to apply a series of R_Y gates to the ancillary qubit, uniformly controlled by each qubit in the l -register and tuned to a specific $\theta \in \Theta$. Efficient constructions of uniformly controlled rotation gates have been explored previously [27], and the corresponding circuit is shown in Fig. 23; however, even with these efficient constructions, the number of elementary single-qubit gates and CNOT gates scales exponentially as $\mathcal{O}(2^{n_l})$.

To overcome the inefficiency of representing the function $f(x) = \arcsin(C/x)$ exactly, innovative strategies have been developed that draw inspiration from numerical analysis techniques to create gate-based numerical interpolation methodologies. Notable approaches include using Newton's method with bisection search [15] and Chebyshev interpolation [24]; however, within the context of this paper, we use uniformly controlled rotation gates to achieve an exact representation of f .

APPENDIX D: VON NEUMANN STABILITY ANALYSIS

The following section presents evidence supporting the claim made in Sec. III B that the line-Jacobi scheme achieves faster convergence and more effective error

FIG. 23. Efficient circuit decomposition of a uniformly controlled R_Y rotation gate [27].

damping for high-wavenumber solution components compared to the point-Jacobi scheme.

To demonstrate this, we conduct a von Neumann stability analysis. Given a periodic domain, the numerical solution of a PDE at iteration n can be expressed in terms of a spatial Fourier series that grows in time,

$$u_i^n = \sum_m \hat{u}_m(t) e^{I k_m x}, \quad (\text{D1})$$

where $I = \sqrt{-1}$, $\hat{u}_m(t)$ is the time-varying wave amplitude of the m th Fourier mode, and $k_m = 2\pi/\ell_m$ is the wavenumber associated with wavelength ℓ_m . Since the Poisson equation is a time-independent PDE, the wave amplitude $\hat{u}_m(t)$ can be treated as time invariant. Thus, we denote the time-invariant wave amplitude as $\hat{u}_m^n$, representing the magnitude of the m th Fourier mode at iteration n . Given that the smallest wavelength that can be resolved on a mesh with spacing Δx is $2\Delta x$, the maximum wavenumber that should be considered in a Fourier analysis is $\pi/\Delta x$.

For linear finite-difference schemes, we can consider the evolution of a single Fourier component at a given iteration and use superposition to find the complete evolution of the solution. For a discrete problem with $x = i\Delta x$, where i is the index of the i th mesh node, a single Fourier mode at mesh node i and iteration n is given by

$$u_i^n = \hat{u}_m^n e^{I k_m i \Delta x} = \hat{u}_m^n e^{I \phi_{m,x}}, \quad (\text{D2})$$

where $\phi_{m,x} = k_m \Delta x$ is the nondimensional phase angle of the m th Fourier mode. Extending this to a two-dimensional PDE on a mesh with spacing Δx and Δy , a single Fourier mode at mesh node (i, j) and iteration n is given by

$$u_{i,j}^n = \hat{u}_m^n e^{I(k_x i \Delta x + k_y j \Delta y)} = \hat{u}_m^n e^{I(\phi_{m,x} + \phi_{m,y})}. \quad (\text{D3})$$

The von Neumann stability analysis substitutes a single Fourier mode into an iterative scheme and derives an expression for the amplification factor, G :

$$G := \frac{\hat{u}_m^{n+1}}{\hat{u}_m^n}. \quad (\text{D4})$$

If $|G| \leq 1$ for all $\phi \in [0, \pi]$, then a scheme is considered stable. The amplification factor can also be used to compute the rate of convergence of an iterative scheme. Let $u_{i,j}^*$

denote the exact solution to the finite-difference approximation at mesh node $x_{i,j}$. The iteration error at step n is $\varepsilon_{i,j}^n := u_{i,j}^n - u_{i,j}^*$. After n iterations, the iteration error $|\varepsilon_{i,j}^n|$ is given by $|\varepsilon_{i,j}^n| = |G|^n |\varepsilon_{i,j}^0|$, where $\varepsilon_{i,j}^0$ is the initial iteration error.

Assume we wish to reduce the iteration error by m orders of magnitude after n iterations:

$$\frac{|\varepsilon_{i,j}^n|}{|\varepsilon_{i,j}^0|} = 10^{-m} = |G|^n.$$

Taking logarithms gives

$$\begin{aligned} -m \ln(10) &= n \ln(|G|) \implies \frac{m \ln(10)}{n} \\ &= -\ln(|G|) = \ln\left(\frac{1}{|G|}\right). \end{aligned}$$

With this, we define the rate of convergence

$$R_c := \frac{m \ln(10)}{n} = \ln\left(\frac{1}{|G|}\right). \quad (\text{D5})$$

With the theory behind the von Neumann stability analysis established, the following subsections will examine both the point-Jacobi and line-Jacobi iterative schemes. For the reader's convenience, the finite-difference approximation of the two-dimensional Poisson equation using a second-order centered-in-space scheme is provided again.

$$\begin{aligned} u_{i+1,j} + u_{i-1,j} - 4u_{i,j} + u_{i,j+1} + u_{i,j-1} &= (\Delta h^2) f_{i,j}, \\ i, j &= 0, \dots, N-1. \end{aligned} \quad (\text{D6})$$

1. Point-Jacobi scheme

Consider the point-Jacobi scheme applied to Eq. (D6),

$$u_{i,j}^{n+1} = \frac{1}{4} \left(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - (\Delta h^2) f_{i,j} \right). \quad (\text{D7})$$

Using the definition of the iteration error, the point-Jacobi scheme can be rewritten in terms of $\varepsilon_{i,j}^n$,

$$\varepsilon_{i,j}^{n+1} = \frac{1}{4} \left(\varepsilon_{i+1,j}^n + \varepsilon_{i-1,j}^n + \varepsilon_{i,j+1}^n + \varepsilon_{i,j-1}^n \right). \quad (\text{D8})$$

From the definition of the iteration error, it follows that it can also be expressed in terms of a spatial Fourier series. Thus, we may insert a single Fourier mode of the iteration error, as described in Eq. (D3), into Eq. (D8), and the von Neumann stability analysis can then be carried out with respect to $\varepsilon_{i,j}^n$. For clarity, the subscript m will be omitted.

Substituting $\hat{\varepsilon}^n e^{I(k_x i \Delta x + k_y j \Delta y)}$ into Eq. (D8) gives

$$\begin{aligned} & \hat{\varepsilon}^{n+1} e^{I(k_x i \Delta x + k_y j \Delta y)} \\ &= \frac{1}{4} \left(\hat{\varepsilon}^n e^{I(k_x(i+1)\Delta x + k_y j \Delta y)} + \hat{\varepsilon}^n e^{I(k_x(i-1)\Delta x + k_y j \Delta y)} + \dots \right. \\ & \quad \left. \dots + \hat{\varepsilon}^n e^{I(k_x i \Delta x + k_y(j+1)\Delta y)} + \hat{\varepsilon}^n e^{I(k_x i \Delta x + k_y(j-1)\Delta y)} \right). \end{aligned}$$

Factoring out the common factor of $\hat{\varepsilon}^n e^{I(k_x i \Delta x + k_y j \Delta y)}$, and using the known result that $\cos(x) = \frac{1}{2}(e^{ix} + e^{-ix})$, yields the amplification factor

$$G = \frac{1}{2} (\cos(\phi_x) + \cos(\phi_y)). \quad (\text{D9})$$

For a structured mesh with uniform spacing in x and y , we then have $\phi_x = \phi_y = \phi$. Thus, Eq. (D9) reduces to

$$G = \cos(\phi).$$

Since the most prominent features in a solution correspond to modes where $\phi \ll 1$, we can use small-angle approximations. Using the small-angle approximation for $\cos(x)$,

$$\cos(\phi) = 1 - \frac{\phi^2}{2} + \mathcal{O}(\phi^4) \implies G \simeq 1 - \frac{\phi^2}{2}.$$

With the rate of convergence defined in Eq. (D5) and $\ln[1/(1-x)] = x + \mathcal{O}(x^2)$ for $x \ll 1$, we have

$$R_c = \ln \left(\frac{1}{1 - \frac{\phi^2}{2}} \right) \simeq \frac{\phi^2}{2}.$$

Finally, substituting $\phi = k\Delta x$ gives

$$R_c \simeq \frac{(k\Delta x)^2}{2}.$$

2. Line-Jacobi scheme

Consider the line-Jacobi scheme applied to Eq. (D6),

$$u_{i+1,j}^{n+1} - 4u_{i,j}^{n+1} + u_{i-1,j}^{n+1} = (\Delta h^2) f_{i,j} - (u_{i,j+1}^n + u_{i,j-1}^n). \quad (\text{D10})$$

Using the definition of the iteration error, the line-Jacobi scheme can be rewritten in terms of $\varepsilon_{i,j}^n$ as follows:

$$\varepsilon_{i+1,j}^n - 4\varepsilon_{i,j}^n + \varepsilon_{i-1,j}^n = -(\varepsilon_{i,j+1}^n + \varepsilon_{i,j-1}^n). \quad (\text{D11})$$

As with the approach used for the point-Jacobi scheme, the von Neumann stability analysis can be carried out with respect to $\varepsilon_{i,j}^n$. For clarity, the subscript m is omitted.

Substituting $\hat{\varepsilon}^n e^{I(k_x i \Delta x + k_y j \Delta y)}$ into Eq. (D11) gives

$$\begin{aligned} & \hat{\varepsilon}^{n+1} e^{I(k_x(i+1)\Delta x + k_y j \Delta y)} - 4\hat{\varepsilon}^{n+1} e^{I(k_x i \Delta x + k_y j \Delta y)} \\ & + \hat{\varepsilon}^{n+1} e^{I(k_x(i-1)\Delta x + k_y j \Delta y)} \\ & = \dots = -(\hat{\varepsilon}^{n+1} e^{I(k_x i \Delta x + k_y(j+1)\Delta y)} \\ & + \hat{\varepsilon}^{n+1} e^{I(k_x i \Delta x + k_y(j-1)\Delta y)}). \end{aligned}$$

Factoring out the common phase factor $e^{I(k_x i \Delta x + k_y j \Delta y)}$ yields

$$\hat{\varepsilon}^{n+1} (e^{I k_x \Delta x} - 4 + e^{-I k_x \Delta x}) = -\hat{\varepsilon}^n (e^{I k_y \Delta y} + e^{-I k_y \Delta y}).$$

Using the known result that $\cos(x) = \frac{1}{2}(e^{ix} + e^{-ix})$ yields the amplification factor

$$G = \frac{-\cos(\phi_y)}{\cos(\phi_x) - 2}. \quad (\text{D12})$$

For a structured mesh with uniform spacing in x and y , we then have $\phi_x = \phi_y = \phi$. Thus, Eq. (D12) is simply

$$G = \frac{-\cos(\phi)}{\cos(\phi) - 2}.$$

Since the most prominent features in a solution correspond to modes where $\phi \ll 1$, we can use small-angle approximations. Using the small-angle approximation for $\cos(x)$,

$$\cos(\phi) = 1 - \frac{\phi^2}{2} + \mathcal{O}(\phi^4) \implies G \simeq \frac{1 - \frac{\phi^2}{2}}{1 + \frac{\phi^2}{2}}.$$

Since $1/(1+x) = 1 - x + \mathcal{O}(x^2)$, we have

$$G \simeq \left(1 - \frac{\phi^2}{2}\right) \left(1 - \frac{\phi^2}{2}\right) = 1 - \phi^2 + \mathcal{O}(\phi^4).$$

With the rate of convergence defined in Eq. (D5) and $\ln[1/(1-x)] = x + \mathcal{O}(x^2)$ for $x \ll 1$, we have

$$R_c = \ln \left(\frac{1}{1 - \phi^2} \right) \simeq \phi^2.$$

Finally, substituting $\phi = k\Delta h$ gives

$$R_c \simeq (k\Delta h)^2.$$

Compared to the von Neumann stability analysis of the point-Jacobi scheme, the line-Jacobi scheme exhibits a

convergence rate that is twice as fast. It also consistently maintains a lower amplification factor across all values of ϕ , indicating superior error reduction. This is especially evident at higher wavenumbers, where the line-Jacobi scheme more effectively dampens high-frequency error propagation due to its significantly reduced amplification factor.

-
- [1] J. L. Guermond, P. Mineev, and J. Shen, An overview of projection methods for incompressible flows, *Comput. Methods Appl. Mech. Eng.* **195**, 6011 (2006).
 - [2] J. L. Hennessy and D. A. Patterson, A new golden age for computer architecture, *Commun. ACM* **62**, 48 (2019).
 - [3] N. Jansson, M. Karp, A. Podobas, S. Markidis, and P. Schlatter, Neko: A modern portable, and scalable framework for high-fidelity computational fluid dynamics, *Comput. Fluids* **275**, 106243 (2024).
 - [4] F. D. Witherden, A. M. Farrington, and P. E. Vincent, PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach, *Comput. Phys. Commun.* **185**, 3028 (2014).
 - [5] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. McRae, G. T. Bercea, G. R. Markall, and P. H. Kelly, Firedrake: Automating the finite element method by composing abstractions, *ACM Trans. Math. Software (TOMS)* **43**, 1 (2016).
 - [6] A. Montanaro, Quantum algorithms: An overview, *npj Quantum Inf.* **2**, 1 (2016).
 - [7] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, *Phys. Rev. Lett.* **103**, 150502 (2009).
 - [8] A. M. Childs, R. Kothari, and R. D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision, *SIAM J. Comput.* **46**, 1920 (2017).
 - [9] Y. Subaşı, R. D. Somma, and D. Orsucci, Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing, *Phys. Rev. Lett.* **122**, 060504 (2019).
 - [10] C. C. McGeoch, *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice* (Morgan & Claypool Publishers, San Rafael, California, United States, 2014).
 - [11] K. Sharma, S. Khatri, M. Cerezo, and P. J. Coles, Noise resilience of variational quantum compiling, *New J. Phys.* **22**, 043006 (2020).
 - [12] P. García-Molina, J. Rodríguez-Mediavilla, and J. J. García-Ripoll, Quantum Fourier analysis for multivariate functions and applications to a class of Schrödinger-type partial differential equations, *Phys. Rev. A* **105**, 012433 (2022).
 - [13] M. Lubasch, J. Joo, P. Moinier, M. Kiffner, and D. Jaksch, Variational quantum algorithms for nonlinear problems, *Phys. Rev. A* **101**, 010301 (2020).
 - [14] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles, Variational quantum linear solver, *Quantum* **7**, 1188 (2023).
 - [15] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, and S. Kais, Quantum algorithm and circuit design solving the Poisson equation, *New J. Phys.* **15**, 013021 (2013).
 - [16] S. Wang, Z. Wang, W. Li, L. Fan, Z. Wei, and Y. Gu, Quantum fast Poisson solver: The algorithm and complete and modular circuit design, *Quantum Inf. Process.* **19**, 1 (2020).
 - [17] A. M. Childs, J.-P. Liu, and A. Ostrander, High-precision quantum algorithms for partial differential equations, *Quantum* **5**, 574 (2021).
 - [18] J. M. Arrazola, T. Kalajdziewski, C. Weedbrook, and S. Lloyd, Quantum algorithm for nonhomogeneous linear partial differential equations, *Phys. Rev. A* **100**, 032306 (2019).
 - [19] Y. Sato, R. Kondo, S. Koide, H. Takamatsu, and N. Imoto, Variational quantum algorithm based on the minimum potential energy for solving the Poisson equation, *Phys. Rev. A* **104**, 052409 (2021).
 - [20] K. K. Saha, W. Robson, C. Howington, I.-S. Suh, Z. Wang, and J. Nabrzycki, Enhancing scalability and accuracy of quantum poisson solver, *Quantum Inf. Process.* **23**, 209 (2024).
 - [21] M. Ali and M. Kabel, Performance study of variational quantum algorithms for solving the Poisson equation on a quantum computer, *Phys. Rev. Appl.* **20**, 014054 (2023).
 - [22] B. Daribayev, A. Mukhanbet, and T. Imankulov, Implementation of the HHL algorithm for solving the Poisson equation on quantum simulators, *Appl. Sci.* **13**, 11491 (2023).
 - [23] Y. Cao, A. Daskin, S. Frankel, and S. Kais, Quantum circuit design for solving linear systems of equations, *Mol. Phys.* **110**, 1675 (2012).
 - [24] A. C. Vazquez, R. Hiptmair, and S. Woerner, Enhancing the quantum linear systems algorithm using Richardson extrapolation, *ACM Trans. Quantum Comput.* **3**, 1 (2022).
 - [25] B. Arseniev, D. Guskov, R. Sengupta, J. Biamonte, and I. Zacharov, Tridiagonal matrix decomposition for Hamiltonian simulation on a quantum computer, *Phys. Rev. A* **109**, 052629 (2024).
 - [26] R. Jozsa, Quantum algorithms and the Fourier transform, *Proc. R. Soc. London, A* **454**, 323 (1998).
 - [27] M. Mottonen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, Transformation of quantum states using uniformly controlled rotations, *Quantum Inf. Comput.* **5**, 467 (2005).
 - [28] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, Quantum computing with Qiskit, <https://www.ibm.com/quantum/blog/defining-and-citing-the-qiskit-sdk>.
 - [29] A. G. Özbay, A. Hamzehloo, S. Laizet, P. Tzirakis, G. Rivos, and B. Schuller, Poisson CNN: Convolutional neural networks for the solution of the Poisson equation on a Cartesian mesh, *Data-Centric Eng.* **2**, e6 (2021).
 - [30] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, United Kingdom, 2010).
 - [31] S. Noschese, L. Pasquini, and L. Reichel, Tridiagonal Toeplitz matrices: Properties and novel applications, *Numerical Linear Algebra Appl.* **20**, 302 (2013).