Quantum multirow iteration algorithm for linear systems with nonsquare coefficient matrices

Weitao Lin,[*] Guojing Tian,[†] and Xiaoming Sun[‡]
State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China and University of Chinese Academy of Sciences, Beijing 100049, China

(Received 3 January 2024; revised 13 June 2024; accepted 8 August 2024; published 27 August 2024)

In the field of quantum linear system algorithms, quantum computing has realized exponential computational advantages over classical computing. However, the focus has been on square coefficient matrices, with few quantum algorithms addressing nonsquare matrices. Towards this kind of problems defined by $Ax = b$ where $A \in \mathbb{R}^{m\times n}$, we propose a quantum algorithm inspired by the classical multirow iteration method and provide an explicit quantum circuit based on the quantum comparator and quantum random access memory. The time complexity of our quantum multirow iteration algorithm is $O(K \log_2 m)$, with K representing the number of iteration steps, which demonstrates an exponential speedup compared to the classical version.. Based on the convergence of the classical multirow iteration algorithm, we prove that our quantum algorithm converges faster than the quantum one-row iteration algorithm presented by Shao and Xiang [C. Shao and H. Xiang, Phys. Rev. A **101**, 022322 (2020)]. Moreover, our algorithm places less demand on the coefficient matrix, making it suitable for solving inconsistent systems and quadratic optimization problems.

DOI: 10.1103/PhysRevA.110.022438

I. INTRODUCTION

Quantum linear system algorithms (QLSAs) are widely used solvers in quantum computation, which play essential roles in solving problems in finance [1,2], bio-computing [3], fluid dynamics [4], machine learning [5], etc. Since Harrow, Hassidim, and Lloyd's pioneering presentation of the first quantum linear system solver [6], the QLSA has been proved to have an exponential acceleration over the classical one on the dependence of the problem size. In addition to the exponential acceleration achieved from the presentation of quantum states, much effort has been devoted to improving the dependence on the condition number, sparsity, and accuracy [7–12]. The quantum algorithm with $O[\kappa \log_2(1/\epsilon)]$ complexity for solving linear systems in [12] is asymptotically optimal, where κ is the condition number and ϵ is the error tolerance.

The above results mainly focus on linear equations with square coefficient matrix, i.e., the number of constraints (rows) and variables (columns) are equal, thus there exists one unique solution. However, when the coefficient matrices are not square, the unique solutions may not exist. We refer to this kind of linear system as an extended linear system (ELS) in this paper. ELS problems are commonly seen in areas such as image processing [13], machine learning [14], and computed tomography [15]. Under these circumstances, almost all existed methods could not work. First, the QLSAs mentioned above, e.g., solving the inverse matrices, are unsuitable because solving the Moore-Penrose pseudoinverse differs from solving an inverse matrix. Secondly, in fact, in classical computation theory, there are some standard methods for solving ELS problems, such as the QR decomposition (orthogonal matrix and upper triangular matrix decomposition) [16], the single value decomposition (SVD) [17], and iteration methods [18]. The time complexity of the QR decomposition and the SVD is $O(mn^2)$, where m represents the number of rows of the coefficient matrix and n means the number of columns. The time complexity for the iteration methods is $O(Kn)$, where K is the number of iteration steps that rely on the specific iteration methods, which explicitly or implicitly depend on m. As the dimension of the coefficient matrix increases, it becomes more complex and more costly to solve ELS problems.

So the natural question is whether quantum computation accelerates the process of solving these ELS problems. Two avenues may be feasible. First, we may consider transforming the nonsquare matrix into a square one and applying the QLSA to solve it. But we need to add more restrictions to the linear systems to achieve this. For example, we can introduce a square symmetric matrix $A' = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$ when solving equations $\boldsymbol{Ax} = \boldsymbol{b}$ with $A \in \mathbb{R}^{m\times n}$. However, A' is not full rank. Therefore, it is not invertible, which leads to a failure to solve it through solving the inverse of A'. Then, we consider constructing a square and invertible matrix from A. One choice is $(A^TA)^{-1}$ and the solution will be $x = (A^TA)^{-1}A^Tb$, but it needs A to be column full rank. While this idea has been employed in [19], that study focuses on designing a quantum-classical hybrid algorithm instead of a pure quantum one. Therefore, transforming the nonsquare matrix to a square one may not be an effective way to solve ELS problems.

The second avenue is to consider the quantum iteration method. Some QLSAs have the potential to solve the ELS problems through this idea [20,21]. The quantum one-row iteration method [21], the quantum version of the Kaczmarz

[*]Contact author: linweitao22s@ict.ac.cn
[†]Contact author: tianguojing@ict.ac.cn
[‡]Contact author: sunxiaoming@ict.ac.cn

(one-row) iteration method [22], is a viable technique to directly solve ELS, though that study focuses on the problem with the coefficient matrix being square. The running time of that algorithm is $O[\kappa_s^2(A)\log_2\frac{n}{\epsilon}]$, where $\kappa_s(A)=\frac{\|A\|_F}{\|A^{-1}\|}$, $\|A\|_F$ is the Frobenius norm of the matrix A, $\|A^{-1}\|$ is the norm of A^{-1}, n is the size of the problem, and ϵ is the error tolerance. The algorithm exhibits an exponential speedup over the randomized Kaczmarz algorithm (classical one-row iteration algorithm). Yet, the convergence rate remains similar to the classical one-row randomized Kaczmarz algorithm. Fortunately, we found that the classical multirow iterations in [23,24] demonstrated quicker convergence rates for solving linear systems with both square and nonsquare coefficient matrices. Naturally, we consider whether the quantum version of the multirow methods leads to a higher convergence rate while keeping the exponential speedup.

In this paper, we propose an approach that solves linear equations with a coefficient matrix of size $m\times n, m\geqslant n$, using the idea of linear combinations of unitaries [7]. We design a quantum multirow iteration algorithm with an explicit quantum circuit. The quantum circuit builds on the quantum comparator and quantum random access memory (QRAM), and can complete multirow iteration with different iteration weights. The gates used for one iteration step scale logarithmically on the problem size. Our quantum multirow iteration algorithm converges faster than the quantum one-row iteration algorithm [21] and keeps the same exponential speedup. This acceleration will yield considerable advantages in large-scale problems. In addition, since the convergence rate improvement is controlled by $\frac{\alpha_A^2}{q}$, where α_A is the relaxation parameter and q is the number of rows chosen, increasing α_A to a critical point and increasing q will not only significantly improve the convergence rate but also achieve a better convergence horizon. Moreover, regardless of the existence of the solution of the system, the solver will return an exact solution in consistent systems (systems with a unique solution) or the least-square solution in inconsistent systems (systems without a unique solution). The solver can also be used as a stochastic gradient descent iterator for some specific loss functions.

The outline of the paper is as follows. Section II gives some preliminaries including the classical multirow iteration method, the quantum one-row iteration, and block encoding. In Sec. III, we present our quantum multirow algorithm. We first show the key points of the algorithm. Then, we define the date structure for efficient state preparation and finally the sketch of the algorithm. Analysis for the resource is provided in Sec. IV. Numerical experiment is shown in Sec. V. A brief illustration of the application is presented in Sec. VI.

II. PRELIMINARIES

In this section, we will present some existing significant results including the classical multirow iteration method [24] and the quantum one-row iteration approach [21]. Next we will briefly introduce block encoding, which is a technique of embedding a nonunitary matrix into a large unitary one. The explicit quantum circuit for block encoding will be postponed to the next section.

A. Classical iteration method

In classical computation theory, the randomized one-row iteration method is usually used to solve linear systems, especially when the coefficient matrix is nonsquare [22]. Given $A\in\mathbb{R}^{m\times n}$ and $\boldsymbol{b}\in\mathbb{R}^m$, the purpose of iteration methods is to find $\boldsymbol{x}\in\mathbb{R}^n$ which satisfies the linear system of equations

$$A\boldsymbol{x}=\boldsymbol{b}. \tag{1}$$

Otherwise, we define the least-square solution

$$\boldsymbol{x}^*:=\arg\min_{x\in\mathbb{R}^n}\frac{1}{2}\|\boldsymbol{b}-A\boldsymbol{x}\|^2, \tag{2}$$

where $\|\cdot\|$ denotes the two-norm.

The randomized Kaczmarz iteration method [15,22] gives the following iteration scheme:

$$\boldsymbol{x}^{k+1}=\boldsymbol{x}^k+\frac{b_{i_k}-A_{i_k}\boldsymbol{x}^k}{\|A_{i_k}\|^2}A_{i_k}^T \tag{3}$$

where b_{i_k} is the i_kth element of the vector $\boldsymbol{b}$ and A_{i_k} is the i_kth row of the matrix A. This method is an alternating projection method. That is, $\boldsymbol{x}^{k+1}$ is the orthogonal projection of $\boldsymbol{x}^k$ onto the hyperplane $A_{i_k}\boldsymbol{x}=b_{i_k}$, via which it continuously approximates the exact solution by alternating projections. The randomized one-row iteration has been proven to have exponential convergence rates [18] and is generalized to solve inconsistent systems [25]. Many studies have attempted to enhance the rate of convergence for this approach [23,26–28], including the randomized multirow iteration method [24].

The classical multirow iteration method [24] gives the following iteration strategy.

Lemma 1 (see [24]). Given a linear system of equations, $A\boldsymbol{x}=\boldsymbol{b}$, where $A\in\mathbb{R}^{m\times n}, m\geqslant n$, and $\boldsymbol{b}\in\mathbb{R}^m$. Then, there exists a multirow iteration protocol

$$\boldsymbol{x}^{k+1}=\boldsymbol{x}^k+\frac{1}{q}\sum_{i\in\tau_k}\omega_i\frac{b_i-A_i\boldsymbol{x}^k}{\|A_i\|^2}A_i^T \tag{4}$$

where τ_k is a random set of q row indices sampled with replacement and ω_i represents the weight corresponding to the ith row.

Let $\boldsymbol{x}^*$ be the exact solution or least-square solution of $A\boldsymbol{x}=\boldsymbol{b}$ and let $\boldsymbol{e}^k=\boldsymbol{x}^k-\boldsymbol{x}^*$ be the iteration error of step k. We introduce the following definition to help show the convergence rate.

Definition 2. Let $\mathrm{diag}(d_1,d_2,\dots,d_m)$ denote the diagonal matrix with $d_1,d_2,\dots,d_m$ on the diagonal. Define the normalization matrix

$$D:=\mathrm{diag}(\|A_1\|,\|A_2\|,\dots,\|A_m\|) \tag{5}$$

such that the matrix $D^{-1}A$ has rows with unit norm, the probability matrix

$$P:=\mathrm{diag}(p_1,p_2,\dots,p_m) \tag{6}$$

where p_i denotes the probability of choosing the ith row, and the weight matrix

$$W:=\mathrm{diag}(\omega_1,\omega_2,\dots,\omega_m). \tag{7}$$

Thus, the convergence rate of Eq. (4) can be given as follows.

Lemma 3 (Theorem 1 [24]). Given the iteration strategy defined in Lemma 1 and supposing the matrices P and W in Definition 2 are chosen such that $PWD^{-2} = \frac{\alpha_A}{\|A\|_F^2}I$, the convergence rate satisfies

$$\mathbb{E}[\|\boldsymbol{e}^{k+1}\|^2] \leqslant \sigma_{\max}\left(\left(I - \alpha_A\frac{A^TA}{\|A\|_F^2}\right)^2 - \frac{\alpha_A^2}{q}\left(\frac{A^TA}{\|A\|_F^2}\right)^2\right)\|\boldsymbol{e}^k\|^2 + \frac{\alpha_A}{q}\frac{\|\boldsymbol{r}^k\|_W^2}{\|A\|_F^2} \tag{8}$$

where $\sigma_{\max}$ is the maximum singular value, $\|A\|_F^2 = \sum_{i,j}A_{ij}^2$, $\alpha_A > 0$ is the relaxation parameter, $\|\cdot\|_W^2 = \langle\cdot, W\rangle$, and $\boldsymbol{r}^k := \boldsymbol{b} - A\boldsymbol{x}^k$ is the residual of the kth iteration.

If the condition $PWD^{-2} = \frac{\alpha_A}{\|A\|_F^2}I$ is satisfied and q goes to infinity, the error will converge to zero. If the above condition is not satisfied, the iteration result will approach a weighted least-square solution instead of the least-square solution itself [24].

B. Quantum one-row iteration method

The quantum version of the one-row iteration method is given in [21]. Set $\boldsymbol{x}^k = \|\boldsymbol{x}^k\||x^k\rangle$ and $A_{i_k} = \|A_{i_k}\||A_{i_k}\rangle$; then, Eq. (3) can be rewritten as

$$|x^{k+1}\rangle = \|\boldsymbol{x}^k\|(I - |A_{i_k}\rangle\langle A_{i_k}|)|x^k\rangle + \frac{b_{i_k}}{\|A_{i_k}\|}|A_{i_k}\rangle, \tag{9}$$

up to a global phase.

Intuitively, one can define the following unitary as the iteration operator (assuming $\|A_{i_k}\| = 1$), which is the basic idea of [21]:

$$\begin{aligned} U_k &= \begin{bmatrix} I - |A_{i_k}\rangle\langle A_{i_k}| & |A_{i_k}\rangle\langle A_{i_k}| \\ |A_{i_k}\rangle\langle A_{i_k}| & I - |A_{i_k}\rangle\langle A_{i_k}| \end{bmatrix} \\ &= \big(I_2 \otimes V_{i_k}\big)(I_2 \otimes (I - |0\rangle\langle 0|) + X \otimes |0\rangle\langle 0|)\big(I_2 \otimes V_{i_k}^\dagger\big) \end{aligned} \tag{10}$$

where V_{i_k} represents the state preparation process $V_{i_k}|0\rangle = |A_{i_k}\rangle$, which can be achieved by the access to QRAM or by a state preparation operator. Applying the operator U_k on the state $\frac{\|x^k\|}{c}|0\rangle|\boldsymbol{x}^k\rangle + \frac{b_{i_k}}{c}|1\rangle|A_{i_k}\rangle$, where c is a normalization factor, can complete an iteration step. Similarly, one may employ a comparable approach in formulating the iteration operators for the multirow iteration method. However, the multirow iteration operator constructed this way is not unitary. The tricky part lies in how to tackle this problem. In the forthcoming sections, we will demonstrate how to realize the nonunitary operator in the multirow case.

C. Block encoding

To perform the nonunitary operator, we will use the technique of block encoding, which embeds the nonunitary operator into a large unitary one. The idea of block encoding [29,30] is widely used in the quantum algorithms associated with matrix multiplication [31] and Hamiltonian simulation [29].

Definition 4 (block encoding [31]). Assume A is an *s-qubit* operator; $\alpha, \epsilon \in \mathbb{R}^+$; and $a \in \mathbb{N}$. Then the $(s+a)$*-qubit* unitary operator U is an $(\alpha; a; \epsilon)$ block encoding of A, if

$$\|A - \alpha(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)\| \leqslant \epsilon, \tag{11}$$

where the norm is the two-norm. The parameters α and a are, respectively, the subnormalization factor of the matrix and the number of ancilla qubits used. Since $\|U\| = 1$, therefore $\|A\| \leqslant \alpha$.

Every unitary operator is already a $(1; 0; \epsilon)$ block encoding, and a nonunitary operator A can be embedded in a $(\|A\|;a;\epsilon)$ block encoding. Given a block encoding U, one can prepare the state $\frac{A|\psi\rangle}{\|A|\psi\rangle\|}$ from an initial state $|0\rangle|\psi\rangle$, that is, $U|0\rangle|\psi\rangle = |0\rangle\frac{A|\psi\rangle}{\|A|\psi\rangle\|}$. This concept enhances the applicability of quantum linear algebra, thus making it more widely applicable. In this paper, based on the existence of the unitary operator U, we further present a specific construction of U by decomposing it into the quantum elementary gates.

III. QUANTUM MULTIROW ITERATION ALGORITHM

In this section, we show several problems needed to be conquered before designing the explicit quantum circuit for the quantum multirow iteration algorithm and provide a sketch of our algorithm. This section is organized as follows. In Sec. III A, we show the main difficulties and the methods to solve them. In Sec. III B, we define the data structure which is used as an efficient state preparation process in the algorithm. The sketch of the algorithm is given in Sec. III C.

A. Key points of the algorithm implementation

In this part, we will clarify the key points of the algorithm implementation, which conquer the main barrier of designing the algorithm. The barrier is the nonunitarity of the matrix corresponding to the idea of the quantum one-row iteration and we should design a proper method to implement the nonunitary operator.

1. Nonunitarity of the matrix

Suppose the coefficient matrix is $A \in \mathbb{R}^{m\times n}$. Based on Eq. (4), setting $\boldsymbol{x}^k = \|\boldsymbol{x}^k\||x^k\rangle$ and $A_i = \|A_i\||A_i\rangle$, we can derive the following formula:

$$\begin{aligned} |x^{k+1}\rangle =& \|\boldsymbol{x}^k\|\left(I - \frac{1}{q}\sum_{i\in\tau_k}\omega_i|A_i\rangle\langle A_i|\right)|x^k\rangle \\ &+ \frac{1}{q}\sum_{i\in\tau_k}\frac{\omega_i b_i}{\|A_i\|}|A_i\rangle, \end{aligned} \tag{12}$$

up to a global phase, where A_i is the vector of the ith row. This is one iteration step from k to $k+1$. The analog of the iteration operator shown in Eq. (10) in the quantum multirow iterations is given as follows:

$$T_k = \begin{bmatrix} I - \frac{1}{q}\sum_{i\in\tau_k}\omega_i|A_i\rangle\langle A_i| & \frac{1}{q}\sum_{i\in\tau_k}\omega_i|A_i\rangle\langle A_i| \\ \frac{1}{q}\sum_{i\in\tau_k}\omega_i|A_i\rangle\langle A_i| & I - \frac{1}{q}\sum_{i\in\tau_k}\omega_i|A_i\rangle\langle A_i| \end{bmatrix}. \tag{13}$$

Unfortunately, unlike the matrix given in Eq. (10), T_k is not unitary. We show this nonunitarity through calculating the upper left corner of $T_kT_k^\dagger$, in which we assume that all the elements are real numbers for simplicity:

$$\left(I-\frac{1}{q}\sum_{i\in\tau_k}\omega_i|A_i\rangle\langle A_i|\right)^2+\left(\frac{1}{q}\sum_{i\in\tau_k}\omega_i|A_i\rangle\langle A_i|\right)^2$$
$$=I-\frac{2}{q}\sum_{i\in\tau_k}\omega_i|A_i\rangle\langle A_i|+2\left(\frac{1}{q}\sum_{i\in\tau_k}\omega_i|A_i\rangle\langle A_i|\right)^2$$
$$\neq I. \tag{14}$$

The rest of $T_kT_k^\dagger$ can be similarly computed. It is indicated that T_k is not a unitary operator. There are two reasons for this nonunitarity. First, the row vectors of matrix A are not orthogonal to each other; second, the iteration weight of each iteration row is not necessarily 1.

2. Overcoming the nonunitarity

Causes for the nonunitarity are already given, i.e., nonorthogonality of rows and nonunity of weight, and next we will overcome them.

a. Orthogonality. We introduce ancillary qubits to generate orthogonality. The state $|x^k\rangle$ is replaced by $\sum_{i\in\tau_k}s_{k,i}|i\rangle|x^k\rangle$, where $s_{k,i}$ represents the amplitude labeled by i. This term is the result of the previous $(k-1)$th iteration step after exchanging the index set from τ_{k-1} to τ_k. To be consistent with Eq. (12), we use $\omega_{k,i}$ as a modified weight term, which represents $\frac{\omega_i}{q}$. We use $s'_{k+1,i}$ to represent the weight of the iteration result before changing the index set from τ_k to τ_{k+1}. Then we derive the iteration step with ancillary qubits below:

$$\sum_{i\in\tau_k}s'_{k+1,i}|i\rangle|x^{k+1}\rangle=\|\boldsymbol{x}^k\|(I-\Sigma_{\tau_k})\sum_{i\in\tau_k}s_{k,i}|i\rangle|x^k\rangle$$
$$+\sum_{i\in\tau_k}\frac{\omega_{k,i}b_i}{\|A_i\|}s_{k,i}|i\rangle|A_i\rangle, \tag{15}$$

where $\Sigma_{\tau_k}=\sum_{i\in\tau_k}\omega_{k,i}(|i\rangle\langle i|\otimes|A_i\rangle\langle A_i|)$. A simple preprocessing procedure can obtain this rescaled weight $\omega_{k,i}$, as all the weights are artificially chosen. The orthogonality of $|i\rangle$ makes the nonorthogonality of rows of A orthogonal. To cancel the effect of the norm $\|A_i\|$, we need to perform a preprocessing procedure to update b_i as $\frac{b_i}{\|A_i\|}$ to achieve the same effect.

b. Weight. The usage of the index register $|i\rangle$ reduces the difficulty of directly attaching the iteration weight to $|A_i\rangle\langle A_i|$. We may achieve $\sum_{i\in\tau_k}\omega_{k,i}|i\rangle\langle i|$ instead. We apply the following operator:

$$\left(\frac{\omega_{k,i}}{2}|j\rangle\langle j|+\frac{\omega_{k,i}}{2}|j+m\rangle\langle j+m|\right)\otimes(|i\rangle\langle i|),$$
$$j=i, i\in\tau_k. \tag{16}$$

If $\sum_{i\in\tau_k}\omega_{k,i}=1$, a state preparation process and its inverse are enough. However, in the general case $\sum_{i\in\tau_k}\omega_{k,i}\neq 1$. Applying a simple state preparation process will result in some redundant parts. To eliminate the influence of the redundant parts, we apply the following operator:

$$(r_{k,i}|j\rangle\langle j|-r_{k,i}|j+m\rangle\langle j+m|)\otimes(|i\rangle\langle i|),\quad j=i, i\notin\tau_k \tag{17}$$

where m is the number of rows of the matrix A and $r_{k,i}$ represents the amplitude of the redundant states. Applying the above operators on the state $\sum_i\sum_i|j\rangle|i\rangle$ has the same effect as applying $\sum_{i\in\tau_k}\omega_{k,i}|i\rangle\langle i|$ on the state $\sum_i|i\rangle$.

Introducing orthogonality and using the block encoding to apply the weight, we can therefore apply the following iteration matrix U_k:

$$U_k=\begin{bmatrix} I-\Sigma_{\tau_k} & \Sigma_{\tau_k}\\ \Sigma_{\tau_k} & I-\Sigma_{\tau_k}\end{bmatrix} \tag{18}$$

where $\Sigma_{\tau_k}=\sum_{i\in\tau_k}\omega_{k,i}(|i\rangle\langle i|\otimes|A_i\rangle\langle A_i|)$. It should be noted that we also use "U_k" denoting the multirow iteration operator.

3. Equivalent implementation

The problem remaining is how to efficiently implement Eqs. (16) and (17) by quantum circuits. Our idea is to consider the application of the weight factor and the process of selecting the desired index $|i\rangle$ separately. Specifically, we rewrite the procedure into the block encoding form:

$$\langle 0|(G\otimes I)\tilde{U}_k(G^\dagger\otimes I)|0\rangle \tag{19}$$

where G is a state preparation operator,

$$G|0\rangle=\sum_{\substack{j\in\tau_k,\\ j\in[m]}}\sqrt{\frac{\omega_{k,j}}{2}}|j\rangle+\sum_{\substack{(j-m)\in\tau_k,\\ j\in[2m]/[m]}}\sqrt{\frac{-\omega_{k,j-m}}{2}}|j\rangle$$
$$+\sum_{\substack{j\notin\tau_k,\\ j\in[m]}}\sqrt{r_{k,j}}|j\rangle+\sum_{\substack{(j-m)\notin\tau_k,\\ j\in[2m]/[m]}}\sqrt{-r_{k,j-m}}|j\rangle, \tag{20}$$

with $[m]=\{0,1,2,\dots,m-1\}$, and $\tilde{U}_k$ is a linear combination of unitary

$$\tilde{U}_k=\sum_{j=0}^{m-1}|j\rangle\langle j|\otimes C_1^{(j,j)}+\sum_{\substack{j=m\\ j-m\in\tau_k}}^{2m-1}|j\rangle\langle j|\otimes C_{-1}^{(j-m,j-m)}$$
$$+\sum_{\substack{j=m\\ j-m\notin\tau_k}}^{2m-1}|j\rangle\langle j|\otimes C_1^{(j-m,j-m)} \tag{21}$$

where $C_h^{(j,j)}$, $h\in\{1,-1\}$, is defined as

$$\left(\sum_{l=0}^{m-1}|(2j-l)\mod m\rangle\langle l|\right)\left(\sum_{l\neq j}|l\rangle\langle l|+h|j\rangle\langle j|\right). \tag{22}$$

It is obvious that $C_h^{(j,j)}$ is unitary when $h=1$ or -1. $\frac{1}{2}(C_1^{(i,i)}-C_{-1}^{(i,i)})$ represents a matrix with the ith diagonal element being 1, which is $|i\rangle\langle i|$, and $\frac{1}{2}(C_1^{(i,i)}-C_1^{(i,i)})$ is a matrix with all elements being zero. The main point of our idea is to apply an operator equivalent to the following linear combination:

$$\sum_{i\in\tau_k}\omega_{k,i}|i\rangle\langle i| = \sum_{i\in\tau_k}\left(\frac{\omega_{k,i}}{2}C_1^{(i,i)} - \frac{\omega_{k,i}}{2}C_{-1}^{(i,i)}\right) + \sum_{i\notin\tau_k}\left(\frac{r_{k,i}}{2}C_1^{(i,i)} - \frac{r_{k,i}}{2}C_1^{(i,i)}\right). \tag{23}$$

To give an explicit circuit for this block encoding, we should consider how to implement the linear combination of unitary $\tilde{U}_k$, since G is a state preparation operator. We can use the idea of equivalent implementation to achieve this, which is inspired by [32].

The application of $\tilde{U}_k$ on the basis state is given as

$$\tilde{U}_k|j\rangle|l\rangle = \begin{cases} |j\rangle|(2j-l) \mod m\rangle, & 0 \leqslant j \leqslant m-1 \\ |j\rangle|(2j-l) \mod m\rangle, & m \leqslant j \leqslant 2m-1 \\ -|j\rangle|(2j-l) \mod m\rangle, & m \leqslant j \leqslant 2m-1, l = (j \mod m), j-m \in \tau_k \end{cases}. \tag{24}$$

This can be equivalently implemented by a process U_{eq}, which contains a quantum comparator [33,34], a quantum modular adder [33,34], and a NOT gate controlled by the comparison results. The comparator selects the states which meet the condition and the modular adder is used to prepare $|(2j-l) \mod m\rangle$. Introducing an ancilla state, which is initialized in $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$, the NOT gate will be applied to this state when the condition $m \leqslant j \leqslant 2m-1, l = (j \mod m), j-m \in \tau_k$ is satisfied. The action of the NOT gate is equivalent to multiplying -1. More details are given in Appendix A.

B. Data structure

We can correspondingly define an iteration matrix U_k from the iteration process as given in Eq. (15). This operator can complete the kth iteration through operating on $\beta_k|0\rangle\sum_{i\in\tau_k}s_{k,i}|i\rangle|x^k\rangle + |1\rangle\sum_{i\in\tau_k}\gamma_{k,i}s_{k,i}|i\rangle|A_i\rangle$, where β_k and $\gamma_{k,i}$ are factors associated with $\|\boldsymbol{x}^k\|$ and b_i. Similar to Eq. (10), we rewrite the matrix as

$$U_k = I_2 \otimes \left(I - \tilde{V}\sum_{i\in\tau_k}\omega_{k,i}|i\rangle|0\rangle\langle 0|\langle i|\tilde{V}^\dagger\right) + X \otimes \left(\tilde{V}\sum_{i\in\tau_k}\omega_{k,i}|i\rangle|0\rangle\langle 0|\langle i|\tilde{V}^\dagger\right) \tag{25}$$

where the implementation of $\sum_{i\in\tau_k}\omega_{k,i}|i\rangle\langle i|$ is achieved by $\langle 0|(G\otimes I)\tilde{U}_k(G^\dagger\otimes I)|0\rangle$ and $\tilde{V}\sum_i|i\rangle|0\rangle = \sum_i|i\rangle|A_i\rangle$.

In addition to the implementation of $\tilde{U}_k$, the efficiency of the algorithm relies on the state preparation processes G and $\tilde{V}$. To efficiently prepare the row vectors and the weight vectors, we introduce a data structure as QRAM, which is different from the one in [35].

This data structure can be visited by an algorithm that outputs the quantum state $|A_i\rangle, A_i \in \mathbb{R}^n$ corresponding to the ith row A_i of the matrix A, and the quantum state $|\boldsymbol{\omega}\rangle, \boldsymbol{\omega} \in \mathbb{R}^{2m}$ corresponding to the weight vector with the redundant part. Specifically, we denote the procedures as unitary operators V and G, where $V|i\rangle|0\rangle = |i\rangle|A_i\rangle$ and $G|0\rangle = |\boldsymbol{\omega}\rangle$. And we denote the multirow readout process as $\tilde{V}\sum_i|i\rangle|0\rangle = \sum_i|i\rangle|A_i\rangle$.

We introduce two kinds of binary trees for the data structure: one is the address tree, and the other is the memory tree. The address tree possesses m leaves, and each leaf stores the address (see Fig. 1). Accessing the address tree can be analogized to a routing process. The corresponding memory tree will be activated once the addressing qubits have been set. There are two types of memory trees: one with n leaves called the data tree, which stores the row vector, and the other with $2m$ leaves called the weight tree, which stores the weight vector. m data trees accompanied with one address tree accomplish the process $\tilde{V}\sum_i|i\rangle|0\rangle = \sum_i|i\rangle|A_i\rangle$ and the weight tree achieves $G|0\rangle = |\boldsymbol{\omega}\rangle$. The leaves of the data trees and weight tree hold the individual amplitudes of the vector, and each internal node holds the square root of the sum of the squares of the norm for the value in children nodes (see Fig. 2 as an example). For a single reading procedure, when the binary tree gets an address as input, which can be a superposition state, the data structure finds the path toward the data trees based on the address. The data structure accesses the address in time $O(\log_2 m)$, and each data tree prepares the states in time $O(\log_2 n)$ for the row vector or $O(\log_2 m)$ for the weight vector. The writing process can be completed for the same cost of time. We summarize this data structure in the following definition.

Definition 5. Suppose $A \in \mathbb{R}^{m\times n}$ is a matrix and $\boldsymbol{\omega} \in \mathbb{R}^{2m}$ is a vector. There exists a data structure with the following properties.

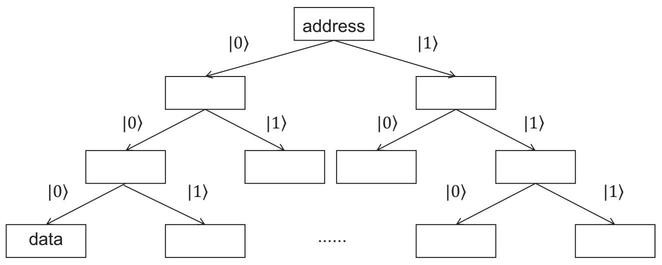

FIG. 1. Schematic diagram of data structure. The root gets an address as an input and finds the routes to the corresponding leaves based on each qubit of the address. Each leaf points to a data tree, which stores the row vector of the matrix A.

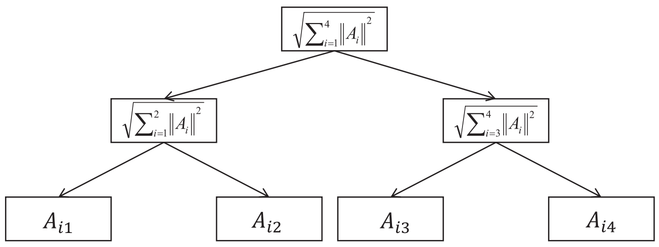
$\sqrt{\sum_{i=1}^{4}\|A_i\|^2}$
$\sqrt{\sum_{i=1}^{2}\|A_i\|^2}$
$\sqrt{\sum_{i=3}^{4}\|A_i\|^2}$
A_{i1}
A_{i2}
A_{i3}
A_{i4}

FIG. 2. An example of the data tree with $n = 4$. The leaves hold the individual amplitudes of the elements of the vector, and each internal node holds the square root of the sum of the squares of the norm for the value in children nodes.

(1) It can be visited by a quantum algorithm that can perform the mapping $V : |i\rangle|0\rangle \rightarrow |i\rangle|A_i\rangle$ for $i \in \{1, 2, \dots, m\}$ in time poly$[\log_2(m+n)]$.

(2) It can be visited by another quantum algorithm that can perform the mapping $G : |0\rangle \rightarrow |\boldsymbol{\omega}\rangle$ in time poly$(\log_2 m)$.

C. Sketch of the algorithm

From the above subsections, we already have the way to construct the iteration matrix and the efficient way to prepare the states. It is enough to design the quantum multirow iteration algorithm.

The explicit sketch of the implementation of the multirow iteration algorithm is stated as follows. We will explain the algorithm step by step.

We prepare the initial state in step 1. In step 2, we define the parameters β_k, $\gamma_{k,i}$, and v_k. $\gamma_{k,i}$ helps to apply b_i, v_k is the normalized factor of each iteration, and β_k is introduced for the purpose of normalization. We can first prepare $\beta_k|0\rangle\sum_{i\in\tau_k}|i\rangle|0\rangle + |1\rangle\sum_{i\in\tau_k}\gamma_{k,i}|i\rangle|0\rangle$ through some rotation gates, then use the controlled operation to apply $|X^k\rangle$ and $|A_i\rangle$.

In step 3, we can obtain the following state after applying the SWAP gate:

ALGORITHM I. Quantum multirow iteration algorithm.

Input: Randomly choose a unit vector x^1. Set $k = 1$, $v_1 = 1$, and the maximum number of iteration steps is K.
Output: $|0\rangle^{\otimes(K-1)}|0\rangle^{\otimes\log_2\lceil q \mod m\rceil}|x^K\rangle + |Gb\rangle$, where $|Gb\rangle$ is the garbage state.
Procedure:

1. Randomly choose q elements from set $\{0, 1, \dots, m-1\}$ as an index set τ_1. The weights $s_{1,i}$, $i \in \tau_1$ are set uniformly. Prepare the state
$$|X^1\rangle = \frac{\|x^1\|}{v_1}\sum_{i\in\tau_1} s_{1,i}|i\rangle|x^1\rangle. \tag{26}$$
2. Define $\beta_k = \frac{v_k}{\sqrt{v_k^2+\sum_{i\in\tau_k}\|b_i\|^2}}$ and $\gamma_{k,i} = \frac{b_i}{\sqrt{v_k^2+\sum_{i\in\tau_k}\|b_i\|^2}}$, $i \in \tau_k$. Set $v_{k+1} = \frac{v_k}{\beta_k}$. Then, through some rotation gates and controlled operation, obtain the state
$$|Y^k\rangle = \beta_k|0\rangle|X^k\rangle + |1\rangle|0\rangle^{\otimes(k-1)} \otimes \sum_{i\in\tau_k}\gamma_{k,i}s_{k,i}|i\rangle|A_i\rangle. \tag{27}$$
3. Apply $(I_2^{\otimes(k-1)} \otimes U_k)SWAP_{1,k}$ to $|Y^k\rangle$, then we can obtain
$$|Z^{k+1}\rangle = \frac{\|x^{k+1}\|}{v_{k+1}}|0\rangle^{\otimes k} \otimes \sum_{i\in\tau_k} s'_{k+1,i}|i\rangle|x^{k+1}\rangle + |Gb\rangle. \tag{28}$$
4. Randomly choose q elements from set $\{0, 1, \dots, m-1\}$ as an index set τ_{k+1} with $k = 1, 2, 3, \dots, K-1$. Implement the exchange operator P to swap the ancillas from $\sum_{i\in\tau_k}|i\rangle$ to $\sum_{i\in\tau_{k+1}}|i\rangle$, then obtain the following state:
$$|X^{k+1}\rangle = \frac{\|x^{k+1}\|}{v_{k+1}}|0\rangle^{\otimes k} \otimes \sum_{i\in\tau_{k+1}} s_{k+1,i}|i\rangle|x^{k+1}\rangle + |Gb\rangle. \tag{29}$$
5. Set $k = k+1$; if the maximum number of iterations is not satisfied, turn to 2; else, turn to 6.
6. Perform an adding procedure U_{add}, then we obtain the output state.

$$\text{SWAP}_{1,k}\left(\beta_k|0\rangle|X^k\rangle + |1\rangle|0\rangle^{\otimes(k-1)} \otimes \sum_{i\in\tau_k}\gamma_{k,i}s_{k,i}|i\rangle|A_i\rangle\right) = |0\rangle^{\otimes(k-1)}\left(\|\boldsymbol{x}^k\|\beta_k|0\rangle\sum_{i\in\tau_k}s_{k,i}|i\rangle|x^k\rangle + |1\rangle\sum_{i\in\tau_k}\gamma_{k,i}s_{k,i}|i\rangle|A_i\rangle\right). \tag{30}$$

Then, applying the operator U_k yields

$$\begin{aligned}
&\left(I_2^{\otimes(k-1)} \otimes U_k\right)\left(\|\boldsymbol{x}^k\|\beta_k|0\rangle\sum_{i\in\tau_k}s_{k,i}|i\rangle|x^k\rangle + |1\rangle\sum_{i\in\tau_k}\gamma_{k,i}s_{k,i}|i\rangle|A_i\rangle\right)\\
&= |0\rangle^{\otimes k}\left(\|\boldsymbol{x}^k\|\beta_k\left(I - \Sigma_{\tau_k}\right)\sum_{i\in\tau_k}s_{k,i}|i\rangle|x^k\rangle + \sum_{i\in\tau_k}\gamma_{k,i}s_{k,i}|i\rangle|A_i\rangle\right) + |Gb\rangle\\
&= \frac{\beta_k}{v_k}|0\rangle^{\otimes k}\left(\|\boldsymbol{x}^k\|\left(I - \Sigma_{\tau_k}\right)\sum_{i\in\tau_k}s_{k,i}|i\rangle|x^k\rangle + \sum_{i\in\tau_k}\omega_{k,i}b_i s_{k,i}|i\rangle|A_i\rangle\right) + |Gb\rangle\\
&= |Z^{k+1}\rangle.
\end{aligned} \tag{31}$$

The first term of the third line is quite similar to the result given in Eq. (15) with a slight difference in the normalization factor. Then, in step 4, we can move the index register to a new subspace with an exchange operator P:

$$P\sum_{i\in\tau_k} s'_{k+1,i}|i\rangle|x^{k+1}\rangle = \sum_{i\in\tau_{k+1}} s_{k+1,i}|i\rangle|x^{k+1}\rangle. \tag{32}$$

The above process completes the iteration of a step. Finally, once a predetermined number of iterations or other termination criteria have been met, a quantum-adding procedure is carried out to derive the ultimate outcome:

$$U_{\text{add}}\sum_{i\in\tau_k} s_{k,i}|i\rangle|x^k\rangle = |0\rangle|x^k\rangle. \tag{33}$$

Such an operator can be applied; for example, we can apply a set of Hadamard gates to accomplish this with the help of oblivious amplitude amplification [36] or design an exact operator to achieve this.

Theorem 6. Assume the memory access operator G and V as defined in Definition 5. In Algorithm 1, for any $K \geqslant 1$, the time complexity to prepare $|X^K\rangle$ is

$$O(K\log_2 m). \tag{34}$$

IV. ANALYSIS FOR THE RESOURCE REQUIREMENT

In the preceding section, we give the sketch of the algorithm and the main techniques we use to design the explicit circuit. In this section, we analyze the resource requirement of the algorithm. We suppose the summation of the weights satisfies $\sum_{i\in\tau_k}\omega_{k,i} = t_k$ and the matrix A is given as $A\in\mathbb{R}^{m\times n}$. The outline of this section is as follows. In Sec. IV A, we analyze the resources needed for the kth iteration step. Then, we analyze the iteration steps needed in Sec. IV B, which are almost the same as in the classic scenario.

A. Analysis for the *k*th iteration step

The techniques which have been shown before can fulfill the box in the following equation:

$$U_k = I_2\otimes\left(I - \boxed{\tilde{V}\sum_{i\in\tau_k}\omega_{k,i}|i\rangle|0\rangle\langle 0|\langle i|\tilde{V}^\dagger}\right) + X\otimes\left(\boxed{\tilde{V}\sum_{i\in\tau_k}\omega_{k,i}|i\rangle|0\rangle\langle 0|\langle i|\tilde{V}^\dagger}\right). \tag{35}$$

To implement the operator U_k, we should implement a controlled version of $\tilde{U}_k$ and controlled memory access operators G and $\tilde{V}$. The circuit for U_k is shown in Fig. 3. The circuit before the dotted box in Fig. 3 applies the operator in the box of Eq. (35) on the states marked by $|010\rangle$, $|011\rangle$, and $|001\rangle$. Then, applying the circuit in the dotted box, we realize the operator U_k. More details are given in Appendix A.

Remark 7. This circuit may trigger confusion for some readers because it does not look as symmetrical as some of the common quantum circuits. However, because the quantum comparators have a symmetric structure, it is quite possible that we can design a symmetrical structure and set the middle part as a multiqubit controlled NOT (CNOT) gate, which completes $I_2\otimes(I-\sum_{i\in\tau_k}|i\rangle\langle i|) + X\otimes\sum_{i\in\tau_k}|i\rangle\langle i|$. But, this involves the specific design of multiple quantum comparators combined, therefore we leave this to future work.

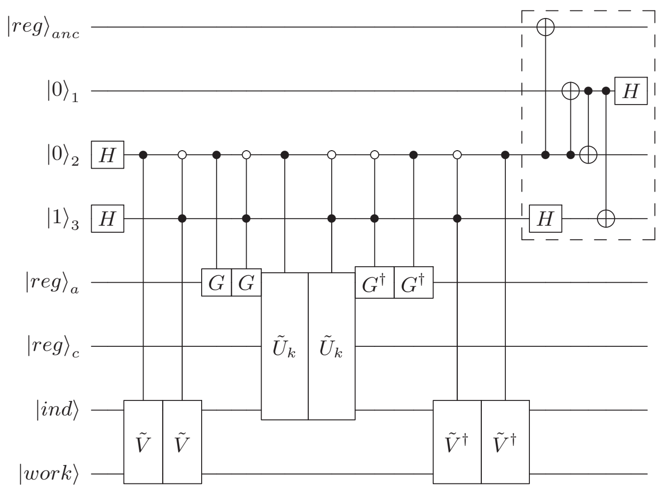

FIG. 3. Quantum circuit implementation of the operator U_k.

The complexity to apply the classical multirow iteration algorithm for one step is $O(m)$ [24]. For the quantum multirow iteration algorithm, as shown in Algorithm 1, the complexity to complete one iteration step is $O(\log_2 m)$. This exhibits an exponential speedup. For a more formal version, the complexity of the quantum multirow iteration algorithm for one iteration step is given by the following theorem.

Theorem 8. Given a system of linear equations, we have $Ax = b, A\in\mathbb{R}^{m\times n}$, and the operators G and $\tilde{V}$. For any kth step, the quantum multirow iteration algorithm can output the state $|x^{k+1}\rangle$ with high probability using $O(\sqrt{\frac{V_{k+1}^2}{t_k}})$ queries to G and $O(\sqrt{V_{k+1}^2})$ queries to $\tilde{V}$, $O(\sqrt{\frac{V_{k+1}^2}{t_k}}\log_2 m)$ extra elementary gates, and $O(\log_2 m)$ ancillary qubits.

The proof of the theorem is given in Appendix B.

B. Analysis for the number of iteration steps

Section II A provides the condition, $PWD^{-2} = \frac{\alpha_A}{\|A\|_F^2}I$, to choose the proper selection probability, which affects the address tree, and the iteration weights. Here we analyze the difference in the convergence rate between the classical setting and the quantum setting.

Without loss of generality, we assume that the norm of each row of the matrix A satisfies $\|A_i\| = 1$. This gives the condition $PW = \frac{\alpha_A}{m}I$. In the quantum setting, we have the condition $\sum_{i\in\tau_k}\omega_{k,i}\leqslant 1$, because of the need for normalization. As shown in Sec. III A 2, $\omega_{k,i}$ is a modified term, which is determined by ω_i and q. Therefore, in the quantum setting, the choice of weights should satisfy $\sum_{i\in\tau_k}\omega_i\leqslant q$. If any row is selected with equal probability and each selected row has the same iteration weight $\omega_i\leqslant 1$, then we have the condition $PW\leqslant\frac{1}{m}I$. It should be noted that $\omega_i\leqslant 1$ is a condition derived from $\sum_{i\in\tau_k}\omega_i\leqslant q$, since τ_k has q elements (may have duplicate elements). Therefore, the parameter α_A should satisfy $\alpha_A \leqslant 1$. This is the difference between the classical setting and the quantum setting. We can obtain the convergence rate in the quantum setting based on this.

Lemma 9 (convergence rate in the quantum setting). Given the quantum multirow iteration algorithm and supposing the matrices P and W in Definition 2 are chosen such that $PWD^{-2} = \frac{\alpha_A}{\|A\|_F^2} I$, the convergence rate of the algorithm satisfies

$$\mathbb{E}[\|\boldsymbol{e}^{k+1}\|^2] \leqslant \sigma_{\max}\left(\left(I-\alpha_A \frac{A^T A}{\|A\|_F^2}\right)^2 - \frac{\alpha_A^2}{q}\left(\frac{A^T A}{\|A\|_F^2}\right)^2\right)\|\boldsymbol{e}^k\|^2 + \frac{\alpha_A}{q}\frac{\|\boldsymbol{r}^k\|_W^2}{\|A\|_F^2} \tag{36}$$

where $\boldsymbol{e}^k = \boldsymbol{x}^k - \boldsymbol{x}^*$ is the iteration error between $\boldsymbol{x}^k$ and the solution or least-square solution $\boldsymbol{x}^*$ of step k, $\sigma_{\max}$ is the maximum singular value, $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$, $0 < \alpha_A \leqslant 1$ is the relaxation parameter, $\|\cdot\|_W^2 = \langle\cdot, W\rangle$, and $\boldsymbol{r}^k := \boldsymbol{b} - A\boldsymbol{x}^k$ is the residual of the kth iteration.

The proof of the convergence rate is given in Appendix C. The expression for the convergence rate is a little bit complex and therefore it is hard to tell how much the quantum multirow iteration algorithm outperforms the quantum one-row iteration algorithm. We will show this in the next section by numerical test. Also, the influence of the difference of the parameter α_A will also be shown numerically.

V. NUMERICAL RESULTS

In this section, we conduct some numerical tests to show the advantage of the quantum multirow iteration over the quantum one-row iteration. Moreover, we show the effect of the parameter α_A to illustrate the difference between the quantum algorithm and the classic one.

For each numerical test, we run 100 independent trials and evaluate the average squared error norms $\|\boldsymbol{e}^k\|^2$ across the trials, where $\boldsymbol{e}^k = \boldsymbol{x}^k - \boldsymbol{x}^*$. We get the iteration result $\boldsymbol{x}^k$ by multiplying the value v_{k+1} with the state vector $|\boldsymbol{x}^k\rangle$, which is obtained through the state-vector compiler in QISKIT. The matrix A is a 100×4 Gaussian matrix with each row normalized. The least-squares solution $\boldsymbol{x}^*$ is a four-dimensional Gaussian vector and satisfies $\|\boldsymbol{x}^*\| = 1$. We randomly choose a residual $\boldsymbol{r}^*$ and normalize it so that $\|\boldsymbol{r}^*\| = 1$. $\boldsymbol{b}$ is computed as $\boldsymbol{r}^* + A\boldsymbol{x}^*$.

In Fig. 4, we compare the quantum one-row iteration with the quantum multirow iteration, which uses uniform weights for simplicity. The figure shows that the quantum multirow iteration possesses a better convergence rate and a smaller convergence radius. Moreover, the quantum multirow iteration can reach a better convergence rate as more rows are selected.

In Fig. 5, we compare the effect of different choices of α_A. The classic algorithm achieves the results with $\alpha_A > 1$. When α_A reaches a critical point, the algorithm can have a slightly better convergence rate. The larger choice of α_A has a worse convergence radius. It is shown in [24] that the optimal choice of α_A is not large. Therefore, the drawback that the quantum multirow iteration cannot choose a large α_A is not severe.

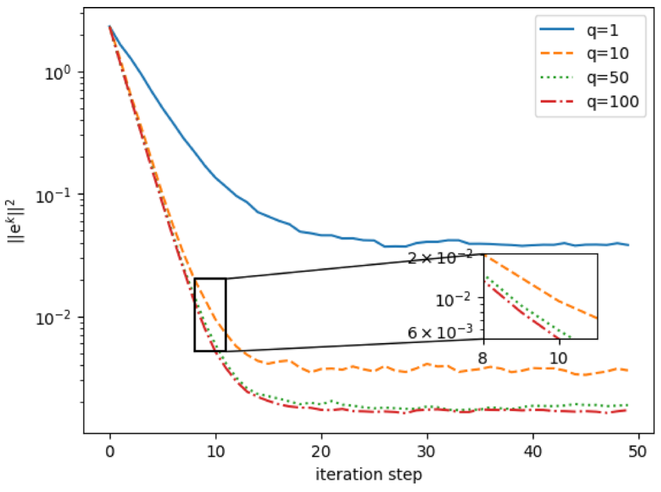

FIG. 4. Comparison of the quantum one-row iteration and the quantum multirow iteration with different choices of the number of iteration rows. q is the number of rows selected in each iteration. $e^k = x^k - x^*$ is the error between the iteration result and the solution or least-square solution. Smaller $\|e^k\|$ indicates a smaller convergence radius.

VI. MORE APPLICATIONS

Our method not only serves as a solver for overdetermined equations but also functions as a solver for linear systems and stochastic gradient descent.

If $Ax = b, A \in \mathbb{R}^{n\times n}$, our methods can serve as a linear system solver. There is no need to assume that the equations are consistent, which implies that at least one solution to the given equation exists because the randomized Kaczmarz iteration method has the potential to solve the inconsistent problem.

Using the algorithm defined in the previous section, we give the following theorem.

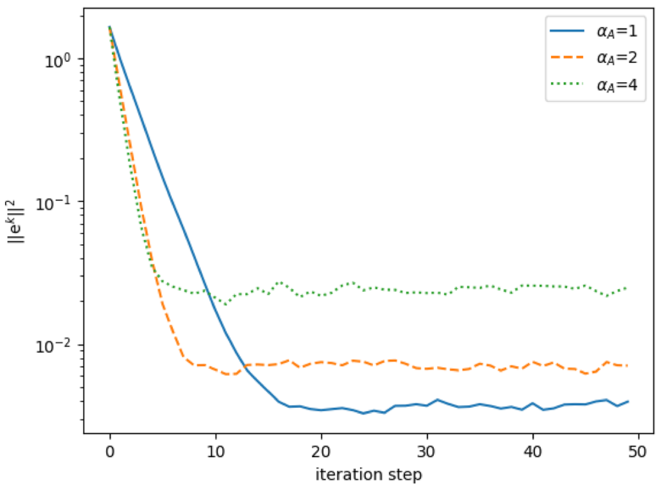

FIG. 5. Comparison of the effect of different choices of α_A. The classic algorithm achieves the results with $\alpha_A > 1$. We choose uniform weights, and the number of iteration rows satisfies $q = 10$. $e^k = x^k - x^*$ is the error between the iteration result and the solution or least-square solution.

Corollary 10. Given a system of linear equations, $Ax = b, A \in \mathbb{R}^{n\times n}$. Suppose the iteration error $\|\boldsymbol{e}^k\| \leqslant \epsilon$ is satisfied after K iteration steps. The quantum algorithm defined in Algorithm 1 can prepare the state that encodes the solution or least-square solution of such equations within the error $\|e^K\|$ with high probability. The query complexity is $O(K)$ and the gate complexity is $O(K \log_2 n)$. The iteration step K has a maximum value $K_{\max} = \kappa_s^2 \log_2(\frac{1}{\epsilon})$ when $q = 1$, where $\kappa_s = \|A\|_F \|A^{-1}\|$ and $\|A\|_F$ is the Frobenius norm.

The term $\kappa_s = \|A\|_F \|A^{-1}\|$, where $\|A\|_F$ is the Frobenius norm of the matrix A and ϵ is the tolerance error. $k_{\max}$ only occurs when only one selected iteration row ($q = 1$) exists.

The randomized Kaczmarz iteration can be viewed as a subcase of stochastic gradient descent for the following loss function [37]:

$$F(x) = \sum_{i=1}^{n} f_i(x) = \sum_{i=1}^{n} \frac{1}{2}(a_i x - b_i)^2 \tag{37}$$

which covers the case when the gradient is an affine function for quadratic optimization problems of the form $\min_{x\in\mathbb{R}^n} x^T A x + b^T x + c$ for $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n, c \in \mathbb{R}$ [20]. Therefore, the randomized Kaczmarz iteration method is a reweighted version of the stochastic gradient descent:

$$\begin{aligned} x^{k+1} &= x^k - \frac{a_i x^k - b_i}{\|a_i\|^2} a_i^T \\ &= x^k - \frac{\nabla f_i(x)}{\|a_i\|^2}. \end{aligned} \tag{38}$$

The multirow iteration method can be seen as the mini-batch stochastic gradient descent [24]:

$$\begin{aligned} x^{k+1} &= x^k - \frac{1}{q}\sum_{i\in\tau_k} \omega_i \frac{a_i x^k - b_i}{\|a_i\|^2} a_i^T \\ &= x^k - \frac{1}{q}\sum_{i\in\tau_k} \frac{\omega_i}{\|a_i\|^2} \nabla f_i(x). \end{aligned} \tag{39}$$

Thus, we summarize this idea as the following theorem.

Corollary 11. Given a loss function as defined in (37), the quantum algorithm described in Algorithm 1 can prepare a quantum state that encodes the result of K iterations of stochastic gradient descent of the loss function with query complexity $O(K)$ and gate complexity $O(K \log_2 n)$.

VII. SUMMARY

In this paper, we present a quantum algorithm for a linear system with nonsquare coefficient matrix. We show that the quantum version of multirow iterations possesses exponential speedups in problem size n and a faster convergence rate in the constraints m while keeping the logarithmic dependence on the error tolerance. There are still many open questions. For example, combining the row and column iteration shows a faster convergence rate. Still, the pure quantum version of this method does not exist because this poses new challenges for both state generation and circuit implementation. Is there a proper quantum data structure for this method? Moreover, for many iteration methods with different strategies, such as iteration with a small block, can we construct pure quantum version algorithms for these methods? Moreover, is there a fast and general quantum algorithm for all kinds of linear systems? What is the lower bound on the complexity of such an approach?

ACKNOWLEDGMENTS

This work was supported in part by National Natural Science Foundation of China Grants No. 62325210, No. 62272441, No. 12204489, and No. 62301531 and Strategic Priority Research Program of Chinese Academy of Sciences Grant No. XDB28000000.

APPENDIX A: DETAILS OF THE APPLICATION OF U_k

As shown in Sec. III A 3, applying the iteration matrix U_k requires one to apply the block encoding given in Eq. (19). Such block encoding achieves the linear combination of unitary

$$\sum_{i\in\tau_k}\left(\frac{\omega_{k,i}}{2}C_1^{(i,i)} - \frac{\omega_{k,i}}{2}C_{-1}^{(i,i)}\right) + \sum_{i\notin\tau_k}\left(\frac{r_{k,i}}{2}C_1^{(i,i)} - \frac{r_{k,i}}{2}C_1^{(i,i)}\right). \tag{A1}$$

We introduce an ancilla register to help apply the coefficients. Then, the combination can be treated as

$$\begin{aligned} \tilde{U}_k = &\sum_{j=0}^{m-1} |j\rangle\langle j| \otimes C_1^{(j,j)} + \sum_{\substack{j=m \\ j-m\in\tau_k}}^{2m-1} |j\rangle\langle j| \otimes C_{-1}^{(j-m,j-m)} \\ &+ \sum_{\substack{j=m \\ j-m\notin\tau_k}}^{2m-1} |j\rangle\langle j| \otimes C_1^{(j-m,j-m)} \end{aligned} \tag{A2}$$

with the coefficients applying on different indices j. We can apply the operator $\tilde{U}_k$ equivalently by an operator U_{eq}, as they achieve the same result on the basis state:

$$\tilde{U}_k|j\rangle|l\rangle = \begin{cases} |j\rangle|(2j-l) \mod m\rangle, & 0 \leqslant j \leqslant m-1 \\ |j\rangle|(2j-l) \mod m\rangle, & m \leqslant j \leqslant 2m-1 \\ -|j\rangle|(2j-l) \mod m\rangle, & m \leqslant j \leqslant 2m-1, l = (j \mod m), j-m \in \tau_k \end{cases}. \tag{A3}$$

We define the following function:

$$f(j,l)=\begin{cases}0, & 0\leqslant j\leqslant m-1\\ 0, & m\leqslant j\leqslant 2m-1\\ 1, & m\leqslant j\leqslant 2m-1,\\ & l=(j \mod m),(j \mod m)\in\tau_k\end{cases}. \tag{A4}$$

This function is associated with the flip on the corresponding state. To compute such classical function with a quantum circuit, we use the quantum comparator [33,34]. The comparator compares the natural numbers a and b in two registers and outputs the result c in the third register, if $a\leqslant b$, $c=0$; otherwise, $c=1$. The specific circuit of U_f is given as follows.

(1) For $(j \mod m)\in\tau_k$, prepare the initial state

$$\begin{aligned}&|j\rangle_{a_1}|l\rangle_{a_2}|0\rangle_{b_1}|0\rangle_{b_2}|0\rangle_{b_3}|0\rangle_{c_1}|0\rangle_{c_2}|0\rangle_{c_3}|-\rangle_{c_4}\\ \rightarrow&|j\rangle_{a_1}|l\rangle_{a_2}|m-1\rangle_{b_1}|(j-1) \mod m\rangle_{b_2}|j \mod m\rangle_{b_3}\\ &|0\rangle_{c_1}|0\rangle_{c_2}|0\rangle_{c_3}|-\rangle_{c_4}.\end{aligned}$$

For $(j \mod m)\notin\tau_k$, prepare the initial state

$$\begin{aligned}&|j\rangle_{a_1}|l\rangle_{a_2}|0\rangle_{b_1}|0\rangle_{b_2}|0\rangle_{b_3}|0\rangle_{c_1}|0\rangle_{c_2}|0\rangle_{c_3}|-\rangle_{c_4}\\ \rightarrow&|j\rangle_{a_1}|l\rangle_{a_2}|2m-1\rangle_{b_1}|(j-1) \mod m\rangle_{b_2}|j \mod m\rangle_{b_3}\\ &|0\rangle_{c_1}|0\rangle_{c_2}|0\rangle_{c_3}|-\rangle_{c_4}.\end{aligned}$$

(2) Perform the quantum comparator on registers $\{a_1,b_1,c_1\}$, $\{a_2,b_2,c_2\}$, and $\{a_3,b_3,c_3\}$ respectively.

(3) Perform the TOFFOLI gate on the registers on $\{c_1,c_2,c_4\}$, $\{c_1,c_3,c_4\}$, and $\{c_2,c_3,c_4\}$.

(4) Reverse the computation on registers $\{c_3,c_2,c_1,b_3,b_2,b_1\}$.

The states of registers c_1, c_2, and c_3 are set as $|1\rangle$, when $a_1>b_1$, $a_2>b_2$, and $a_2\leqslant b_3$ are satisfied respectively. The mapping on registers a_1, a_2, and c_4 is

$$U_f|j\rangle|l\rangle|-\rangle=(-1)^{f(j,l)}|j\rangle|l\rangle|-\rangle. \tag{A5}$$

Using a quantum modular adder, we can implement

$$U_{\rm adder}|j\rangle|l\rangle=|j\rangle|(2j-l) \mod m\rangle. \tag{A6}$$

Therefore, the equivalent process $U_{\rm eq}$ can be implemented by U_f and $U_{\rm adder}$. Then, the box in Eq. (35) is completed by

$$(I\otimes\tilde{V})(G\otimes I)\tilde{U}_k(G^\dagger\otimes I)(I\otimes\tilde{V}^\dagger) \tag{A7}$$

where we omit the subscript of I. If we attempt to treat the operators G and $\tilde{V}$ as state preparation operators instead of memory access operators, we should have the following observation.

Observation 12. In practice, the memory access operator V can be replaced by the state preparation operator. However, besides satisfying the assumption $V|i\rangle|0\rangle=|i\rangle|A_i\rangle$, the matrix associated with the state preparation operator should possess symmetry.

The proof is given in Appendix D.

However, this is not enough for the implementation of the operator U_k. We need to implement a controlled version of $\tilde{U}_k$ and the memory access operators G and $\tilde{V}$. Therefore, we introduce two extra qubits, and then we can prepare the following state:

$$\begin{aligned}&|00\rangle_{\rm anc}|{\rm ind}\rangle|{\rm work}\rangle\\ &-|01\rangle_{\rm anc}\tilde{V}\big(\langle 0|(G\otimes I)\tilde{U}_k(G^\dagger\otimes I)|0\rangle\big)\tilde{V}^\dagger|{\rm ind}\rangle|{\rm work}\rangle\\ &+|10\rangle_{\rm anc}\tilde{V}\big(\langle 0|(G\otimes I)\tilde{U}_k(G^\dagger\otimes I)|0\rangle\big)\tilde{V}^\dagger|{\rm ind}\rangle|{\rm work}\rangle\\ &-|11\rangle_{\rm anc}\tilde{V}\big(\langle 0|(G\otimes I)\tilde{U}_k(G^\dagger\otimes I)|0\rangle\big)\tilde{V}^\dagger|{\rm ind}\rangle|{\rm work}\rangle.\end{aligned} \tag{A8}$$

Introducing one more ancillary qubit and applying some Hadamard gates and CNOT gates, we can obtain the required states after applying the operator U_k. The circuit for the above state is shown in Fig. 3. This equals to a $(1,3,\epsilon)$ block encoding of the operator U_k. The circuit before the dotted box in Fig. 3 prepares the state in (A8). To simplify the notation, we use $|{\rm target}\rangle$ to represent the state $\tilde{V}(\langle 0|G\tilde{U}_kG^\dagger|0\rangle)\tilde{V}^\dagger|{\rm ind}\rangle|{\rm work}\rangle$ and omit the registers a and c. Then, the circuit in the dotted box fulfills the following mapping:

$$\begin{aligned}&|{\rm anc}\rangle|0\rangle(|00\rangle|{\rm ind}\rangle|{\rm work}\rangle-|01\rangle|{\rm target}\rangle+|10\rangle|{\rm target}\rangle-|11\rangle|{\rm target}\rangle)\\ \rightarrow&(I_2\otimes\{I-\tilde{V}[\langle 0|(G\otimes I)\tilde{U}_k(G^\dagger\otimes I)|0\rangle]\tilde{V}^\dagger\}+X\otimes\tilde{V}[\langle 0|(G\otimes I)\tilde{U}_k(G^\dagger\otimes I)|0\rangle]\tilde{V}^\dagger)\\ &|{\rm anc}\rangle|000\rangle|{\rm ind}\rangle|{\rm work}\rangle+|Gb\rangle.\end{aligned} \tag{A9}$$

Therefore, we fulfill the construction of the block encoding of operator U_k.

It should also be noted that the weights satisfy $\sum_{i\in\tau_k}\omega_{k,i}=1$. We suppose the operator G completes $G|0\rangle=\sum_{i\in\tau_k}\omega_{k,i}|i\rangle$, then apply the operators G and $\tilde{V}$, a multiqubit controlled NOT gate, which completes $I_2\otimes(I-\sum_{i\in\tau_k}|i\rangle\langle i|)+X\otimes\sum_{i\in\tau_k}|i\rangle\langle i|$ and $G^\dagger$ and $\tilde{V}^\dagger$. This can achieve the same result.

APPENDIX B: DETAILS OF THE WHOLE PROCESS

1. Implementation of the whole process

Figure 6 shows the circuit of an iteration step. It requires a storage process s and a reading process r. That is because at the end of any iteration step k, we obtain the state $|X^{k+1}\rangle$. However, at the beginning of any step k, we are required to prepare $\beta_k|0\rangle\sum_{i\in\tau_k}|i\rangle|0\rangle+|1\rangle\sum_{i\in\tau_k}\gamma_{k,i}|i\rangle|0\rangle$ and apply $|X^k\rangle$ and $|A_i\rangle$. Therefore, we should store the result from the last

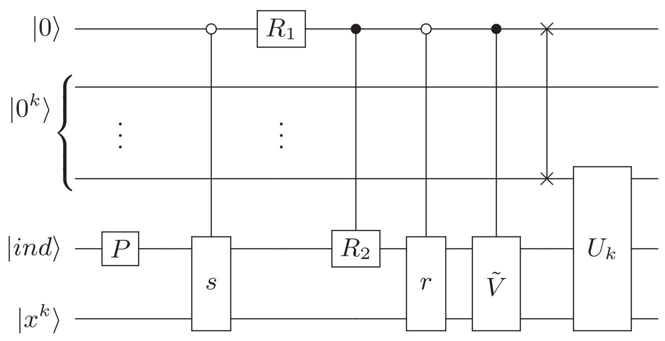

FIG. 6. Implementation of an iteration step with the help of QRAM, where s represents the process for storage and r for the reading process. The operators R_1 and R_2 are the rotation operators.

iteration step. This can be avoided if we apply the rotation at the beginning of the whole process.

a. Preparing at each step

To generate the state $|Y^k\rangle$, we can first prepare $\beta_k|0\rangle\sum_{i\in\tau_k}s_{k,i}|i\rangle|0\rangle+|1\rangle\sum_{i\in\tau_k}\gamma_{k,i}s_{k,i}|i\rangle|0\rangle$ using a set of rotation operators. Then, perform a controlled operator to obtain $|X^k\rangle$ and $|i\rangle|A_i\rangle$ on the index and work register with the help of QRAM (see Fig. 6). At the end of each iteration, we need to store the current result, reset the working register, and then repeat the above process. This would require extra access to the QRAM compared to the other idea, as we need to store the result after each iteration.

b. Preparing before the iteration

The other idea is to prepare $\theta_1|0\cdots0\rangle\sum_{i\in\tau_1}|i\rangle|x^1\rangle+\theta_2|0\cdots01\rangle\sum_{i\in\tau_1}|i\rangle|A_i\rangle+\theta_3|0\cdots010\rangle\sum_{i\in\tau_2}|i\rangle|A_i\rangle+\cdots$ before starting the iteration procedure. The required rotation gates for θ, which depends on the choice of weights, and each set τ_k that depends on the choice of rows, can be obtained through a preprocessing procedure. This will require that the memory access procedure become multiqubit controlled and the iteration operator become one-qubit controlled.

2. Resource analysis for the whole process

In the algorithm given in Algorithm 1, the number of quantum gates used consists of four components: the rotation gates for β_k and $\gamma_{k,i}$, gates for the iteration matrix U_k, gates for the operator P, and gates for U_{add}.

For each step, the rotation requires $O(\log_2 m)$ elementary gates. The application of the quantum comparators and the quantum modular adder needs $O(\log_2 m)$ elementary gates and $O(\log_2 m)$ ancilla qubits. The operator $\langle 0|(G\otimes I)\tilde{U}_k(G^\dagger\otimes I)|0\rangle$ can be applied with a probability of $\sum_{i\in\tau_k}\omega_{k,i}=t_k$, then we can apply amplitude amplification $O(\frac{1}{\sqrt{t_k}})$ times to achieve a high probability. Therefore, the iteration matrix U_k needs $O(\frac{1}{\sqrt{t_k}})$ queries to G and $O(1)$ queries to $\tilde{V}$, $O(\frac{1}{\sqrt{t_k}}\log_2 m)$ elementary gates, and $O(\log_2 m)$ ancilla qubits to complete. Through appropriate selection of weights $\sum_{i\in\tau_k}\omega_{k,i}=t_k$, $\frac{1}{\sqrt{t_k}}$ is a constant and can be neglected. We omit the resource for the operator P, since it depends on specific situations, but in most cases, several SWAP gates and controlled NOT gates are enough.

We also omit the resource for the operator U_{add}, since it also depends on specific situations. If we use $\log_2 m$ Hadamard gates to apply U_{add}, then we will be required to apply amplitude amplification $O(\sqrt{m/\sum_{i\in\tau_K}s_{K,i}})$ times, where K is the number of the total iteration steps, to achieve the result with a high probability. But, if we can design an exact operator that achieves the same effect, then the resource required will be quite different.

For an iteration process with K iteration steps, we can obtain the final result with a probability of $\frac{1}{V_K^2}$. In order to obtain the result with a high probability, we need to perform amplitude amplification $O(\sqrt{V_K^2})$ times. It should be noted that $V_K=\sqrt{1+\sum_{k=1}^K\sum_{i\in\tau_k}b_i}$ is not large, because the parameter b_i here is rescaled as mentioned in Sec. III A 2 and the number of iteration steps is limited. In summary, K iteration steps require $O(K\sqrt{\frac{V_K^2}{t_K}})$ queries to G and $O(K\sqrt{V_K^2})$ queries to $\tilde{V}$, $O(K\sqrt{\frac{V_K^2}{t_K}}\log_2 m)$ elementary gates, and $O(\log_2 m+K)$ ancilla qubits (each iteration requires one extra ancilla qubit). Combining with the complexity shown in Definition 5, the complexity of the quantum multirow iteration algorithm is $O(K\sqrt{\frac{V_K^2}{t_K}}\log_2 m)$.

APPENDIX C: PROOF OF THE CONVERGENCE RATE

The analysis for the convergence rate in the quantum setting is quite similar to the one in the classical setting [24]. To analyze the convergence rate, we begin with the error update at each iteration. The error is defined as $e^k=x^k-x^*$, where x^* is the solution or least-square solution. We suppose the residual is r^* and $Ax^*+r^*=b$. Then, using $A_ie^k-r_i^*=A_ix^k-b_i$, we arrive at the error update:

$$e^{k+1}=e^k-\sum_{i\in\tau_k}\omega_{k,i}\frac{A_ie^k-r_i^*}{\|A_i\|^2}A_i^T \tag{C1}$$

where we suppose $\|A_i\|=1$. We do not omit it for the completeness of the proof.

Define the weighted sampling matrix:

$$M_k:=\sum_{i\in\tau_k}\omega_{k,i}\frac{I_i^TI_i}{\|A_i\|^2}. \tag{C2}$$

Then, the error update can be rewritten as

$$e^{k+1}=(I-A^TM_kA)e^k+A^TM_kr^*. \tag{C3}$$

To evaluate the error update, we give the following lemma.

Lemma 13 (see [24]). Given D, P, and W as defined in Definition 2, then we have

$$\mathbb{E}[M_k]=PDW^{-2} \tag{C4}$$

and

$$\mathbb{E}\big[M_k^TAA^TM_k\big]=\frac{1}{q}PW^2D^{-2}+\left(1-\frac{1}{q}\right)PWD^{-2}AA^TPWD^{-2}. \tag{C5}$$

The proof for this lemma is given in Appendix E. Since M_k is a sample average, as the number of samples goes to infinity, we should have

$$M_k \to PWD^{-2}. \tag{C6}$$

Therefore, as the number of samples goes to infinity, the error update approaches the deterministic update:

$$e^{k+1} = (I - A^T PWD^{-2}A)e^k + A^T PWD^{-2}r^*. \tag{C7}$$

Since we want the error to converge to zero, we should require that this limiting error update has the zero vector, which is

$$A^T PWD^{-2}r^* = 0 \tag{C8}$$

for any least-squares residual r^*. This holds when the following equation is satisfied:

$$PWD^{-2} = \alpha_A I \tag{C9}$$

for $0 < \alpha_A \leqslant 1$ ($\alpha_A > 0$ is the same as the classical setting and $\alpha_A \leqslant 1$ is unique in the quantum setting). The squared error norm is

$$\begin{aligned}\|e^{k+1}\|^2 &= \|(I - A^T M_k A)e^k + A^T M_k r^*\|^2\\ &= \|(I - A^T M_k A)e^k\|^2 + 2\langle (I - A^T M_k A)e^k, A^T M_k r^*\rangle\\ &\quad + \|A^T M_k r^*\|^2.\end{aligned} \tag{C10}$$

Taking expectations, we can get

$$\begin{aligned}\mathbb{E}[\|e^{k+1}\|^2] &= \mathbb{E}[\|(I - A^T M_k A)e^k\|^2] + \mathbb{E}[\|A^T M_k r^*\|^2]\\ &\quad + 2\mathbb{E}[\langle (I - A^T M_k A)e^k, A^T M_k r^*\rangle].\end{aligned} \tag{C11}$$

Using Lemma 13, we can simplify the first term:

$$\begin{aligned}\mathbb{E}[\|(I - A^T M_k A)e^k\|^2] &= \mathbb{E}[\langle e^k, (I - A^T M_k A)^T (I - A^T M_k A)e^k\rangle]\\ &= \left\langle e^k, \left(I - 2A^T\mathbb{E}[M_k]A + A^T\mathbb{E}\left[M_k^T AA^T M_k\right]A\right)e^k\right\rangle\\ &= \left\langle e^k, \left[I - 2\alpha_A\frac{A^TA}{\|A\|_F^2} + \frac{\alpha_A}{q}\frac{A^TWA}{\|A\|_F^2} + \alpha_A^2\left(1 - \frac{1}{q}\right)\left(\frac{A^TA}{\|A\|_F^2}\right)^2\right]e^k\right\rangle\\ &= \left\langle e^k, \left[\left(I - \alpha_A\frac{A^TA}{\|A\|_F^2}\right)^2 + \frac{A^T}{\|A\|_F^2}\left(\frac{\alpha_A}{q}W - \frac{\alpha_A^2}{q}\frac{AA^T}{\|A\|_F^2}\right)\frac{A}{\|A\|_F}\right]e^k\right\rangle.\end{aligned} \tag{C12}$$

For the second term, we can get

$$\mathbb{E}[\|A^T M_k r^*\|^2] = \left\langle r^*, \mathbb{E}\left[M_k^T AA^T M_k\right]r^*\right\rangle = \frac{\alpha_A}{q}\frac{\|r^*\|_W^2}{\|A\|_F^2}. \tag{C13}$$

Similarly, for the third term, we can get

$$2\mathbb{E}[\langle A^T M_k Ae^k, A^T M_k r^*\rangle] = \frac{2\alpha_A}{q\|A\|_F^2}\langle Ae^k, Wr^*\rangle. \tag{C14}$$

Combining three terms together, we have

$$\begin{aligned}\mathbb{E}[\|e^{k+1}\|^2] &= \left\langle e^k, \left(I - \alpha_A\frac{A^TA}{\|A\|_F^2}\right)^2 e^k\right\rangle\\ &\quad + \left\langle e^k, \frac{A^T}{\|A\|_F^2}\left(\frac{\alpha_A}{q}W - \frac{\alpha_A^2}{q}\frac{AA^T}{\|A\|_F^2}\right)\frac{A}{\|A\|_F}e^k\right\rangle\\ &\quad - \frac{2\alpha_A}{q\|A\|_F^2}\langle Ae^k, Wr^*\rangle + \frac{\alpha_A}{q}\frac{\|r^*\|_W^2}{\|A\|_F^2}\\ &= \left\langle e^k, \left[\left(I - \alpha_A\frac{A^TA}{\|A\|_F^2}\right)^2 - \frac{\alpha_A^2}{q}\left(\frac{A^TA}{\|A\|_F^2}\right)^2\right]e^k\right\rangle\\ &\quad + \frac{\alpha_A}{q}\frac{\|r^*\|_W^2}{\|A\|_F^2}\\ &\leqslant \sigma_{\max}\left[\left(I - \alpha_A\frac{A^TA}{\|A\|_F^2}\right)^2 - \frac{\alpha_A^2}{q}\left(\frac{A^TA}{\|A\|_F^2}\right)^2\right]\|e^k\|^2\\ &\quad + \frac{\alpha_A}{q}\frac{\|r^*\|_W^2}{\|A\|_F^2}.\end{aligned} \tag{C15}$$

This completes the proof.

APPENDIX D: PROOF OF OBSERVATION 12

Without loss of generality, we assume that there exists a state preparation operator $|V\rangle$ satisfying

$$V|0\rangle = |a\rangle \tag{D1}$$

where $|a\rangle = a_1|0\rangle + a_2|1\rangle$; without loss of generality, we assume that a_1 and a_2 are real. We anticipate utilizing this state preparation operator to achieve

$$\begin{aligned}U &= (I_2 \otimes V)(I_2 \otimes (I - |0\rangle\langle 0|) + X \otimes |0\rangle\langle 0|)(I_2 \otimes V^\dagger)\\ &= \begin{bmatrix} I - |a\rangle\langle a| & |a\rangle\langle a| \\ |a\rangle\langle a| & I - |a\rangle\langle a| \end{bmatrix}.\end{aligned} \tag{D2}$$

We could represent $[I_2 \otimes (I - |0\rangle\langle 0|) + X \otimes |0\rangle\langle 0|]$ in a matrix format easily:

$$(I_2 \otimes (I - |0\rangle\langle 0|) + X \otimes |0\rangle\langle 0|) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{D3}$$

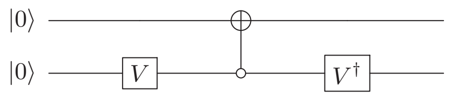

FIG. 7. Implementation of the operator U.

Then, we can obtain the circuit (Fig. 7) to perform the operator U. We choose the operator V as

$$V_1=\begin{bmatrix} a_1 & a_2\\ -a_2 & a_1\end{bmatrix} \tag{D4}$$

and

$$V_2=\begin{bmatrix} a_1 & a_2\\ a_2 & -a_1\end{bmatrix}. \tag{D5}$$

Both V_1 and V_2 are unitary operators and satisfy $V|0\rangle=|a\rangle$. By a simple calculation, we obtain

$$(I_2\otimes V_1)(I_2\otimes(I-|0\rangle\langle 0|)+X\otimes|0\rangle\langle 0|)(I_2\otimes V_1^\dagger)$$
$$=\begin{bmatrix} -a_2^2 & a_1a_2 & a_1^2 & a_1a_2\\ -a_1a_2 & a_1^2 & -a_1a_2 & -a_2^2\\ a_1^2 & a_1a_2 & -a_2^2 & a_1a_2\\ -a_1a_2 & -a_2^2 & -a_1a_2 & a_1^2\end{bmatrix} \tag{D6}$$

and

$$(I_2\otimes V_2)(I_2\otimes(I-|0\rangle\langle 0|)+X\otimes|0\rangle\langle 0|)(I_2\otimes V_2^\dagger)$$
$$=\begin{bmatrix} a_2^2 & -a_1a_2 & a_1^2 & a_1a_2\\ -a_1a_2 & a_1^2 & a_1a_2 & a_2^2\\ a_1^2 & a_1a_2 & a_2^2 & -a_1a_2\\ a_1a_2 & a_2^2 & -a_1a_2 & a_1^2\end{bmatrix}. \tag{D7}$$

Given an input state $|0\rangle|x\rangle$, $|x\rangle=|0\rangle$ (without loss of generality). Applying the operator U on the input state, we will obtain

$$\begin{aligned} U|0\rangle|x\rangle &= |0\rangle|x\rangle-|0\rangle\langle a|x\rangle|a\rangle\\ &=|0\rangle\big(1-a_1^2\big)|0\rangle-|0\rangle a_1a_2|1\rangle\\ &=|0\rangle a_2^2|0\rangle-|0\rangle a_1a_2|1\rangle. \end{aligned} \tag{D8}$$

This result can only be obtained if we choose $V=V_2$. And the matrix in (18) equals the matrix corresponding to the choice of V_2.

Extending this statement to the more general case, the definition of U demonstrates the symmetry of U. Therefore, in the real number case, the operator U should satisfy $U^T=U$, which indicates that $V=V^\dagger$. This assertion extends to the imaginary number case as $V=(V^\dagger)^*$.

When applying $V|0\rangle\langle 0|V^\dagger$, a similar situation arises. After implementing $(|0\rangle\langle 0|)V$ on an arbitrary state $|\psi\rangle$, the resulting state is $|\phi\rangle=(|0\rangle\langle 0|)V|\psi\rangle$. This causes the operator $V^\dagger$ to act on the state as $V^\dagger|\phi\rangle$ instead of $\langle\phi|V^\dagger$. Therefore, it is necessary for the operator V to be symmetric.

APPENDIX E: PROOF OF LEMMA 13

The expectation of M_k is as follows:

$$\begin{aligned} \mathbb{E}[M_k] &= \mathbb{E}\left[\sum_{i\in\tau_k}\omega_{k,i}\frac{I}{\|A_i\|^2}\right]=\mathbb{E}\left[\sum_{i\in\tau_k}\frac{\omega_i}{q}\frac{I}{\|A_i\|^2}\right]\\ &=\mathbb{E}\left[\omega_i\frac{I}{\|A_i\|^2}\right]=\sum_{i=0}^{m-1}p_i\omega_i\frac{I}{\|A_i\|^2}\\ &=PWD^{-2}. \end{aligned} \tag{E1}$$

Similarly, we can compute

$$\begin{aligned} &\mathbb{E}\big[M_k^TAA^TM_k\big]\\ &=\mathbb{E}\left[\left(\sum_{i\in\tau_k}\omega_{k,i}\frac{I_i^TA_i}{\|A_i\|^2}\right)\left(\sum_{j\in\tau_k}\omega_{k,j}\frac{I_j^TA_j}{\|A_j\|^2}\right)\right]\\ &=\frac{1}{q}\mathbb{E}\left[\left(\omega_i\frac{I_i^TA_i}{\|A_i\|^2}\right)\left(\omega_i\frac{A_i^TI_i}{\|A_i\|^2}\right)\right]\\ &\quad+\left(1-\frac{1}{q}\right)\mathbb{E}\left[\omega_i\frac{I_i^TA_i}{\|A_i\|^2}\right]\mathbb{E}\left[\omega_j\frac{A_j^TI_j}{\|A_j\|^2}\right]\\ &=\frac{1}{q}\mathbb{E}\left[\omega_i^2\frac{I_i^TI_i}{\|A_i\|^2}\right]+\left(1-\frac{1}{q}\right)PWD^{-2}AA^TPWD^{-2}\\ &=\frac{1}{q}PW^2D^{-2}+\left(1-\frac{1}{q}\right)PWD^{-2}AA^TPWD^{-2}. \end{aligned} \tag{E2}$$

[1] K. Kubo, Y. O. Nakagawa, S. Endo, and S. Nagayama, Variational quantum simulations of stochastic differential equations, Phys. Rev. A **103**, 052425 (2021).

[2] D. An, N. Linden, J.-P. Liu, A. Montanaro, C. Shao, and J. Wang, Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance, Quantum **5**, 481 (2021).

[3] N. Lambert, Y.-N. Chen, Y.-C. Cheng, C.-M. Li, G.-Y. Chen, and F. Nori, Quantum biology, Nat. Phys. **9**, 10 (2013).

[4] Z.-Y. Chen, C. Xue, S.-M. Chen, B.-H. Lu, Y.-C. Wu, J.-C. Ding, S.-H. Huang, and G.-P. Guo, Quantum finite volume method for computational fluid dynamics with classical input and output, arXiv:2102.03557.

[5] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature (London) **549**, 195 (2017).

[6] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. **103**, 150502 (2009).

[7] 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).

[8] L. Wossnig, Z. Zhao, and A. Prakash, Quantum linear system algorithm for dense matrices, Phys. Rev. Lett. **120**, 050502 (2018).

[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] L. Lin and Y. Tong, Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems, Quantum **4**, 361 (2020).

[11] D. An and L. Lin, Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm, ACM Trans. Quantum Comput. **3**, 1 (2022).

[12] P. C. S. Costa, D. An, Y. R. Sanders, Y. Su, R. Babbush, and D. W. Berry, Optimal scaling quantum linear-systems solver via discrete adiabatic theorem, PRX Quantum **3**, 040303 (2022).

[13] A. H. Andersen, Algebraic reconstruction in CT from limited views, IEEE Trans. Med. Imaging **8**, 50 (1989).

[14] J. A. K. Suykens and J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. **9**, 293 (1999).

[15] R. Gordon, R. Bender, and G. T. Herman, Algebraic reconstruction techniques (art) for three-dimensional electron microscopy and x-ray photography, J. Theor. Biol. **29**, 471 (1970).

[16] W. Gander, Algorithms for the QR decomposition, Res. Rep **80**, 1251 (1980).

[17] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, Singular value decomposition and principal component analysis, *A Practical Approach to Microarray Data Analysis* (Springer, New York, 2003), pp. 91–109.

[18] T. Strohmer and R. Vershynin, A randomized Kaczmarz algorithm with exponential convergence, J. Fourier Anal. Appl. **15**, 262 (2009).

[19] B. Wu, M. Ray, L. Zhao, X. Sun, and P. Rebentrost, Quantum-classical algorithms for skewed linear systems with an optimized Hadamard test, Phys. Rev. A **103**, 042422 (2021).

[20] I. Kerenidis and A. Prakash, Quantum gradient descent for linear systems and least squares, Phys. Rev. A **101**, 022316 (2020).

[21] C. Shao and H. Xiang, Row and column iteration methods to solve linear systems on a quantum computer, Phys. Rev. A **101**, 022322 (2020).

[22] S. M. Kaczmarz, Angenäherte auflösung von systemen linearer gleichungen, Classe des Sciences Mathématiques et Naturelles: Série A, Sciences Mathématiques **35**, 355 (1937).

[23] X. Lian, Y. Huang, Y. Li, and J. Liu, Asynchronous parallel stochastic gradient for nonconvex optimization, Proc. Adv. Neural Inf. Process. Syst. **2**, 2737 (2015).

[24] J. D. Moorman, T. K. Tu, D. Molitor, and D. Needell, Randomized Kaczmarz with averaging, BIT Numer. Math. **61**, 337 (2021).

[25] D. Needell, Randomized Kaczmarz solver for noisy linear systems, BIT Numer. Math. **50**, 395 (2010).

[26] J. Liu and S. Wright, An accelerated randomized Kaczmarz algorithm, Math. Comp. **85**, 153 (2016).

[27] I. Necoara, Faster randomized block Kaczmarz algorithms, SIAM J. Matrix Anal. Appl. **40**, 1425 (2019).

[28] A.-Q. Xiao, J.-F. Yin, and N. Zheng, On fast greedy block Kaczmarz methods for solving large consistent linear systems, Comput. Appl. Math. **42**, 119 (2023).

[29] G. H. Low and I. L. Chuang, Optimal Hamiltonian simulation by quantum signal processing, Phys. Rev. Lett. **118**, 010501 (2017).

[30] S. Chakraborty, A. Gilyén, and S. Jeffery, The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation, in *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming* (ICALP 2019), pp. 33:1–33:14.

[31] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (ACM, 2019), pp. 193–204.

[32] L.-C. Wan, C.-H. Yu, S.-J. Pan, S.-J. Qin, F. Gao, and Q.-Y. Wen, Block-encoding-based quantum algorithm for linear systems with displacement structures, Phys. Rev. A **104**, 062414 (2021).

[33] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, A new quantum ripple-carry addition circuit, arXiv:quant-ph/0410184.

[34] H.-S. Li, P. Fan, H. Xia, H. Peng, and G.-L. Long, Efficient quantum arithmetic operation circuits for quantum image processing, Science China Physics, Sci. China Phys. Mech. Astron. **63**, 280311 (2020).

[35] V. Giovannetti, S. Lloyd, and L. Maccone, Quantum random access memory, Phys. Rev. Lett. **100**, 160501 (2008).

[36] D. W. Berry, A. M. Childs, and R. Kothari, Hamiltonian simulation with nearly optimal dependence on all parameters, in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science* (IEEE, New York, 2015), pp. 792–809.

[37] D. Needell, N. Srebro, and R. Ward, Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm, Math. Program. **155**, 549 (2016).