Quantum Information Processing (2024) 23:209
https://doi.org/10.1007/s11128-024-04420-y

Enhancing scalability and accuracy of quantum poisson solver

Kamal K. Saha[1,5] · Walter Robson[2] · Connor Howington[1] · In-Saeng Suh[1,3] · Zhimin Wang[4] · Jaroslaw Nabrzyski[1]

Received: 25 June 2023 / Accepted: 7 May 2024

Abstract
The Poisson equation has many applications across the broad areas of science and engineering. Most quantum algorithms for the Poisson solver presented so far either suffer from lack of accuracy and/or are limited to very small sizes of the problem and thus have no practical usage. In this regard, our previous work (Robson in 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), 2022) showed a proof-of-concept demonstration in advancing quantum Poisson solver algorithm and validated preliminary results for a simple case of 3×3 problem. In this work, we delve into comprehensive research details, presenting the results on up to 15×15 problems that include step-by-step improvements in Poisson equation solutions, scaling performance, and experimental exploration. In particular, we demonstrate the implementation of eigenvalue amplification by a factor of up to 2^8, achieving a significant improvement in the accuracy of our quantum Poisson solver and comparing that to the exact solution. Additionally, we present success probability results, highlighting the reliability of our quantum Poisson solver. Moreover, we explore the scaling performance of our algorithm against the circuit depth and width, demonstrating how our approach scales with larger problem sizes and thus further solidifies the practicality of easy adaptation of this algorithm in real-world applications. We also discuss a multilevel strategy for how this algorithm might be further improved to explore much larger problems with greater performance. Finally, through our experiments on the

✉ Kamal K. Saha
ksaha@nd.edu; kksaha@lps.umd.edu

1 Center for Research Computing, University of Notre Dame, Notre Dame, IN 46556, USA

2 Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA

3 National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA

4 Faculty of Information Science and Engineering, Ocean University of China, Qingdao 266100, China

5 Laboratory for Physical Sciences, 8050 Greenmead Drive, College Park, Maryland 20740, USA

Published online: 27 May 2024

IBM quantum hardware, we conclude that though overall results on the existing NISQ hardware are dominated by the error in the *CNOT* gates, this work opens a path to realizing a multidimensional Poisson solver on near-term quantum hardware.

Keywords Poisson equation · Poisson solver algorithm · Quantum circuit · Quantum algorithm · Quantum error mitigation

1 Introduction

The Poisson equation is a second-order partial differential equation widely used in various fields of science and engineering. In general, in order to solve the Poisson equation numerically, projection methods such as collocation, spectral, and boundary element methods as well as finite-difference methods [2] are used. The core of these methods is to approximate the solution of the Poisson equation as the solution of a linear system. However, since the dimension of the linear system obtained from the discrete Poisson equation is generally very large, solving such a system demands much computational time. Therefore, the Poisson equation is a problem well suited to quantum computing, a faster and more powerful computation paradigm [3] than classical computing.

A series of quantum algorithms [4–23] have been developed to solve linear equation systems, which have shown significant speedups over their classical counterparts. Recently, variational quantum algorithms (VQAs) [24–27], which have already shown some promise for use on so-called noisy intermediate-scale quantum (NISQ) devices [28], are adopted to solve the Poisson equation [19, 20]. From the experimental point of view, while VQAs-based approaches have some advantages, such as generally using relatively shallow quantum circuits or requiring fewer quantum measurements, they still have challenges in optimizing a set of parameters, especially on larger problems [19]. In addition, instead of producing the direct solution of the Poisson equation, these methods rely on an expectation of certain observables limiting them to be coupled with other general problems and thus may have a limited use case. A non-VQA-based iterative method for the HHL algorithm [12] has been proposed to solve linear system of equations in Ref. [21]. Even though they obtained a more accurate solution by increasing the number of iterations with the same number of measurements, they still have challenges in improving the error convergence speed compared to the state vector calculations. However, in the context of the quantum circuit model, Cao et al. [22] first used the original HHL algorithm [12] to solve the Poisson equation. Later, our co-author Wang et al. [23] pointed out a bottleneck of Cao's algorithm which was the cost associated with the function used for controlled rotation. It was later reformulated by Wang et al. reducing its cost from $O(m^4)$ to $O(m^3)$, where m is the number of qubits of input register. Cao's and Wang's approaches are reviewed in Appendix A.

Even though these past works, including Cao's and Wang's methods, improved the quantum algorithm and circuit for the Poisson solver, they still either suffered from lack of accuracy and/or were limited to demonstrating only a very small size of the problem, and thus, their practical usage is limited. Some of these works focus on minimizing the error in their approaches or in the overall solutions without directly presenting the actual or direct solution of the Poisson equation [19–21], or some others appeared to suggest the feasibility of their methods on quantum hardware without even clearly discussing or validating their works on any hardware [19, 20].

Our previous work [1] serves as an initial outline of our research on the quantum Poisson equation solver. It presents a proof-of-concept demonstration of our quantum Poisson solver algorithm and validates preliminary results for a simple case of 3×3 problem. Here we delve into comprehensive research details, presenting the results on up to 15×15 problems that include step-by-step improvements in Poisson solutions, scaling performance, and experimental exploration.

Specifically, by solving different sizes of problems, this paper demonstrates the advancement of our algorithm for solving the Poisson equation in several aspects: (1) Improve the precision of phase estimation by increasing the accuracy of the eigenvalues. Unlike the previous approach [23] where only the integer part of the eigenvalues was encoded, we implement non-truncated eigenvalues through eigenvalue amplification. We will see that this has a clear impact in drastically reducing the error in the solution of the Poisson solver and compare that to the exact solution; (2) The rotation angles are calculated with full accuracy, which is also essential for ensuring the overall accuracy of the solution; (3) We present and analyze success probability results, highlighting the reliability of our quantum Poisson solver; (4) Without compromising any accuracy of the algorithm, during the run time, our implementation uses an optimized number of qubits representing the rotation angles. We also optimize the *CNOT* gates usage, which is one of the primary sources of error in an experiment; (5) Solutions of the Poisson equations with larger problem size to 7×7 and 15×15 are demonstrated. In fact, our implementation with dynamic allocation of qubits in different segments of the algorithm ensures easy adaptation of this method for solving real-world problems; (6) Additionally, we explore the scaling performance of our algorithm against the circuit depth and width, demonstrating how our approach scales with larger problem sizes. We also offer a multilevel strategy for how this algorithm might be further improved to explore much larger problems with greater performance; (7) The possibilities and difficulties of implementing the algorithm on real quantum hardware are discussed for the first time. This also includes experimenting with the circuit mapping, error mitigation, etc. on the NISQ devices and presenting a vision for near-term hardware; (8) Finally, the algorithm is implemented using the Qiskit package [29], which would bring advantages for practical use. We believe all these aspects are necessary to advance the study of quantum Poisson solvers.

We also want to make it clear that in this work our main focus is to demonstrate the advancement of our hybrid algorithm that accurately simulates the Poisson equation with realistic problem sizes and explore the experimental feasibility of those problems. In particular, we aim to push the scalability of our proposed Poisson solver to larger practical problems on both simulators and real quantum devices. While testing these problems, we also identify the key limiting factors against applying the algorithm to large problems and implement some optimization methods in terms of the number of qubits and gates. We explain that with the current state of the technology, it is difficult to realize a complete quantum description of the algorithm due to its high resource costs. However, we discuss pathways to further improve this hybrid approach in both simulation and experimental environments.

For demonstrating the circuit, we present both simulated and experimental results, discuss the sources of errors, and eliminate them. The Matrix Product State (MPS) simulator is used for simulation, and the experiment is done on IBM's *ibmq_manila* and *ibmq_brooklyn* quantum backends [30]. We examine the measurement error mitigation on a small system and also discuss how the overall results of the Poisson equation on the currently available quantum hardware are dominated by the error in the *CNOT* gates.

We have extended the existing algorithm from Wang et al. beyond a single proof of concept to a fully dynamic, scalable body of work that can be used for numerous applications in mixed computing algorithms. Wang's QRUNES [31]-based machine instructions have been abstracted to more usable Qiskit functions, allowing us to perform fine-tuning of different register sizes so that we can identify key areas of inaccuracy and compare qubit tradeoffs. This is important because the primary limiting factor in accuracy is the total number of qubits in the circuit, and qubits are at a premium in NISQ hardware. Our code is readable and easily usable, allowing a true "black box" approach to be taken to solving the most computationally intensive part of Poisson applications.

The paper is organized as follows. In Sect. 2, we adopt the finite- difference method to discretize the Poisson equation to obtain a linear system. In Sect. 3, we describe the quantum algorithm and circuit design for each module and our algorithm in detail. In Sect. 4, we explain the algorithm improvement and circuit optimization. We show simulated results of different sizes of problems and their improvements, and we discuss more about algorithm scaling and success probability in Sect. 5. In Sect. 6, we demonstrate our improved quantum circuit for the Poisson solver on IBM quantum hardware and discuss error mitigation. Finally, we conclude our works in Sect. 7.

2 Overview of the problem

The goal of this work is to implement an efficient quantum algorithm solving the multidimensional Poisson equation with boundary conditions. Let us consider the Poisson equation defined in an open bounded domain $\Omega \subset \Re^d$, where d is the number of spatial dimensions.

$$-\nabla^2 v(x) = b(x), \; x \text{ in } \Omega \tag{1}$$

$$v(x) = 0, \; x \text{ on } \delta\Omega \;\; \Omega = (0, 1)^d \tag{2}$$

where $\delta\Omega$ is the boundary of Ω and $b(x)$ is a given smooth function representing different problem applications, such as charge or velocity distribution. One way to solve this problem is to discretize Ω to $N' = N + 1$ grid points in each dimension, where N is an exponent of base 2. The solution $v(x)$ is a vector of $(N - 1)^d$ entries.

In this work, we focus on the one-dimensional Poisson equation with Dirichlet boundary conditions. Using the central difference approximation to discretize the

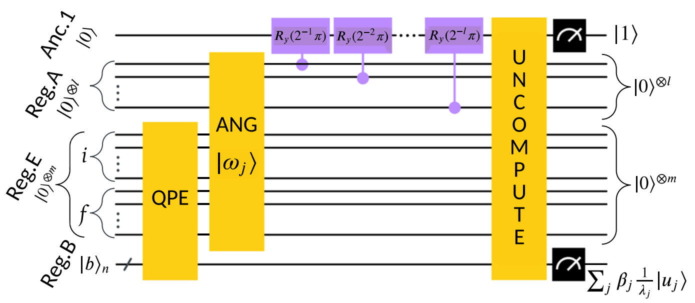

Fig. 1 Overall circuit representation of the algorithm for solving the one-dimensional Poisson equation. The numbers of qubits of registers A, E, and B are l, m, and n, respectively. Here $m = i + f$, where i and f number of qubits in reg. E hold the integer and fractional parts of the eigenvalue. $|\omega_j\rangle$ is the angular coefficient evolved from the approximated eigenvalue $|\lambda_j\rangle$, the output of the QPE. The input $|b\rangle_n = \sum_{i=1}^{2^n-1} b_i|i\rangle$ is prepared and stored in register B

second-order derivative, Eq. (1) can be converted to finite-difference form as

$$A \cdot \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-1} \end{pmatrix} = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-1} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{N-1} \end{pmatrix} \qquad (3)$$

We now have the $N-1$ linear equation system, i.e., $A|v\rangle = |b\rangle$ to be solved. Here A is a Hermitian matrix, and due to the zero boundary condition, its dimension is $(N-1) \times (N-1)$, and the mesh size h equals $1/N$.

The eigenvalues of A are $\lambda_j = 4N^2 \sin^2(j\pi/2N)$, and its corresponding eigenvectors are $u_j(k) = \sqrt{2/N}\sin(j\pi k/N)$ [32].

The best classical algorithms for solving this problem run polynomially with matrix size [33], so the run time increases exponentially with the dimension of the problem. In this paper, a quantum algorithm is used to produce a quantum state representing the normalized solution of the problem. Since this technique runs in polylog time, the curse of dimensionality can be broken. Thus, we can solve the linear system of equations based on the HHL algorithm [12]. Our algorithm exploits properties of matrix A to efficiently implement the HHL algorithm by simulating the unitary operator e^{iAt}. Though we are presenting an algorithm for the one-dimensional Poisson equation, this can be easily extended to the d-dimensional case [19, 23] as

$$A^{(d)} = \underbrace{A \otimes I \otimes \cdots \otimes I}_{d} + I \otimes A \otimes I \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes I \otimes A. \qquad (4)$$

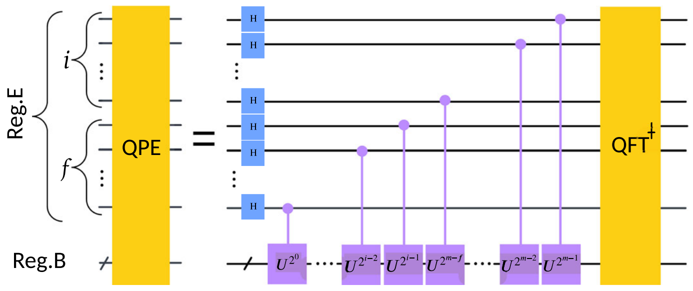

Fig. 2 Overall circuit for quantum phase estimation (QPE) that uses both integer and fractional parts of the eigenvalues through reg. E. U^{2^k} represents the unitary operator of exp $(i2\pi A/2^{m-k})$. QFT† represents the inverse quantum Fourier transform

with the exponential $A^{(d)}$ expressed in the form

$$e^{iA^{(d)}t} = \underbrace{e^{iAt} \otimes e^{iAt} \otimes \cdots \otimes e^{iAt}}_{d}. \tag{5}$$

So, the quantum circuit simulating $e^{iA^{(d)}t}$ is just the parallel execution of the circuit simulating e^{iAt} along the d dimension. In the following sections, we will focus on the one-dimensional Poisson equation.

3 Quantum algorithm and circuit design

The overall circuit diagram of our algorithm for solving the one-dimensional Poisson equation is presented in Fig. 1 [1, 23]. As the figure shows, the algorithm consists of three stages: phase estimation, controlled rotation, and uncomputation. Its circuit diagram has three main registers—reg. B, reg. E, and reg. A.

- Reg. B is used to encode the coefficients of the right-hand side of Eq. (1). Its number of qubits is $n = \lceil \log(N') \rceil$, where N' is defined in Sect. 2.
- Reg. E is used to store the approximated eigenvalues of matrix A. Its number of qubits is $m = i + f$, where the first $i = 2n + 2$ qubits hold the integer part and the remaining f qubits the fractional part of the eigenvalue.
- Reg. A is used to store pre-calculated angular coefficients for the controlled rotation operation. Its number of qubits is chosen to be $l \geq m$.

In this work, we assume that the input state $|b\rangle$ of reg. B is prepared as $\sum_i b_i |i\rangle$, where b_i is the value on the right-hand side of Eq. (3), and $|i\rangle$ is the computational basis [34]. That is, the input $|b\rangle$ contains the prerequisite state vector, the problem that we are trying to solve, which we then entangle with the approximated eigenvalues λ_j on reg. E. The output of the algorithm thus is a quantum state that encodes the solutions of the Poisson equation as probability amplitudes on reg. B. Thus, this circuit is a process of quantum state preparation, with the output written as $|v\rangle = A^{-1}|b\rangle = \sum_i \alpha_i |i\rangle$, where α_i is the value of the solutions of the Poisson equation after normalization.

We will now discuss a few key steps of the algorithm.

3.1 Phase estimation

Through the quantum phase estimation (QPE) circuit shown in Fig. 2, we estimate the eigenvalues of the discretized matrix A and entangle the states encoding the eigenvalues with the corresponding eigenstates [35]. We will now discuss how the quantum states evolve through the QPE section of the circuit. The initial state of reg. E and reg. B is

$$|0\rangle^{\otimes m}|b\rangle = \sum_{i=1}^{2^n-1} b_i|0\rangle^{\otimes m}|i\rangle = \sum_{j=1}^{2^n-1} \beta_j|0\rangle^{\otimes m}|u_j\rangle. \tag{6}$$

where $|i\rangle$ is the computational basis and $|u_j\rangle$ is the jth eigenvector of matrix A. Then the Hadamard gates across reg. E prepare a uniform superposition state, which the sequence of controlled U^{2^m} operation evolves as follows:

$$\begin{aligned}\sum_{k'=0}^{2^m-1}(|k'\rangle\langle k'| \otimes U^{k'}) \cdot \frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1}|k\rangle \otimes \sum_{j=1}^{2^n-1}\beta_j|u_j\rangle \\ = \sum_{j=1}^{2^n-1}\beta_j\left[\frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1} e^{2\pi i\frac{\lambda_j}{2^m}k}|k\rangle\right]|u_j\rangle.\end{aligned} \tag{7}$$

Note that the state in the square bracket of Eq. 7 is simply the output of the quantum Fourier transform acting on the state $|\lambda_j\rangle$, so after the application of the inverse Fourier transform the states evolve to $\sum_{j=1}^{2^n-1}\beta_j|\lambda_j\rangle|u_j\rangle$. This entangles the eigenvalues $|\lambda_j\rangle$ with the eigenstates $|u_j\rangle$ from reg. B.

Though there are methods [36, 37] available for simulating the time evolution of e^{iAt}, Wang et al. take advantage of using specific properties of the tri-diagonal matrix A to reduce the complexity of the algorithm. They first decompose the unitary operator e^{iAt} with a Hermitian matrix S (S being an orthogonal matrix composed of the eigenvectors of A) and then diagonalize it via the sine transform, and finally use phase kickback [38] to operate it on the state $|b\rangle$. We adopt Wang's approach for phase estimation; its detailed circuit composition is available in Ref. [23].

3.2 Phase verification

An eigenvalue problem involving an arbitrary unitary operator A and its eigenvector $|v_j\rangle$ and eigenvalue λ_j satisfies $A|v_j\rangle = \lambda_j|v_j\rangle$. Using this, we can verify the correctness of the QPE part of the circuit. In fact, this would also implicitly verify the phase kickback operation, which, through the controlled U operations (in Fig. 2), entangles the eigenvalues of matrix A with the eigenstates associated with the input in reg. B. We can think of reg. B as containing the problem we are trying to solve for the HHL algorithm. Each eigenvalue of matrix A is associated with an eigenvector, so the first way to perform the verification is to input the individual eigenvectors as the input to reg. B and then measure reg. E before the controlled rotations. For example, a Qiskit simulation with $A(3\times 3)$ in Eq. 3 acting on its eigenstates $|v_j\rangle = \begin{pmatrix} 1 \\ \sqrt{2} \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -\sqrt{2} \\ 1 \end{pmatrix}$ produces the eigenvalues $\lambda_j = 9, 32, 54$ in binary (using only the integer part for simplicity) with 100% probability; this is shown in Fig. 3 (a-c). Further, for any input with an arbitrary combination of eigenvectors, for example, $|v\rangle = \frac{1}{2}|v_1\rangle + \frac{1}{\sqrt{2}}|v_2\rangle + \frac{1}{2}|v_3\rangle$, the QPE circuit produces the combination of the eigenvalues with correct probabilities, as presented in Fig. 3d.

3.3 Controlled rotation

After the phase estimation $\sum_{j=1}^{2^n-1} \beta_j |\lambda_j\rangle|u_j\rangle$ is obtained on regs. B and E, we perform the linear map taking the state of $|\lambda_j\rangle$ to $(1/\lambda_j)|\lambda_j\rangle$. This process consists of two parts: calculating the rotation angular coefficients and performing the controlled R_y operation. The probability amplitude of $1/\lambda_j$ can be produced by implementing the controlled R_y rotation, that is, $R_y(2\theta_j)|0\rangle = \cos\theta_j|0\rangle + \sin\theta_j|1\rangle$, where the rotation angle θ_j can be expressed in terms of λ_j as

$$\sin\theta_j = 1/\lambda_j, \tag{8}$$

which can be rewritten as

$$\cot\theta_j = \sqrt{\lambda_j^2 - 1}, \quad \theta_j \in (0, \pi/2). \tag{9}$$

Taking $\theta_j = \omega_j\pi$, Eq. (9) becomes

$$\omega_j = \frac{1}{\pi}\text{arccot}\big(\sqrt{\lambda_j^2 - 1}\big), \quad \omega_j \in (0, 1/2), \tag{10}$$

where ω_j is the rotation angular coefficient. In this hybrid approach, we prepare ω_j classically and encode them into the circuit.

For an angular coefficient state $|\omega_j\rangle$ in reg. A, the binary representation can be written as $\omega_j = \omega_{j_1}\omega_{j_2}\cdots\omega_{j_l} = \sum_{k=1}^{l} 2^{-k}\omega_{j_k}$. Then, using $R_y(2\theta_j) = e^{-i\theta_j Y}$, the R_y rotation can be expressed as [39],

$$\begin{aligned} R_y(2\omega_j\pi) &= e^{-i(\sum\limits_{k=1}^{l}\frac{\omega_{j_k}}{2^{-k}})\pi Y} \\ &= \prod_{k=1}^{l} e^{-i\frac{\omega_{j_k}}{2^k}\pi Y} = \prod_{k=1}^{l} R_y^{\omega_{j_k}}(\frac{\pi}{2^{k-1}}). \end{aligned} \tag{11}$$

where ω_{j_k} are the control qubits in reg. A. For a given k, if the bits of ω_{j_k} for all j are zero, then the corresponding $R_y^{\omega_{j_k}}$ operation has no effect on the Ancillary register. This allows us to further optimize the circuit by removing any control qubits with bit $\omega_{j_k} = 0$ from reg. A. This is further discussed in the next section.

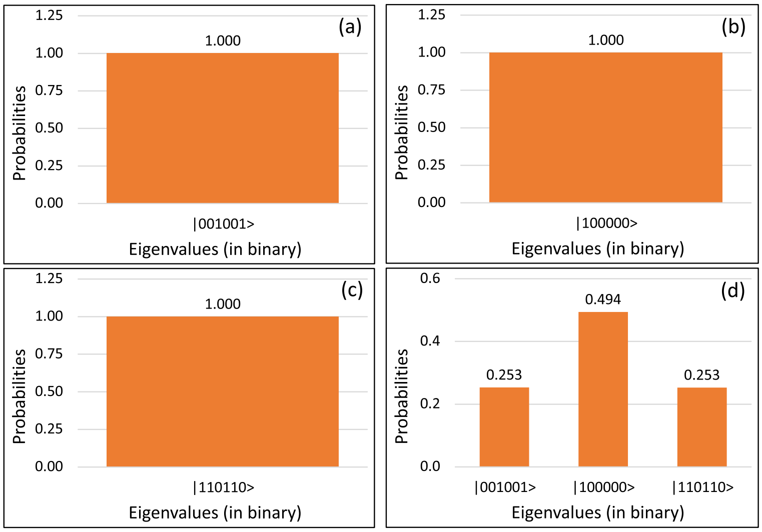

Fig. 3 Verification of the QPE section of the circuit. Panels **a–c** show that for a given eigenstate of $A(3 \times 3)$, the QPE produces the corresponding eigenvalue with 100% probabilities. Panel **d** shows that inputting the combinations of eigenstates with arbitrary weights produces eigenvalues with similar weights

The workflow used in this work follows several steps and is presented in Algorithm 1.

```
Algorithm 1: Quantum Poisson Solver
Data: Input state $\sum_i b_i |i\rangle$ in reg. B. For QPE, assign a number of qubits for the integer and fractional
      parts of $|\lambda_j\rangle$ in reg. E. Also, assign the initial number of qubits for reg. A for $|\omega_j\rangle$
Result: Get the solution $|v\rangle = A^{-1}|b\rangle$ in terms of probability amplitudes
Algorithm Start:
  1. Prepare the initial quantum state: $\sum_{j=1}^{2^n-1} \beta_j |0\rangle^{\otimes m} |u_j\rangle$
  2. Use QPE algorithm on regs. B and E. This algorithm applies several Hamiltonian simulations of
     $U = e^{iAt}$ with $t = 2\pi \frac{1}{2^n} 2^k,\ k = 0, ..., n-1$, to reg. B and entangles the eigenvalues $\lambda_j$ of matrix
     $A$ in reg. E with the eigenstates $|u_j\rangle$ in reg. B. The system has now the state: $\sum_{j=1}^{2^n-1} \beta_j |\lambda_j\rangle |u_j\rangle$
  3. Apply the controlled rotation which consists of two parts: preparing the rotation angular coefficients
     $|\omega_j\rangle$ in reg. A and performing the controlled $R_y(2\omega_j \pi)$ operation on the ancillary qubit
  4. Uncompute QPE and $|\omega_j\rangle$ operations on regs. A, E, and B
  5. Measure the ancillary qubit. If the measurement of the qubit results in state $|1\rangle$, the algorithm
     successfully transforms reg. B into the solution $|v\rangle = A^{-1}|b\rangle = \sum_{j=1}^{2^n-1} \beta_j \frac{1}{\lambda_j} |u_j\rangle$. Otherwise,
     the algorithm has to be restarted
  6. Take a repeated number of trials for a good sampling
  7. Take the sum of individual successful states $|v_i\rangle$ and derive final probability amplitudes as
     $\sqrt{v_i / \sum_i v_i}$
```

4 Algorithm improvements, circuit optimization, and challenges

Wang's method has already reduced its complexity to $O(m^2)$ qubits and $O(m^3)$ operations [23]. After implementing the algorithm as a quantum circuit, we look for options for further improving it so that even with a limited number of qubits on the quantum hardware, we are able to more accurately solve the Poisson equation for a realistic problem size, i.e., with a larger matrix A. Below, we discuss some shortcomings of the existing approaches and the ways we improve them:

4.1 Eigenvalue amplification

The first source of inaccuracy in the existing algorithm [22, 23] is the truncation of the eigenvalues of matrix A in the phase estimation. Wang's implementation uses only the integer eigenvalues of the A matrix, presumably in order to save qubits. As more qubits become available in quantum devices, however, we can improve accuracy by using non-truncated values. Therefore, we extend the algorithm by taking into account both the integer and fractional parts of the eigenvalues. This is done by amplifying the eigenvalue by a factor of 2^f, which shifts the decimal point of the binary λ_j to right by an integer f. For example, for a given $\lambda_j = 10111.11011011101011$, with no amplification, 2^4 amplification and 2^8 amplification, the circuit carries $\lambda_j =$ 10111, 101111101, and 1011111011011, respectively. Essentially, when we include fractional part for the eigenvalue, we are encoding a bitshifted/amplified eigenvalue that is still an integer but contains bits of the fractional part. This way, by using a large f, one actually includes more bits of the fractional part of the eigenvalue, and the shifted position of the decimal point of the eigenvalue is adjusted by a normalization factor 2^{-f} in the controlled R_y operation of the circuit to match. Due to the dynamic nature of our code, we are able to experiment using any number of bits on the eigenvalues, taking a more accurate representation of the critical matrix A for our computation.

4.2 Rotation angular coefficient accuracy

The second source of inaccuracy is in the calculation of the rotation angular coefficient. The previous method [23] omitted the subtrahend 1 under the square root in Eq. (10); we instead include it. Additionally, we retain full accuracy in the calculation of the rotation angular coefficients by using the full eigenvalues. Furthermore, our implementation allows us to dynamically expand the number of qubits to represent rotation angular coefficients with higher accuracy. Finally, we are able to use the optimum number of qubits based on the convergence of the error in the solution, which is discussed in the next section.

4.3 Optimize the number of qubits used for rotation angles

While we initially presented a rotation on all bits of reg. A, in practice, this is not necessary. In Eq. (11), if the bits of ω_{j_k} for all j are zero for a given k, then the corresponding

$R_y^{\omega_{jk}}$ operation has no effect on the Ancillary register. In other words, if there is no information conveyed on a given qubit in reg. A by any of the rotation angular coefficients, then we can safely omit that qubit and its rotation without impacting the results. Intuitively, this makes sense as the controlled rotations do not happen if a given control bit is 0. Imagine a case where our $\omega_j = 0.0000100110, 0.0000001010, 0.0000000101$. The first four qubits, as well as the sixth qubit, are 0 for all ω_j, so the respective $R_y(2^{-1}\pi)$, $R_y(2^{-2}\pi)$, $R_y(2^{-3}\pi)$, $R_y(2^{-4}\pi)$, and $R_y(2^{-6}\pi)$ rotations never happen. We have no need to include these controlled rotations nor the qubits in reg. A that they correspond to. This allows us to further optimize the circuit by removing any control qubits with bit $\omega_{j_k} = 0$ from reg. A. As a result, though at the beginning we chose $l \geq m$ qubits for reg. A, after the circuit optimization, the register has fewer than l qubits.

4.4 Optimize CNOT gates usage

The rotation angular coefficient allows us to entangle the prepared state on reg. E with the controlled rotation on reg. A. This is achieved with multi-controlled multi-target (MCMT) gates controlled on the binary expansion of the eigenvalues on reg. E. However, MCMT gates transpile to many *CNOT* gates, which carry significant errors into the experiment. We want to minimize the number of controlled bits in this operation. At the end of the phase estimation, the qubits of reg. E are entangled; thus, the phase information can be accessed through fewer qubits in reg. E. This allows us to control our encoding of the rotation angular coefficients on only the unique most significant bits of reg. E. For example, in the 3×3 case, if our eigenvalues are 9, 32, 54 (taking only the integer part for simplicity), then their binary encodings are 001001, 10000, 110110, respectively. It is then evident that the two most significant bits of the binary encodings are enough to differentiate between different eigenvalues: 00, 10, and 11. Controlling the angular rotations on only these two qubits allows us to reduce the number of *CNOT* gates in the circuit significantly.

4.5 Classical versus quantum approach to rotation angular coefficient

Preparing the angular coefficient state $|\omega_j\rangle$ (see Eq. 10 here) using a quantum circuit presented in Ref. [23] has a cost that grows exponentially with the problem size. This proves to be a challenge because the current state of the simulator, and quantum hardware supports a limited number of qubits. Therefore, though from a theoretical standpoint, the quantum approach to $|\omega_j\rangle$ is appealing, from a practical standpoint its classical treatment is the viable option. This is particularly true because our main goal is to scale the Poisson solver to realistic problem sizes, which requires us to appropriately allocate computational resources.

Therefore, in this work, we pre-calculate ω_j classically and encode them into the circuit. These two steps of the workflow are already fast and for large number of w_j's; the total processing time can be significantly reduced by parallelizing them over j on CPU or GPU hardware and thus avoids exponential processing time growth on large problems. Please note that this computation is required only once, independent of the

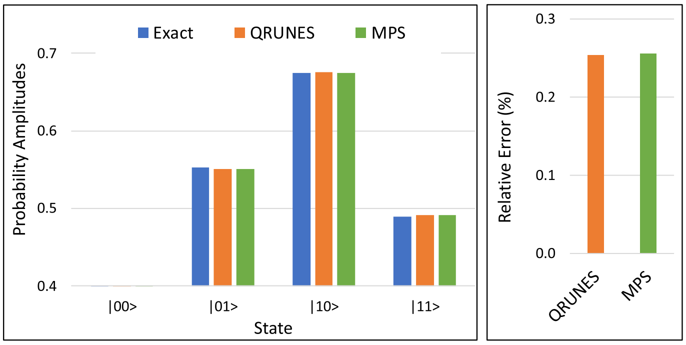

Fig. 4 (Left) Comparison of the Matrix Product State (MPS)-based simulated solution of the Poisson equation with exact and existing QRUNES [23] solutions for a 3 × 3 problem size. (Right) The relative error in QRUNES and MPS simulated outputs with respect to the exact result

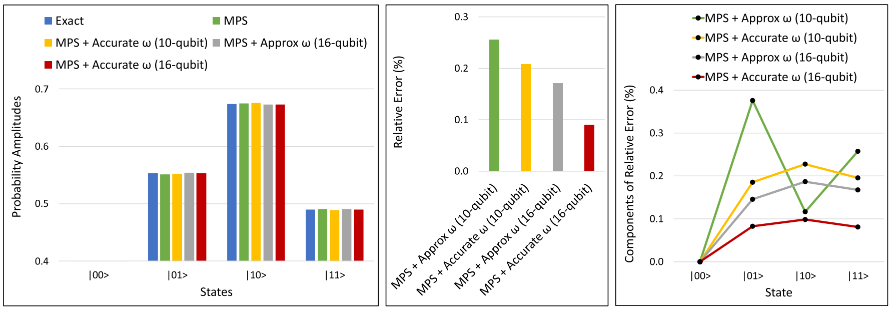

Fig. 5 (Left) Step-by-step improvement in MPS-simulated solution by using accurate rotation angular coefficients ω_j and encoding them into up to 16 qubits on reg. A. (Middle) relative error in the improved MPS-simulated solutions with respect to the exact result. (Right) relative errors at the level of individual states

number of repeated shots, and we make the process substantially more efficient by dynamically calculating it for any problem size.

However, even within this hybrid approach, dealing with very large problems, e.g., encoding 10^8 values of ω_j for a $10^8 \times 10^8$ matrix, would be challenging. Such a large problem would make the circuit depth prohibitively large from an experimental standpoint. A multilevel solution to this problem is discussed in the subsequent section.

5 Simulated results and discussions

We constructed our algorithm in the Python programming language using IBM's Qiskit package [29]. This allowed us to create our circuit in a modular fashion as well as use some of Qiskit's abstractions, such as the MCMT gate and simple implementations of quantum Fourier transform.

To the best of our knowledge, previous work has not included the simulation of a solution of the one-dimensional quantum Poisson equation beyond a 3 × 3 matrix

A. However, here we present the simulation of solutions of much larger problems, that is, for larger sizes of A. In fact, we will present that our algorithm and its circuit representation are capable of dynamically controlling problem size in NISQ devices. For simulation, we use IBM's Matrix Product State (MPS) simulator since it supports a relatively large number of qubits (up to 100) necessary for presenting the circuit for larger problems while also maintaining reasonable accuracy. In this section, we analyze the source of error in the solution and accordingly demonstrate step-by-step improvements in the algorithm that secure higher accuracy in the solution.

Reproduce existing results As shown in Fig. 4 (left), we first produce the solution of a 3×3 problem with $|b\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{2}(|10\rangle + |11\rangle)$ being the right-hand side of the Poisson equation. In order to compare this with the existing QRUNES results [23], we use their same inputs, that is, only the integer part of λ_j and the approximated ω_j encoded on 10 qubits of reg. A. Notice that in Fig. 4 (left), we show the vertical axis starting from 0.4, so that even any tiny differences in the heights of the histograms are clearly visible. Though our MPS-based simulated solution shows an excellent agreement with QRUNES, there are some discrepancies compared to the exact solution. To analyze further, the accuracy of our MPS-based result is depicted using the relative error in the MPS solution with respect to the exact result and here the relative error, e. g., for MPS, is defined as $\|\text{Exact} - \text{MPS}\|_2/\|\text{Exact}\|_2$ [40] and shown in the right panel of Fig. 4. Relative errors in both QRUNES and MPS are virtually equivalent.

Improvements in results To improve our MPS-based result presented above and have a better agreement with the exact solution, we made the following two improvements: First, we used the accurate formula for ω_j given in Eq. (10); then, we encoded these values in up to 16 qubits on reg. A. The results are shown in the left panel of Fig. 5, which displays the gradual improvements in the solutions as compared to the exact result. The improvements in solutions are clearly visible through the relative error presented in the middle panel of Fig. 5, and its right panel explicitly shows the components of those relative errors.

Next, we extend the problem size to 7×7 with $|b\rangle = \frac{1}{4}(|001\rangle + |010\rangle + |011\rangle + |100\rangle) + \frac{1}{2}(|101\rangle + |110\rangle + |111\rangle)$ and further investigate the effects of using a more precise ω_j by increasing the number of qubits in reg. A. As clear in Fig. 6, improvements in the solutions and reduction in the relative errors resulted as the qubit number increased from 12 to 20. The relative error of the 7×7 problem with ω_j encoded in 16 qubits is about 0.88%, which is about 9 times larger than that of the 3×3 problem. This is understood by the fact that the error due to the truncated eigenvalues λ_j used in phase estimation plays a major role here. This is because a larger problem requires a larger number of controlled-U operations (see Fig. 2), resulting in more error accumulation. The overall accuracy of the results is relatively similar when using 16 and 20 qubits; thus, we chose to fix ω_j at 16 qubits as we investigated further improvements to the solutions.

To further reduce the error discussed in the previous paragraph, we used eigenvalue amplification (as discussed in Sect. 4.1) with a factor of 2^f where f takes the value 0, 4, and 8. A larger f includes more number of bits in the fractional part of the eigenvalue and thus retains more accuracy in the solution. The effects of eigenvalue amplification on the 7×7 problem are shown in Fig. 7, which confirms the significant reduction to the

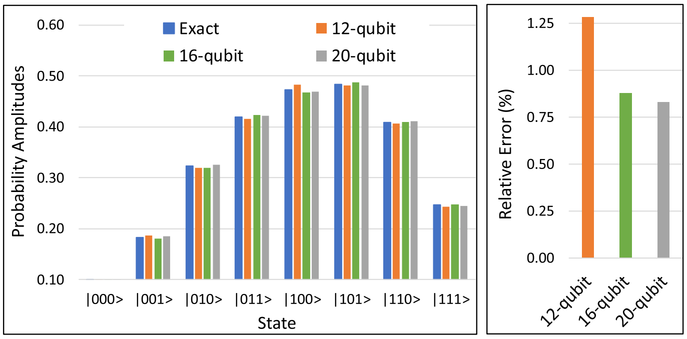

Fig. 6 Effects of increasing the number of qubits on the rotation angular coefficients ω_j on a 7×7 problem size. (Left) Exact solution with the MPS result simulated by encoding ω_j on different numbers of qubits of reg. A. (Right) Relative errors of the left panel results with respect to the exact solution

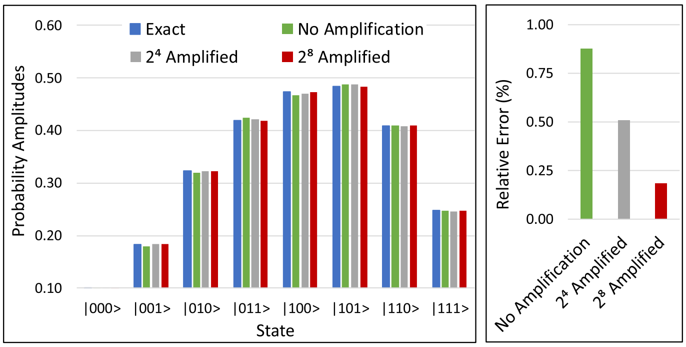

Fig. 7 Effects of eigenvalue amplification on a 7×7 problem size. (Left) Comparing solutions with varying levels of eigenvalue amplification while using 16 qubits for ω_j. (Right) Drastic reduction in relative errors of the solutions on the left panel with respect to the exact result

relative error when we use eigenvalue amplification. At 2^8 amplification, the relative error is 0.18%, a 5-fold improvement in accuracy compared to using no amplification. We are confident that a higher amplification factor ($f > 8$) would further reduce this error.

To confirm the robustness of our algorithm and its accuracy in solving the Poisson equation for practical problem sizes, we present the solution for a 15×15 problem, including its exact result, in Fig. 8. For an arbitrarily chosen input state $|b\rangle$ (see Table 1 for its expression), the overall solution is encouraging. The relative error with respect to the exact result again quickly goes down as we apply eigenvalue amplification and increase its amplification factor.

Success probability Analytically, the success probability (SP) of the measurement is determined by the eigenvalue distribution and their levels of accuracy. As can be seen in the state before the measurement, i.e., $|0\rangle \otimes \sum_{j=1}^{2^n-1} \beta_j |u_j\rangle \left(\sqrt{1 - C^2/\lambda_j^2}|0\rangle + C/\lambda_j|1\rangle\right)$ (C being a normalizing constant) [12], the SP is determined by the summation of the squares of reciprocals of eigenvalues. So the values of SP using the truncated (i.e.,

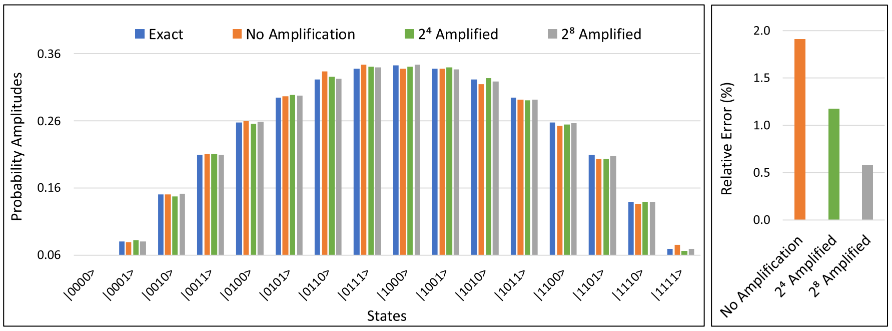

Fig. 8 Solution of a 15×15 problem size. (Left) Comparing the solutions with different levels of eigenvalue amplification (with a fixed ω_j of 16 qubits) and exact result. (Right) Significant reduction in the relative errors with respect to the exact result

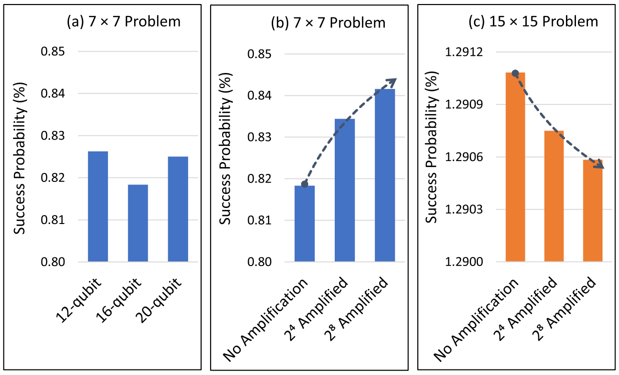

Fig. 9 Success probability of obtaining the desired state on 7×7 and 15×15 problem sizes. Here (**a**), (**b**), and (**c**) correspond to the cases presented in Figs. 6, 7, and 8, respectively. The arrow line in (b) and (c) indicates the steady improvements in success probability toward their respective analytical values with the increase in eigenvalue amplification

integer) eigenvalues of 3×3 and 7×7 problems are 1.367% and 1.337%, respectively (SP is greater than 1 as $C = 1$ is considered here). However, on the simulation side, we compute the SP by dividing the number of trials with correct output by the total number of repeated trials and then multiplying the factor by 100 [41]. When no eigenvalue amplification is used, compared to the analytical SP of 1.367% on a 3×3 problem, we computed an SP of 1.103%, which is very close to the number 1.120% reported by Wang et al. [23]. On the 7×7 problem, as shown in Fig. 9a, the SP appears to vary between 0.818% and 0.826%, but without showing any steady movement toward its analytical value 1.337% as more accurate $|\omega_j\rangle$'s were used by increasing the number of qubits in reg. A. This suggests that the SP is more sensitive to the other dominant source of error that involves the truncation of eigenvalues used in phase estimation. Therefore, controlling such error requires using eigenvalue amplification. Figure 9b, and c shows the SP on 7×7 and 15×15 problems plotted with different levels of amplification. As expected, both figures confirm the steady improvements in the SP rightly proceeding to their analytical values 1.154% and 1.122% (those calculated using the exact eigenvalues), respectively, with higher levels of amplification. Though we have no doubt that a higher amplification factor ($f > 8$) would further improve the SP approaching it to its respective analytical value, we are unable to fully characterize the reason for two different variation trends of SP shown by the dotted arrow in Fig. 9b and c. In both cases, though we have taken a large number of trials (1.2 million for each), theoretically, a 15×15 problem requires a polynomially greater number of trials than that of a 7×7 problem as required by the relation of κ with N (the size of the discretized matrix A). In practice, one could still add more trials to the 15×15 case, but doing that only might not fix the problem. This is because, theoretically, both the amplification factor f and the number of trials need to be infinitely large to ensure an accurate and reproducible solution. To resolve this two-factor constraint, in the scaling section, we discuss a plan to extend our algorithm by combining it with an iterative method [21]. This will further optimize the number of qubits required for eigenvalue amplification with a higher f, resulting in achieving a converged solution using a relatively lower number of trials.

Also, as κ grows, matrix A becomes more and more difficult to invert, and the solutions become less stable [12]. Furthermore, the basic error of the solutions caused by the central difference approximation is related to the condition number as $\kappa = O(\epsilon^{-2\alpha})$ (ϵ being error and α being a smoothness parameter), and therefore, an additive preconditioner [42] may be used to reduce κ.

Summarizing the input and output In Table 1, we present all the problems we discussed so far, along with each input state $|b\rangle$ and Poisson solution $|v\rangle$. Note that the relative errors shown in Table 1 gradually increase with the problem size. This may be explained by the fact that even a small inaccuracy in the encoded eigenvalues would cause the accumulation of a larger amount of error due to the extra controlled-U operations required for larger problems. Therefore, an optimum solution of a larger problem would require using an even higher amplification factor f, which ensures more number of bits (0 and 1) of the fractional part of the eigenvalues to be taken into account for this computation. In the current implementation, though it is possible to pre-set the value of f, one may not want to do that. Instead, it is desirable to increase f until the convergence of the solution with the desired accuracy is achieved. Also, for all of our simulations, even though we use angular coefficients encoded to a fixed number of qubits (16), encoding them to a higher number of qubits would certainly improve the accuracy of the solution.

Algorithm scaling and further improvement direction Compared to Cao's algorithm [22], Wang's method reduces the cost of the problem by one order, from $O(m^4)$ to $O(m^3)$, by performing the controlled rotation of HHL using the arc cotangent function, meaning the rotation angles are prepared directly from the eigenvalues instead of their reciprocals. While the theoretical complexity of the algorithm is discussed in Ref. [23]; in this work, we focus on both computational and experimental feasibility.

On the computational aspect, we not only ensure better accuracy of solutions that are lacking in the existing approaches [12, 22, 23] but also successfully demonstrate the scaling of the problem to larger matrices. Within our implementation, our code-base dynamically generates optimized circuits for any given size of the problem. The practicality of the algorithm is demonstrated through the scaling performance of problems for up to 31×31 (see Fig. 10). During runtime, we recorded the total number of qubits used and the circuit depth on the basis of elementary gates after the circuit decomposition. As shown in Fig. 10, though the circuit depth grows exponentially as the problem size increases, the number of qubits scales linearly, which is encouraging. This is because, for an existing simulator or quantum hardware, relatively the total number of qubits is more critical factor than the circuit depth.

However, one may also point out that the exponential increase in the circuit depth may require a longer coherent time, which indeed is still a challenge to increase from the technological development point of view. Also, the exponential growth of the circuit depth may have a similar effect on the time complexity of the algorithm. The circuit depth issue and the overall scaling can be further optimized by: (1) Optimally mapping the logical to physical qubits when compiling quantum circuits onto hardware with restricted connectivity by trading off circuit depth and gate count [43]. We have already implemented this and discuss more about it later in the experimental section; (2) Combining our algorithm structure with an iterative method [21] would further optimize qubit usage, especially for the eigenvalue expression, while improving the computational speedup by requiring fewer repeated measurements. In fact, our algorithm is well suited for coupling with an iterative solution process, which would ensure even higher accuracy in results; and (3) Adapting a circuit knitting technique [44–47], which allows partitioning of large quantum circuits into subcircuits that fit on smaller devices and then knitting the results back together using a classical computer. Although there is some overhead associated with the knitting process, it would open a path to explore massive problems, including multidimensional ones. In our current implementation, due to the full circuit being processed in a single quantum processor, the section of the workflow is relatively slow, especially on large problems. Circuit knitting would require locating processing bottlenecks through profiling and accordingly distributing the tasks on multiple quantum processing units (QPUs), ensuring the tasks' parallelism with load balancing, which would result in the speeding up of the whole computation. In fact, this is the path IBM takes in realizing their near-term hardware development by combining multiple QPUs [48–50] through circuit knitting techniques.

If we extend this to d dimensions, the main difference would be the Hamiltonian simulation for $e^{iA^{(d)}t}$, which can be parallelized across d (see Eq. 5). Therefore, for the multidimensional case, the complexity of our algorithm still grows linearly. This is encouraging as the cost of any classical algorithm solving the d-dimensional Poisson equation grows exponentially with d. The linear cost of the quantum algorithm makes it ideal for experiments solving the d-dimensional Poisson equation on near-term quantum hardware and achieving exponential speedup in terms of d.

6 Circuit demonstration on quantum hardware

Qiskit allows for easy circuit optimization and the running of circuits on IBM's quantum hardware [29]. Their *ibmq_manila* and *ibmq_brooklyn* systems containing 5 and 65 qubits, respectively, [30] are used to run our circuit experiments. These systems support only the *CNOT*, I, R_z, $\sqrt{X}$, and X gates, so any other gates used must be compiled down to these basic components, for example, the MCMT operation is compiled

Table 1 One-dimensional Poisson equation, i.e., $A|v\rangle = |b\rangle$'s inputs and solution states, and relative errors in the solutions for different sizes of problems

Size of A	Input States, $\|b\rangle$	MPS-Simulated Poisson Solution, $\|v\rangle$	Relative Error (%)
3×3	$\frac{1}{\sqrt{2}}\|01\rangle + \frac{1}{2}(\|10\rangle + \|11\rangle)$	$0.553\|01\rangle + 0.673\|10\rangle + 0.490\|11\rangle$	0.0899
7×7	$\frac{1}{4}(\|001\rangle + \|010\rangle + \|011\rangle + \|100\rangle) + \frac{1}{2}(\|101\rangle + \|110\rangle + \|111\rangle)$	$0.184\|001\rangle + 0.323\|010\rangle + 0.418\|011\rangle + 0.473\|100\rangle + 0.484\|101\rangle + 0.409\|110\rangle + 0.248\|111\rangle$	0.1839
15×15	$\frac{1}{4}(\|0001\rangle + \|0010\rangle + \|0011\rangle + \|0100\rangle + \|0101\rangle + \|0110\rangle + \|0111\rangle + \|1000\rangle + \|1001\rangle + \|1010\rangle + \|1011\rangle + \|1100\rangle) + \frac{1}{2}\|1101\rangle + 0.000\|1110\rangle + 0.000\|1111\rangle$	$0.080\|0001\rangle + 0.152\|0010\rangle + 0.209\|0011\rangle + 0.258\|0100\rangle + 0.297\|0101\rangle + 0.322\|0110\rangle + 0.340\|0111\rangle + 0.344\|1000\rangle + 0.337\|1001\rangle + 0.319\|1010\rangle + 0.292\|1011\rangle + 0.257\|1100\rangle + 0.208\|1101\rangle + 0.139\|1110\rangle + 0.070\|1111\rangle$	0.5825

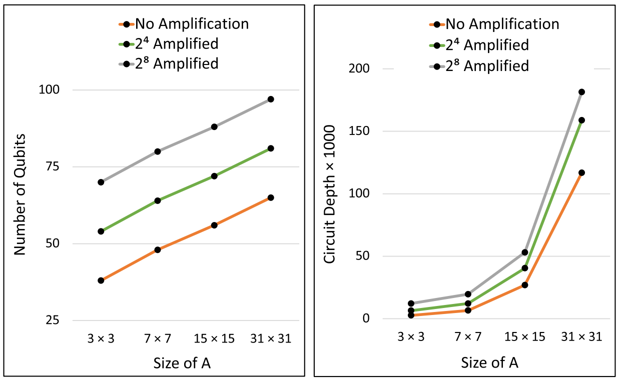

Fig. 10 An optimum number of qubits is used for constructing the circuit representing the Poisson solver algorithm. (Left) How the resources, that is, the number of qubits, scales with the size of the problem. The scaling is shown with varying levels of amplification. (Right) The circuit depth with the size of the problem

to *CNOT* gates. The circuit transformation is performed using Qiskit's transpiler [51] with samplings that ensure a minimum depth of the optimized circuit.

One crucial part of experimenting with circuits on physical hardware is finding the optimal mapping of virtual qubits to physical qubits on the hardware [43, 52–55]. Qiskit does this automatically via stochastic mappings of virtual to physical qubits and offers different levels of transpilation for circuit optimization. We experimented with multiple levels of optimization, conducting stochastic searches of mappings in an effort to find an optimal mapping for our circuit. Our final circuit was scholastically sampled over 1500 times to find such an optimal mapping. However, as we will discuss, the accuracy of our experiments was ultimately limited by the accumulated error of the large number of *CNOT* gates required in our circuit.

6.1 Measurement error mitigation on $|b\rangle$

The current state of quantum hardware presents many challenges, particularly the short coherence time and accumulation of noise in experiments [56]. In addition, on physical devices like IBM's *ibmq_manila* or *ibmq_brooklyn*, different pairs of qubits have different *CNOT* error rates, which also affects the ultimate accuracy of the system as many qubits are directly entangled with other qubits using *CNOT* gates in the course of an experiment [30]. Therefore, it makes sense to first set up a small system with a limited number of *CNOT* gates and to experiment on that.

Additionally, there are two more purposes for this experiment: (1) Setting up a test model with the exact input state used in the full circuit for the 3×3 problem (corresponding to Figs. 4 and 5), from which we get an estimated error related to the measurement part of the algorithm, and (2) Determining how much of that measurement error may be mitigated through the existing model and how the error associated with the relatively small number of CNOT gates affects the overall result.

Based on the available options for experimentation, we first investigate errors using a simple noise model generated from the properties of real device *ibmq_manila* from the IBM Quantum [30] and mitigate those errors on the measurement qubits [57–60]. To estimate the amount of error in our actual circuit, it was enough to use a test circuit involving only the input/output state $|b\rangle$ (those acting on reg. B) where we do the measurement. A diagram of the circuit is shown in the top panel of Fig. 11. Following the circuit transformation through the transpiler for *ibmq_manila*, the circuit decomposes to a number of basis gates that includes 2 *CNOT* gates and a few single qubit gates. We experiment with the circuit on *ibmq_manila* with and without mitigating errors on both measurement qubits and compare those results with the MPS-simulated result. The results with two optimization levels 0 and 3 are shown in the bottom-left panel of Fig. 11. While there are some noticeable differences in probability for some states, the overall result appears to improve with the error mitigation and for higher levels of transpiler optimization. This is clearly evident in the bottom-right panel of the figure, where it shows the relative error with respect to the simulation. It confirms that to reduce the error in the experiment significantly, it is not enough just to tune the optimization levels; error mitigating is also essential on the NISQ hardware.

We want to mention that while this experiment does not completely represent the full circuit of the Poisson equation solver, we believe that it offers us a projection as to what one could expect if NISQ or near-term hardware could support the experiment of the full circuit. Our experimental results of this test system project a significant reduction in the relative error in the measurement part of the circuit, hence indicating the possibility of mitigating a similar magnitude of error on the full system.

6.2 Effects of CNOT error on the experiment of 3 $\times$ 3 problem on quantum hardware

One measure of the fidelity of quantum systems is in terms of their *CNOT* error rates, that is, the accuracy of individually entangled bits when performing a two-qubit *CNOT* gate [61, 62]. The average *CNOT* error on the *ibmq_brooklyn* system is 8.094e-2; in other words, they have an accuracy of about 0.92. Thus, we can estimate the overall accuracy of the experiment based on the final number of *CNOT* gates transpiled from the more abstract circuit, approximated by 0.92^c where c is the final number of *CNOT* gates after transpilation. Ultimately, every Toffoli gate, as well as more complex gates such as MCMT, is transpiled into many *CNOT*s. After a series of transformations using different levels of transpiler optimization, our best circuit for the 3×3 problem required roughly 5.5k *CNOT* gates. Unfortunately, this number is quite large given the experimental fidelity of current NISQ devices. As a result, the accumulated errors of the *CNOT*-gates result in the washing out of the experimental accuracy, which contributes to the artifact of a nonzero contribution for the $|00\rangle$ state (see the figure in Ref. [1]). In general, experimenting with the circuit on different IBM hardware would end up with similar results, as the best *CNOT* accuracy on any system is less than 0.99. Therefore, the *CNOT* error rate appears to be the most dominant bottleneck in realizing the algorithm on NISQ hardware. This experiment helped us pinpoint this key limiting factor of the NISQ device. In spite of the instrumental difficulties involving the *CNOT* errors, for the first time, we showed that such a full circuit can easily be mapped (logical to physical gates) and experimented on the existing quantum hardware.

7 Conclusions

We have successfully demonstrated several crucial improvements and optimizations essential for scaling the Poisson Solver to larger problem sizes within a hybrid algorithm. By identifying two major sources of error accumulation in the algorithm, one in the phase estimation involving truncating eigenvalues and the other related to the accuracy of the rotation angular coefficients, we were able to build a circuit implementation that was dynamically tunable with respect to those two sources of inaccuracy. Adding accuracy to the eigenvalues through eigenvalue amplification yielded the best improvements and proved to be necessary when expanding to larger, unsolved problem sizes. Not only did we perform more accurate computations with these amplified eigenvalues, but we also were able to achieve a higher success probability on every sin-

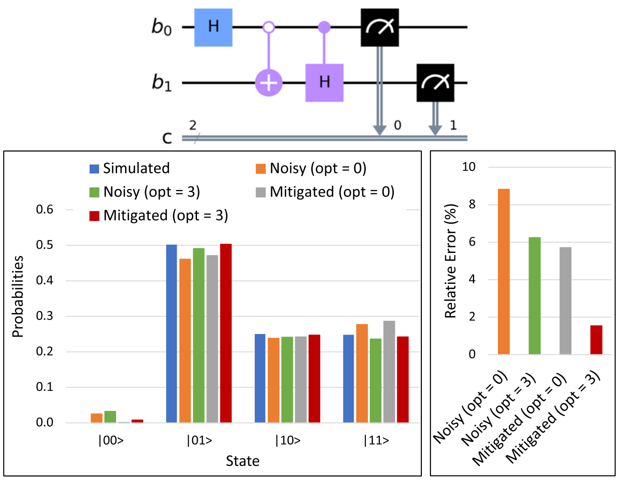

Fig. 11 Measurement error mitigation of the simplified circuit built out of 3 × 3 problem. (Top) Quantum circuit built using the input state $|b\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{2}(|10\rangle + |11\rangle)$ of the 3 × 3 problem. (Bottom left) The MPS-simulated result is compared with the experimental result from the noisy IBM's *ibmq_manila* device and after mitigating the error on the measurement qubits. Results are shown with the optimization levels 0 and 3 of transpiler. (Bottom right) The relative error in the experimental results with respect to MPS-simulated result

gle circuit than previously possible with truncated eigenvalues. We presented results on significantly larger problem sizes than previous works, as well as improved accuracy on existing problem sizes. These accuracy improvements also translated to the larger problems we demonstrated.

Clearly, our algorithm represents an advancement in accuracy and usability and more closely represents what will be put into real-world applications of this theory in the near future. Scalability is a critical step toward breaking the curse of dimensionality that currently plagues solving the Poisson equation, and our multilevel optimized circuit alleviates many of the pressures holding this technology back by dynamically controlling the problem size and register size of crucial segments of the algorithm.

While we were successful in demonstrating our advancements to the Quantum Poisson Solver on a simulator, current quantum hardware proved to be too error-prone to provide accurate results [56, 63, 64]. In spite of that, we were able to demonstrate the improvements in the experimental result on IBM's *ibmq_manila* device by mitigating error on the measurement qubits on a simplified circuit built out of an exact input/output state of a 3 × 3 problem and including a small number of *CNOT* gates. Ultimately, the accumulated error of the large number of *CNOT* gates required in the full circuit for the Poisson solver, in conjunction with the number of qubits necessary for larger problems, was the limiting factor in our exploration. However, this work has laid the foundation for advanced algorithms that will become usable in the near future as hardware improvements continue. As we see the arrival of more accurate systems with lower *CNOT* error rates, our algorithm will become usable in larger and more practical problems.

We have also discussed a vision of how the problem size can be further extended while managing the circuit width and depth at a level suitable to the current technology. In this regard, we have prescribed multilevel solutions, including combining an iterative framework as proposed by Saito et al. [21], in order to ensure even higher accuracy in results with fewer repeated shots while requiring an optimum number of qubits. Encouraged by the industry's near-term hardware development roadmap (such as IBM's upcoming quantum-centric supercomputing hardware [48], for example), we proposed partitioning large circuits through circuit knitting techniques and then running the subcircuits on multiple QPUs in parallel. This would allow us to explore significantly larger problems, including multidimensional ones, with greater computational speed-up.

Appendix A: Reviewing Cao et al. [22] and Wang et al. [23] works

In the context of the quantum circuit model, Cao et al. [22] first used the original HHL algorithm [12] to solve the Poisson equation. Later, our co-author Wang et al. [23] pointed out a bottleneck of Cao's algorithm where the controlled rotation is implemented by the arc sine function evaluation. In other words, the bottleneck comes from the process of performing a linear mapping from state $|\lambda_j\rangle$ to $\lambda_j^{-1}|\lambda_j\rangle$, where λ_j represents the eigenvalues of a matrix of the linear system of equations. More precisely, after having the eigenvalue state $|\lambda_j\rangle$ by phase estimation, Cao et al. evaluate the reciprocal state $|1/\lambda_j\rangle$ through the Newton iteration method, after that the binary state of $|1/\lambda_j\rangle$ is converted to the probability amplitude $1/\lambda_j$ through the controlled R_y rotations with the angle of $\theta = \arcsin(1/\lambda_j)$, where the arc sine function is evaluated by the cut-and-try method. Since the cost of calculating the sine function is $O(m^3)$ where m is the number of qubits of input register, then the evaluation cost of the arc sine function is $O(m^4)$ [23].

Wang's approach resolved the bottleneck of Cao's algorithm and developed a quantum fast Poisson solver with complete and modular circuit representation. First, they proposed a new way of implementing the controlled rotation in the HHL algorithm. That is, they introduced a method in which they take the state $|\lambda_j\rangle$ to $\lambda_j^{-1}|\lambda_j\rangle$ directly without passing through the $|1/\lambda_j\rangle$ state. In this process, they adopted a novel method called qFBE (quantum function-value binary expansion) to evaluate the arc cotangent function [65, 66]. With this method, they reduced the cost of the problem from $O(m^4)$ to $O(m^3)$. Second, they developed the inverse qFBE method to compute the cosine function in order to simplify the Hamiltonian simulation subroutine of HHL, making the circuit design easier and more modular. Finally, they also exploited quantum algorithms for solving the reciprocal and square root operations using the classical non-restoring method [67]. By developing a new way of implementing the controlled rotation within HHL and quantum circuits for solving the Poisson equation, they not only reduced the algorithm's complexity but also made the circuit complete and implementable. However, in reducing the cost and complexity of the quantum circuit, Wang et al. truncated both the eigenvalues of the matrix and the rotation angular coefficients. As a result, numerical errors are accumulated, and eventually that compromises the accuracy of the solution of the Poisson equation.

Acknowledgements This research was supported in part by the Notre Dame Center for Research Computing through Notre Dame Research. The authors admire Scott Hampton's comments on the manuscript. The authors also appreciate access to IBM Quantum Hardware through IBM Quantum Network. This work was supported in part by the Department of Energy, Office of Science, Advanced Scientific Computing Research program. This research used resources from the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the US Department of Energy under Contract No. DE-AC05-00OR22725. *Notice*: This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the US Department of Energy. The publisher, by accepting the article for publication, acknowledges that the US Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of the manuscript, or allow others to do so, for US Government purposes. The DOE will provide public access to these results in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Data Availability All data generated or analyzed during this study are included in this published article.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Robson, W., Saha, K.K., Howington, C., Suh, I.-S., Nabrzyski, J.: Advanced quantum Poisson solver in the NISQ era. In: 2022 IEEE International Conference on Quantum Computing and Engineering (QCE) , 741 (2022), https://doi.org/10.1109/QCE53715.2022.00103
2. Poisson equation, numerical methods; encyclopedia of mathematics.,https://encyclopediaofmath.org/index.php?title=Poisson_equation,_numerical_methods&oldid=48217
3. Feynman, R.P.: Simulating physics with computers. Int. J. Theor. Phys. **21**, 467 (1982). https://doi.org/10.1007/BF02650179
4. Leyton, S.K., Osborne, T.J.: A quantum algorithm to solve nonlinear differential equations (2008) arXiv:0812.4423
5. Berry, D.W.: High-order quantum algorithm for solving linear differential equations. J. Phys. A: Math. Theor. **47**, 105301 (2014). https://doi.org/10.1088/1751-8113/47/10/105301
6. Berry, D.W., Childs, A.M., Ostrander, A., Wang, G.: Quantum algorithm for linear differential equations with exponentially improved dependence on precision. Commun. Math. Phys. **356**, 1057 (2017). https://doi.org/10.1007/s00220-017-3002-y
7. Childs, A.M., Liu, J.-P.: Quantum spectral methods for differential equations. Commun. Math. Phys. **375**, 1427 (2020). https://doi.org/10.1007/s00220-020-03699-z
8. Childs, A.M., Liu, J.-P., Ostrander, A.: High-precision quantum algorithms for partial differential equations. Quantum **5**, 574 (2021). https://doi.org/10.22331/q-2021-11-10-574
9. Costa, P.C.S., Jordan, S., Ostrander, A.: Quantum algorithm for simulating the wave equation. Phys. Rev. A **99**, 012323 (2019). https://doi.org/10.1103/PhysRevA.99.012323
10. Arrazola, J.M., Kalajdzievski, T., Weedbrook, C., Lloyd, S.: Quantum algorithm for nonhomogeneous linear partial differential equations. Phys. Rev. A **100**, 032306 (2019). https://doi.org/10.1103/PhysRevA.100.032306
11. Dervovic, D., Herbster, M., Mountney, P., Severini, S., Usher,N., Wossnig, L.: Quantum linear systems algorithms: a primer (2018) arXiv:1802.08227

12. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Phys. Rev. Lett. **103**, 150502 (2009). https://doi.org/10.1103/PhysRevLett.103.150502
13. Cao, Y., Daskin, A., Frankel, S., Kais, S.: Quantum circuit design for solving linear systems of equations. Mol. Phys. **110**, 1675 (2012). https://doi.org/10.1080/00268976.2012.668289
14. Childs, A.M., Kothari, R., Somma, R.D.: Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. SIAM J. Comput. **46**, 920 (2017). https://doi.org/10.1137/16M1087072
15. Berry, D.W., Childs, A.M., Kothari, R.: Hamiltonian simulation with nearly optimal dependence on all parameters. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (2015) pp. 792–809, https://doi.org/10.1109/FOCS.2015.54
16. Kalajdzievski, T., Arrazola, J.M.: Exact gate decompositions for photonic quantum computing. Phys. Rev. A **99**, 022341 (2019). https://doi.org/10.1103/PhysRevA.99.022341
17. Huang, H.-Y., Bharti, K., Rebentrost, P.: Near-term quantum algorithms for linear systems of equations with regression loss functions. New J. Phys. **23**, 113021 (2021). https://doi.org/10.1088/1367-2630/ac325f
18. Subaşı, Y., Somma, R.D., Orsucci, D.: Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing. Phys. Rev. Lett. **122**, 060504 (2019). https://doi.org/10.1103/PhysRevLett.122.060504
19. Liu, H.-L., Wu, Y.-S., Wan, L.-C., Pan, S.-J., Qin, S.-J., Gao, F., Wen, Q.-Y.: Variational quantum algorithm for the poisson equation. Phys. Rev. A **104**, 022418 (2021). https://doi.org/10.1103/PhysRevA.104.022418
20. Sato, Y., Kondo, R., Koide, S., Takamatsu, H., Imoto, N.: Variational quantum algorithm based on the minimum potential energy for solving the Poisson equation. Phys. Rev. A **104**, 052409 (2021). https://doi.org/10.1103/PhysRevA.104.052409
21. Saito, Y., Lee, X., Cai, D., Asai, N.: An iterative improvement method for HHL algorithm for solving linear system of equations (2021) arXiv:2108.07744
22. Cao, Y., Papageorgiou, A., Petras, I., Traub, J., Kais, S.: Quantum algorithm and circuit design solving the Poisson equation. New J. Phys. **15**, 013021 (2013). https://doi.org/10.1088/1367-2630/15/1/013021
23. Wang, S., Wang, Z., Li, W., Fan, L., Wei, Z., Gu, Y.: Quantum fast Poisson solver: the algorithm and complete and modular circuit design. Quantum Inf. Process. **19**, 170 (2020). https://doi.org/10.1007/s11128-020-02669-7
24. McClean, J.R., Romero, J., Babbush, R., Aspuru-Guzik, A.: The theory of variational hybrid quantum-classical algorithms. New J. Phys. **18**, 023023 (2016)
25. Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S.C., Endo, S., Fujii, K., McClean, J.R., Mitarai, K., Yuan, X., Cincio, L., Coles, P.J.: Variational quantum algorithms (2021) arXiv:2012.09265
26. Zhou, L., Wang, S.-T., Choi, S., Pichler, H., Lukin, M.D.: Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices. Phys. Rev. X **10**, 021067 (2020). https://doi.org/10.1103/PhysRevX.10.021067
27. Barron, G.S., Wood, C.J.: Measurement error mitigation for variational quantum algorithms (2020) arXiv:2010.08520
28. Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018). https://doi.org/10.22331/q-2018-08-06-79
29. Anis, M.S. et al.: Qiskit: An open-source framework for quantum computing (2021), https://qiskit.org
30. IBM Quantum, https://quantum-computing.ibm.com, 2021
31. QRUNES, https://github.com/OriginQ/QRunes
32. Demmel, J.W.: Applied Numerical Linear Algebra. Society for Industrial and Applied Mathematics, Philadelphia (1997). https://doi.org/10.1137/1.9781611971446
33. Shewchuk, J.R.: An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Tech. Rep. (USA, 1994), https://dl.acm.org/doi/10.5555/865018
34. Aaronson, S.: New quantum algorithms promise an exponential speed-up for machine learning, clustering and finding patterns in big data, but to achieve a real speed-up, we need to delve into the details. Nat. Phys. **11**, 291 (2015). https://doi.org/10.1038/nphys3272
35. Luis, A., Peřina, J.: Optimum phase-shift estimation and the quantum description of the phase difference. Phys. Rev. A **54**, 4564 (1996). https://doi.org/10.1103/PhysRevA.54.4564
36. Lloyd, L.: Universal quantum simulators. Science **273**, 1073 (1996)

37. Berry, D.W., Ahokas, G., Cleve, R., Sanders, B.C.: Efficient quantum algorithms for simulating sparse hamiltonians. Commun. Math. Phys. **270**, 359 (2007). https://doi.org/10.1007/s00220-006-0150-x
38. Cleve, R., Ekert, A., Macchiavello, C., Mosca, M.: Quantum algorithms revisited. Proc. R. Soc. Lond. A **454**, 339 (1998). https://doi.org/10.1098/rspa.1998.0164
39. Wang, S., Wang, Z., Li, W., Fan, L., Cui, G., Wei, Z., Gu, Y.: A quantum Poisson solver implementable on NISQ devices (2020) arXiv:2005.00256
40. How to Measure Errors, https://netlib.org/lapack/lug/node75.html
41. Qi, F., Smith, K.N., LeCompte, T., Tzeng, N., Yuan, X., Chong, F.T., Peng, L.: Quantum vulnerability analysis to accurate estimate the quantum algorithm success rate (2022) arXiv:2207.14446
42. Pana, V.Y., Ivolgin, D., Murphy, B., Rosholt, R.E., Tang, Y., Yan, X.: Additive preconditioning for matrix computations. Linear Algebra Appl. **432**, 1070 (2010). https://doi.org/10.1016/j.laa.2009.10.020
43. Li, G., Ding, Y., Xie, Y.: Tackling the qubit mapping problem for NISQ-era quantum devices (2019) arXiv:1809.02573
44. Eddins, A., Motta, M., Gujarati, T.P., Bravyi, S., Mezzacapo, A., Hadfield, C., Sheldon, S.: Doubling the size of quantum simulators by entanglement forging. PRX Quantum **3**, 010309 (2022). https://doi.org/10.1103/PRXQuantum.3.010309
45. Bravyi, S., Smith, G., Smolin, J.A.: Trading classical and quantum computational resources. Phys. Rev. X **6**, 021043 (2016). https://doi.org/10.1103/PhysRevX.6.021043
46. Peng, T., Harrow, A.W., Ozols, M., Wu, X.: Simulating large quantum circuits on a small quantum computer. Phys. Rev. Lett. **125**, 150504 (2020). https://doi.org/10.1103/PhysRevLett.125.150504
47. Tang, A.W., Tomesh, T., Suchara, M., Larson, J., Martonosi, M.: Cutqc: using small quantum computers for large quantum circuit evaluations. In: Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems , 473 (2021), https://doi.org/10.1145/3445814.3446758
48. Bravyi, S., Dial, O., Gambetta, J. M., Gil, D., Nazario, Z.: The future of quantum computing with superconducting qubits (2022) arXiv:2209.06841
49. Tham, E., Khait, I., Brodutch, A.: Quantum circuit optimization for multiple QPUS using local structure (2022) arXiv:2206.09938
50. Piveteau, C., Sutter, D.: Circuit knitting with classical communication (2023) arXiv:2205.00016
51. Younis, E., Iancu, C.: Quantum circuit optimization and transpilation via parameterized circuit instantiation (2022) arXiv:2206.07885
52. Zulehner, A., Paler, A., Wille, R.: An efficient methodology for mapping quantum circuits to the IBM QX architectures. IEEE Trans. Comput. Aid. Des. Integr. Circuits Syst. **38**, 1226 (2019). https://doi.org/10.1109/TCAD.2018.2846658
53. Wille, R., Burgholzer, L., Zulehner, A.: Mapping quantum circuits to ibm qx architectures using the minimal number of swap and h operations (2019) arXiv:1907.02026
54. Murali, P., Baker, J.M., Abhari, A.J., Chong, F.T., Martonosi, M.: Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers (2019) arXiv:1901.11054
55. Burgholzer, L., Schneider, S., Wille, R.: Limiting the search space in optimal quantum circuit mapping (2022) arXiv:2112.00045
56. Corcoles, A.D., Kandala, A., Javadi-Abhari, A., McClure, D.T., Cross, A.W., Temme, K., Nation, P.D., Steffen, M., Gambetta, J.M.: Challenges and opportunities of near-term quantum computing systems (2019) arXiv:1910.02894
57. Ferris, K.J., Rasmusson, A.J., Bronn, N.T., Lanes, O.: Quantum simulation on noisy superconducting quantum computers (2022) arXiv:2209.02795
58. Funcke, L., Hartung, T., Jansen, K., Kühn, S., Stornati, P., Wang, X.: Measurement error mitigation in quantum computers through classical bit-flip correction. Phys. Rev. A **105**, 062404 (2022). https://doi.org/10.1103/PhysRevA.105.062404
59. Alexandrou, C., Funcke, L., Hartung, T., Jansen, K., Kühn, S., Polykratis, G., Stornati, P., Wang, X., Weber, T.: Investigating the variance increase of readout error mitigation through classical bit-flip correction on ibm and rigetti quantum computers (2021) arXiv:2111.05026
60. Acampora, G., Grossi, M., Vitiello, A.: Genetic algorithms for error mitigation in quantum measurement, in 2021 IEEE Congress on Evolutionary Computation (CEC) (2021) pp. 1826–1832, https://doi.org/10.1109/CEC45853.2021.9504796
61. Calderon-Vargas, F.A., Kestner, J.P.: Dynamically correcting a CNOT gate for any systematic logical error. Phys. Rev. Lett. **118**, 150502 (2017). https://doi.org/10.1103/PhysRevLett.118.150502

62. Chow, J.M., Gambetta, J.M., Córcoles, A.D., Merkel, S.T., Smolin, J.A., Rigetti, C., Poletto, S., Keefe, G.A., Rothwell, M.B., Rozen, J.R., Ketchen, M.B., Steffen, M.: Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits. Phys. Rev. Lett. **109**, 060501 (2012). https://doi.org/10.1103/PhysRevLett.109.060501
63. Johnstun, S., Huele, J.-F.V.: Understanding and compensating for noise on IBM quantum computers. Am. J. Phys. **89**, 935 (2021). https://doi.org/10.1119/10.0006204
64. Baum, Y., Amico, M., Howell, S., Hush, M., Liuzzi, M., Mundada, P., Merkh, T., Carvalho, A.R., Biercuk, M.J.: Experimental deep reinforcement learning for error-robust gate-set design on a superconducting quantum computer. PRX Quantum **2**, 040324 (2021). https://doi.org/10.1103/PRXQuantum.2.040324
65. Borwein, J.M., Girgensohn, R.: Addition theorems and binary expansions. Can J. Math. **47**, 262 (1995). https://doi.org/10.4153/CJM-1995-013-4
66. Wang, S., Wang, Z., Li, W., Fan, L., Cui, G., Wei, Z., Gu, Y.: Quantum circuits design for evaluating transcendental functions based on a function-value binary expansion method. Quantum Inf. Process. **19**, 347 (2020). https://doi.org/10.1007/s11128-020-02855-7
67. Sutikno, T.: An efficient implementation of the non restoring square root algorithm in gate level. Int. J. Comput. Theory Eng. **3**, 46 (2011). https://doi.org/10.7763/IJCTE.2011.V3.281