Row and column iteration methods to solve linear systems on a quantum computer

Changpeng Shao*
School of Mathematics, University of Bristol, Bristol BS8 1UG, United Kingdom

Hua Xiang†
School of Mathematics and Statistics, Wuhan University, Wuhan 430072, China
and Hubei Key Laboratory of Computational Science, Wuhan University, Wuhan 430072, China

(Received 28 May 2019; revised manuscript received 16 January 2020; accepted 29 January 2020; published 19 February 2020)

We consider the quantum implementations of two classical iterative solvers for a system of linear equations, including the row (Kaczmarz) method which utilizes a row of the coefficient matrix in each iteration step, and the column (coordinate descent) method which uses a column instead. These two methods are widely applied in big data science due to their simple iteration schemes. We propose fast quantum algorithms for these two approaches by constructing efficient unitary operators in each iteration step based on the block-encoding technique. The construction is based on the unitaries to prepare the quantum states of rows or columns of the coefficient matrix. If the quantum states are efficiently prepared, for example, by QRAM, then the quantum iterative methods achieve an exponential speedup in the problem size n over the classical versions. Meanwhile the dependence on the number of steps is linear, which is the same as the classical counterparts. The complexity of the quantum iterative linear solvers is $O(\kappa_s^2(\log n)(\log \frac{1}{\epsilon}))$, where κ_s is the scaled condition number, and ϵ is the error. The quantum iterative linear solvers do not depend on Hamiltonian simulations and quantum phase estimation, and also work for the dense case.

DOI: 10.1103/PhysRevA.101.022322

I. INTRODUCTION

A. Motivations

The classical solvers for a linear system of equations $A\mathbf{x} = \mathbf{b}$ are generally categorized into two types: direct methods and iterative methods. The latter is usually more practical in the realm of a large-scale system of equations. Among the iterative methods, the row and column iterative methods are popular due to the simplicity and efficiency. The row iteration method is the famous Kaczmarz method, first discovered by Kaczmarz in 1937 [1], and rediscovered in the field of image reconstruction by Gordon, Bender, and Herman in 1970 [2] under the appellation algebraic reconstruction technique (ART). The column iterative method is also called the coordinate descent method [3]. As the name suggests, this method uses a column of coefficient matrix in each iterative, while the row (Kaczmarz) method utilizes a row instead. These two methods seek to solve different problems: the column iteration method converges to a least-squares solution generally; the row iteration method calculates a minimum-norm solution for a consistent system of equations and exhibits cyclic convergence for an inconsistent problem [4]. They can be generalized to many variants (see [5,6] and the references therein). The advantage of these two methods lies in the fact that at each iteration they only need access to an individual row (or column) rather than the entire coefficient matrix. Due to the simplicity, they have numerous applications in the fields ranging from computer tomography to image processing and digital signal processing, especially the big data science [7,8].

Suppose that the size of the problem is n and the number of required iterations is T, then the complexity of the classical iteration algorithm is usually polynomial in n and linear in T. The quantum iterative methods have been extensively investigated before (e.g., see [9–11]). Due to the quantum no-cloning theorem, an iteration algorithm is usually not easy to implement on a quantum computer.

For instance, in [9], Rebentrost *et al.* proposed a quantum gradient and Newton's method to solve polynomial optimization. Compared to the classical gradient or Newton's method, this quantum algorithm achieves exponential speedup in n. However, the complexity depends exponentially on T. The main reason is as follows, taking Newton's method as an example. One critical step of classical Newton's method is to solve a Newton linear system. The coefficient matrix (i.e., the Hessian matrix) H of the linear system depends on $\mathbf{x}_k$, the computed result in step k. In a quantum computer, we only have the quantum state of $\mathbf{x}_k$, which is an unknown quantum state. To solve the linear system by Harrow-Hassidim-Lloyd (HHL) algorithm, for instance, one critical step is the Hamiltonian simulation of H. In [9], a similar idea to the quantum principal component analysis [12] is used to compute the Hamiltonian simulation of H by viewing it as an unknown density matrix. Together with quantum phase estimation, the Newton linear system can be solved efficiently. However, the Hamiltonian simulation of an unknown density operator requires $O(t^2/\epsilon)$ copies of this density operator, where t is

*changpeng.shao@bristol.ac.uk
†Corresponding author: hxiang@whu.edu.cn

the evolution time and ϵ is the precision. Since H depends on $|\mathbf{x}_k\rangle$, we need to prepare at least $O(t^2/\epsilon)$ copies of $|\mathbf{x}_k\rangle$. In other words, with at least $O(t^2/\epsilon)$ copies of $|\mathbf{x}_k\rangle$, we can prepare $|\mathbf{x}_{k+1}\rangle$. As a result, we need exponential copies of the initial state to prepare $|\mathbf{x}_T\rangle$.

In [10], Kerenidis and Prakash considered a quantum version of the interior-point method to solve semidefinite programming and linear programming. One critical step of the classical interior-point method is also to solve a linear system. Different from the idea used above, to reduce the exponential dependence on T, at each each step, they compute a unit vector $\widetilde{\mathbf{x}}_k$ whose quantum state is an approximation of $|\mathbf{x}_k\rangle$ by quantum tomography at the cost $O(n/\delta^2)$, where δ is the error. By doing so $|\mathbf{x}_k\rangle$ becomes a known quantum state as we can prepare it from $\widetilde{\mathbf{x}}_k$. Consequently, it can be prepared many times for the next iteration. Tomography is one source of complexity, so the complexity of their quantum algorithm is at least linear in n. The dependence on T is polynomial due to the accumulated errors caused in quantum tomography in each step.

The techniques used in the above two quantum iteration algorithms are currently standard. As we can see, using these techniques the quantum iterative algorithms [9,10] cannot outperform classical iteration algorithms both in n and T. The quantum iteration algorithm is a simulation of the classical iteration algorithm on a quantum computer, so it seems especially hard to achieve a speedup in T. However, we still expect the quantum iteration algorithm to achieve high speedup in n and have a reasonable dependence on T meanwhile. Therefore, an ideal simulation of the iteration algorithm on a quantum computer should have the complexity polynomial-logarithm in n and linear or polynomial in T. In this paper, we will give such implementations to the row (Kaczmarz) method and the column (coordinate descent) method based on the idea of block-encoding.

B. Iterative schemes

We review these two iteration schemes in the following. Assume that A is an n-by-n matrix. For any $1 \leqslant i \leqslant n$, denote the ith row of A as $\mathbf{a}_i^T$, and the ith component of $\mathbf{b}$ as b_i. Let $\mathbf{x}_0$ be an arbitrary initial approximation to the solution of $A\mathbf{x} = \mathbf{b}$. For $k \geqslant 0$, randomly choose an $i_k \in \{1, \ldots, n\}$ with probability proportional to $\|\mathbf{a}_{i_k}\|^2$, the randomized Kaczmarz iteration updates the solution $\mathbf{x}_k$ as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\frac{\mathbf{a}_{i_k}^T \mathbf{x}_k}{\|\mathbf{a}_{i_k}\|} - \frac{b_{i_k}}{\|\mathbf{a}_{i_k}\|} \right) \frac{\mathbf{a}_{i_k}}{\|\mathbf{a}_{i_k}\|}, \tag{1}$$

which is equivalent to the Gauss-Seidel method on $AA^T\mathbf{y} = \mathbf{b}$, where $A^T\mathbf{y} = \mathbf{x}$. It can be also derived from the minimization of a quadratic form $\psi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T AA^T\mathbf{y} - \mathbf{y}^T\mathbf{b}$ along the i_kth direction. Geometrically, $\mathbf{x}_{k+1}$ is the orthogonal projection of $\mathbf{x}_k$ onto the hyperplane $\mathbf{a}_{i_k}^T\mathbf{x} = b_{i_k}$ (see Fig. 1). In each iteration step, only one row of the coefficient matrix is needed. It is also called the row-action method, and another names, such as component-solution method, cyclic projection or successive projection, are used on certain occasions.

Correspondingly, we have the column-action method. Let $\mathbf{c}_j$ be the jth column of A, and $\mathbf{e}_j$ the jth column of the unit matrix. The column-action method reads

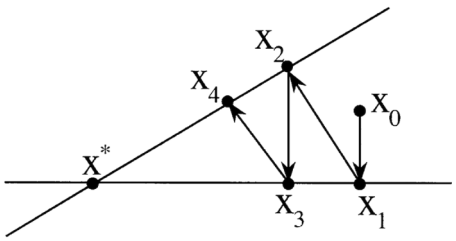

FIG. 1. Illustrative example of Kaczmarz iteration in dimension 2.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\mathbf{c}_{j_k}^T(\mathbf{b} - A\mathbf{x}_k)}{\|\mathbf{c}_{j_k}\|^2}\mathbf{e}_{j_k}, \tag{2}$$

where j_k is a random number from $\{1, \ldots, n\}$ with probability proportional to $\|\mathbf{c}_{j_k}\|^2$. It is equivalent to the randomized Gauss-Seidel method on $A^TA\mathbf{x} = A^T\mathbf{b}$. In fact, in each iteration step the approximate solution is obtained by minimizing a quadratic function $\varphi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$ (or $\|\mathbf{b} - A\mathbf{x}\|^2$) in one coordinate direction, then it is also called the coordinate descent method. Define $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$. This column iteration method can be re-expressed as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\mathbf{c}_{j_k}^T\mathbf{r}_k}{\|\mathbf{c}_{j_k}\|^2}\mathbf{e}_{j_k}, \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \frac{\mathbf{c}_{j_k}\mathbf{c}_{j_k}^T}{\|\mathbf{c}_{j_k}\|^2}\mathbf{r}_k. \tag{3}$$

The convergence of Kaczmarz method (1) was studied in [13]. Let $\mathbf{x}_{\text{true}}$ be the exact solution of the linear system $A\mathbf{x} = \mathbf{b}$, and $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}_{\text{true}}$ be the iteration error in step k. It results in a convergence with [13, Theorem 2]

$$\mathbb{E}[\|\mathbf{e}_k\|^2] \leqslant \left(1 - \kappa_s^{-2}\right)^k \|\mathbf{e}_0\|^2, \tag{4}$$

where $\kappa_s = \|A\|_F\|A^{-1}\|$ is known as the scaled condition number, $\|\cdot\|_F$, $\|\cdot\|$ are the Frobenius and spectral norms, respectively. The usual condition number is defined as $\kappa = \|A\|\|A^{-1}\|$, thus $\kappa \leqslant \kappa_s \leqslant \sqrt{n}\kappa$. The convergence of the column (coordinate descent) method (3) was considered in [3]. By Theorem 3.6 of [3]

$$\mathbb{E}\left[\|\mathbf{e}_k\|_{A^TA}^2\right] \leqslant \left(1 - \kappa_s^{-2}\right)^k \|\mathbf{e}_0\|_{A^TA}^2. \tag{5}$$

Suppose the relative error is ϵ, then by Eqs. (4) and (5), the number of required iterative steps is $\kappa_s^2 \log\frac{1}{\epsilon}$.

C. Main results

The main idea of our implementations of row and column iterations on a quantum computer is to use block-encoding technique [14] to implement the procedures (1), (3) by unitary operators. These unitary operators are explicitly constructed. Moreover, they can help to reduce the exponential dependence on T to linear by overcoming the calculations of the inner product of quantum states, which usually need many copies of the states in each iteration step.

The unitaries that we construct to implement the row and column iterations dominate the total cost of the quantum algorithms. However, they highly depend on the unitaries to prepare the quantum states of the rows and columns of the coefficient matrix. Thus, we need to assume that these states can be prepared efficiently. QRAM [15] is one general method to prepare quantum states. But it needs a tree data structure

TABLE I. The complexity comparison of classical and quantum Kaczmarz and column iteration methods.

Algorithm	Cost
Classical	$O(Tn) = O\big(\kappa_s^2 n \log \frac{1}{\epsilon}\big)$
Quantum	$O(T \log n) = O\big(\kappa_s^2 (\log n) \log \frac{1}{\epsilon}\big)$

which is often difficult to obtain [16]. In some other special cases [17], such as sparse or uniform, we can prepare the states efficiently without QRAM. For simplicity, we will make the assumption on the availability of QRAM in this paper. As a result, we will show that the row (Kaczmarz) method and the column (coordinate descent) method can be implemented on a quantum computer in time $O(T \log n)$. In comparison, the corresponding classical iteration methods have complexity $O(nT)$. Thus our quantum algorithms achieve an exponential speedup in the dimension n, while keeping the same dependence on the number of steps T as classical algorithms.

Table I summarizes the difference of the classical and quantum iterative algorithms obtained in this paper. In the table we choose $T = \kappa_s^2 \log \frac{1}{\epsilon}$.

Table II compares the efficiency of previous quantum linear solvers and ours. In Table III, we compare the different techniques used in previous quantum linear solvers and ours. The HHL and the Childs-Kothari-Somma (CKS) [19] algorithms need the matrix A to be sparse, while the other three have no such restriction. The Wossnig-Zhao-Prakash (WZP) [20] algorithm and ours depend on the QRAM. But there is still a little difference on the dependence. WZP's algorithm needs a unitary operator that can prepare all the quantum states of the rows of A in parallel. More precisely, the unitary achieves $|i, 0\rangle \mapsto |i, \mathbf{a}_i\rangle$ for all i. This is a direct application of the QRAM data structure. However, in our algorithms, we do not need such a strong assumption. Instead, for each row or column, we need a unitary operator to prepare its quantum state. The unitaries of different rows or columns can be different. So QRAM is not a necessary assumption, and our algorithms work when we can prepare the quantum states of the rows or columns. The Chakraborty-Gilyén-Jeffery (CGJ) [14] algorithm is an application of block-encoding. With a QRAM data structure, we can construct a block-encoding of A. But the block-encoding can be constructed in many ways.

Throughout this paper, we use the following notation.

(i). A quantum state of the form $\sum_{i=0}^{2^k-1} \alpha_i |i\rangle|\psi_i\rangle$ will be simply denoted by $\alpha_0|0\rangle^{\otimes k}|\psi_0\rangle + |0^\perp\rangle^{\otimes k}|\cdots\rangle$, when we are only concerned about $|\psi_0\rangle$ and neglect the garbage state.

TABLE II. The complexity comparison of quantum linear solvers, where s is the sparsity of A and $\mu \leqslant \|A\|_F$.

Algorithm	Cost
HHL [18]	$O(s\kappa^2(\log n)/\epsilon)$
CKS [19]	$O(s\kappa(\log n)\text{poly}\log(s\kappa/\epsilon))$
WZP [20]	$O(\kappa^2\|A\|_F(\text{poly}\log n)/\epsilon)$
CGJ [14]	$O(\mu\kappa\text{poly}\log(n/\epsilon))$
This paper (Alg. 1, 2)	$O(\kappa_s^2(\log n)(\log \frac{1}{\epsilon}))$

TABLE III. The techniques used in different quantum linear solvers, where HS = Hamiltonian simulation, QPE = quantum phase estimation.

Algorithm	HS	QPE	QRAM
HHL	√	√	
CKS	√		
WZP	√	√	√
CGJ	√	√	
This paper (Alg. 1, 2)			√

(ii). We denote $\text{SWAP}_{i,j}$ as the swap operator that swaps the ith qubit and the jth qubit.

II. QUANTUM KACZMARZ ALGORITHM

Assume that the quantum state of $\mathbf{a}_t$ can be prepared efficiently in the quantum computer, such as by QRAM. So for each $t \in \{1, \ldots, n\}$, there is an efficiently implemented unitary operator V_t such that $V_t|0\rangle = |\mathbf{a}_t\rangle$. Based on the iteration formula (1), without loss of generality, we suppose that $\|\mathbf{a}_t\| = 1$ for all t. This is not a mandatory assumption; however, it can simplify the notations below. By Eq. (1), we have $\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{a}_{i_k}(\mathbf{a}_{i_k}^T \mathbf{x}_k - b_{i_k})$, that is,

$$|\mathbf{x}_{k+1}\rangle \propto \|\mathbf{x}_k\|\big(I_n - |\mathbf{a}_{i_k}\rangle\langle\mathbf{a}_{i_k}|\big)|\mathbf{x}_k\rangle + b_{i_k}|\mathbf{a}_{i_k}\rangle. \tag{6}$$

For any row index t, define a unitary operator

$$\begin{aligned} U_t &= \begin{bmatrix} I_n - |\mathbf{a}_t\rangle\langle\mathbf{a}_t| & |\mathbf{a}_t\rangle\langle\mathbf{a}_t| \\ |\mathbf{a}_t\rangle\langle\mathbf{a}_t| & I_n - |\mathbf{a}_t\rangle\langle\mathbf{a}_t| \end{bmatrix} \\ &= I_2 \otimes (I_n - |\mathbf{a}_t\rangle\langle\mathbf{a}_t|) + X \otimes |\mathbf{a}_t\rangle\langle\mathbf{a}_t|, \end{aligned} \tag{7}$$

where X is the Pauli X matrix. We can rewrite U_t as

$$U_t = (I_2 \otimes V_t)(I_2 \otimes (I_n - |0\rangle\langle 0|) + X \otimes |0\rangle\langle 0|)(I_2 \otimes V_t^\dagger). \tag{8}$$

From this decomposition, we find that U_t is determined by V_t. By the QRAM assumption, V_t is efficiently implemented on the quantum computer, and so is U_t.

By Eq. (7), U_t can be viewed as a control operator: if the second register is $|\mathbf{a}_t\rangle$, then apply X to the first register; if the second register lies in the orthogonal complement space of $|\mathbf{a}_t\rangle$, then do nothing to the first register.

The basic idea of the quantum implementation of Kaczmarz iteration is as follows: Suppose that the quantum information of $|\mathbf{x}_k\rangle$ is contained in $|X_k\rangle = \sqrt{p}\,|0\rangle|\mathbf{x}_k\rangle + \sqrt{1-p}\,|1\rangle|\cdots\rangle$. Let $\beta^2 + \gamma^2 = 1$, then we can prepare

$$\begin{aligned} |\psi\rangle &= \text{SWAP}_{1,2}(\beta|0\rangle|X_k\rangle + \gamma|1\rangle|0\rangle|\mathbf{a}_t\rangle) \\ &= |0\rangle(\beta\sqrt{p}\,|0\rangle|\mathbf{x}_k\rangle + \gamma|1\rangle|\mathbf{a}_t\rangle) + \beta\sqrt{1-p}\,|1\rangle|0\rangle|\cdots\rangle. \end{aligned} \tag{9}$$

As for the first term, a simple calculation yields

$$\begin{aligned} &U_t(\beta\sqrt{p}\,|0\rangle|\mathbf{x}_k\rangle + \gamma|1\rangle|\mathbf{a}_t\rangle) \\ &\quad = |0\rangle \otimes (\beta\sqrt{p}(I_n - |\mathbf{a}_t\rangle\langle\mathbf{a}_t|)|\mathbf{x}_k\rangle + \gamma|\mathbf{a}_t\rangle) \\ &\qquad + \beta\sqrt{p}\langle\mathbf{a}_t|\mathbf{x}_k\rangle|1\rangle|\mathbf{a}_t\rangle. \end{aligned} \tag{10}$$

Properly choosing the parameters β, γ, for example, $\beta = \|\mathbf{x}_k\|\delta$, $\gamma = b_t\sqrt{p}\delta$ for some δ to ensure $\beta^2 + \gamma^2 = 1$, then the first term of Eq. (10) is a state proportional to the right-hand side of Eq. (6).

The explicit procedure to implement Kaczmarz iteration is stated in Algorithm 1 as follows.

Algorithm 1. The quantum Kaczmarz method.

1: Randomly choose a unit vector $\mathbf{x}_0$ such that its quantum state is prepared in time $O(\log n)$. Set $k = 0$ and $\nu_k = 1$. The initial state can be expressed in the following general form:

$$|X_k\rangle = \frac{\|\mathbf{x}_k\|}{\nu_k}|0\rangle^{\otimes k} \otimes |\mathbf{x}_k\rangle + |0^{\perp}\rangle^{\otimes k}|\cdots\rangle. \tag{11}$$

2: Randomly choose t_k from the index set $\{1, \ldots, n\}$. Define $\beta_{t_k}^2 = \frac{\nu_k^2}{\nu_k^2 + b_{t_k}^2}$, $\gamma_{t_k}^2 = 1 - \beta_{t_k}^2$ and $\nu_{k+1} = \frac{\nu_k}{\beta_{t_k}}$. Then prepare the state

$$|Y_k\rangle = \beta_{t_k}|0\rangle|X_k\rangle + \gamma_{t_k}|1\rangle|0\rangle^{\otimes k}|\mathbf{a}_{t_k}\rangle.$$

3: Apply $(I_2^{\otimes k} \otimes U_{t_k})\,\mathrm{SWAP}_{1,k+1}$ to $|Y_k\rangle$, then we obtain

$$|X_{k+1}\rangle = \frac{\|\mathbf{x}_{k+1}\|}{\nu_{k+1}}|0\rangle^{\otimes(k+1)} \otimes |\mathbf{x}_{k+1}\rangle + |0^{\perp}\rangle^{\otimes(k+1)}|\cdots\rangle.$$

4: Set $k = k + 1$, and go to step 2 until convergence.

To generate the state $|Y_k\rangle$ in step 2, we first prepare $(\beta_{t_k}|0\rangle + \gamma_{t_k}|1\rangle)|0\rangle$ using the rotation determined by β_{t_k} and γ_{t_k}, then use the controlled operator that applies $|X_k\rangle$ and $|\mathbf{a}_{t_k}\rangle$ on the second register. In step 3, we calculate that

$$\begin{aligned}
&I_2^{\otimes k} \otimes U_{t_k}\left(\frac{\beta_{t_k}\|\mathbf{x}_k\|}{\nu_k}|0\rangle^{\otimes k}|0\rangle|\mathbf{x}_k\rangle + \gamma_{t_k}|0\rangle^{\otimes k}|1\rangle|\mathbf{a}_{t_k}\rangle\right.\\
&\quad\left. + |0^{\perp}\rangle^{\otimes k}|0\rangle|\cdots\rangle\right)\\
&= |0\rangle^{\otimes(k+1)} \otimes \left[\frac{\beta_{t_k}\|\mathbf{x}_k\|}{\nu_k}\left(I_n - |\mathbf{a}_{t_k}\rangle\langle\mathbf{a}_{t_k}|\right)|\mathbf{x}_k\rangle + \gamma_{t_k}|\mathbf{a}_{t_k}\rangle\right]\\
&\quad + |0^{\perp}\rangle^{\otimes(k+1)}|\cdots\rangle\\
&= \frac{\beta_{t_k}}{\nu_k}|0\rangle^{\otimes(k+1)} \otimes \left[\|\mathbf{x}_k\|\left(I_n - |\mathbf{a}_{t_k}\rangle\langle\mathbf{a}_{t_k}|\right)|\mathbf{x}_k\rangle + b_{t_k}|\mathbf{a}_{t_k}\rangle\right]\\
&\quad + |0^{\perp}\rangle^{\otimes(k+1)}|\cdots\rangle,
\end{aligned} \tag{12}$$

where we use the fact that $\gamma_{t_k} = \sqrt{1 - \beta_{t_k}^2} = \beta_{t_k} b_{t_k}/\nu_k$ in the last equality.

Similar to the classical Kaczmarz method, Algorithm 1 is also simple to implement on a quantum computer. Let $\mathbf{x}_k$ be the result obtained by the classical Kaczmarz method in the kth step. Then the first term of $|X_k\rangle$ defined in Eq. (11) contains all the information of $\mathbf{x}_k$ precisely, i.e., $\|\mathbf{x}_k\||\mathbf{x}_k\rangle$.

Theorem 1. Assume that $|\mathbf{a}_t\rangle$ is prepared in $O(\log n)$ for any t. In Algorithm 1, for any $k \geqslant 1$, we have

$$\nu_k^2 = 1 + \sum_{i=1}^{k-1} b_{t_i}^2. \tag{13}$$

The complexity to prepare $|X_k\rangle$ is $O(k \log n)$.

Proof. By definition in step 2 of Algorithm 1,

$$\nu_{k+1}^2 = \frac{\nu_k^2}{\beta_{t_k}^2} = \nu_k^2 + b_{t_k}^2.$$

Since $\nu_0 = 1$, we have

$$\nu_k^2 = 1 + \sum_{i=1}^{k-1} b_{t_i}^2.$$

Assume that the complexity to prepare $|X_k\rangle$ is τ_k, then the complexity for $|X_{k+1}\rangle$ in step 3 is $\tau_k + O(\log n)$ since the time for preparing $|\mathbf{a}_{t_k}\rangle$ is $O(\log n)$. Thus, $\tau_{k+1} = \tau_k + O(\log n)$. Since $\tau_0 = O(\log n)$, we have $\tau_k = O(k \log n)$. ∎

Strohmer *et al.* [13] sample a row in a random fashion with probability proportional to the squared 2-norm of that row at each iteration, and prove an exponential expected convergence rate of the randomized Kaczmarz method. Under such discrete sampling we have the expectation

$$\mathbb{E}\left[b_{t_i}^2\right] = \sum_{i=1}^{n} b_i^2 \frac{\|\mathbf{a}_i\|^2}{\|A\|_F^2} = \frac{\|D_{\mathbf{b}}A\|_F^2}{\|A\|_F^2},$$

where $D_{\mathbf{b}} = \mathrm{diag}(b_1, \ldots, b_n)$ is the diagonal matrix with the diagonal entries defined by the vector $\mathbf{b}$. Therefore,

$$\mathbb{E}\left[\nu_k^2\right] = 1 + (k-1)\mathbb{E}\left[b_{t_i}^2\right] = 1 + (k-1)\frac{\|D_{\mathbf{b}}A\|_F^2}{\|A\|_F^2}.$$

Since $\|D_{\mathbf{b}}A\|_F^2 = \sum_{i=1}^n b_i^2\|\mathbf{a}_i\|^2 \leqslant \|\mathbf{b}\|_\infty^2\|A\|_F^2$, we have $\mathbb{E}[\nu_k^2] \leqslant 1 + (k-1)\|\mathbf{b}\|_\infty^2$. In addition, $\mathbb{E}[\nu_k] = O(\sqrt{k})$.

After it converges, in $|X_k\rangle$, we can apply the amplitude amplification technique to increase the amplitude of $|\mathbf{x}_k\rangle$. The complexity is $O(k(\log n)\nu_k/\|\mathbf{x}_k\|)$. But it maybe better not performing amplitude amplification since the norm information of the solution $\mathbf{x}_k$ is stored in the first term of $|X_k\rangle$. For some problems in machine learning, such as data fitting, the final output extracts certain global information, rather than each component of a state. For example, to estimate the inner product between $\mathbf{x}_k$ and the vector $\mathbf{c}$, we can first prepare the quantum state $|\mathbf{c}\rangle$, then apply the swap test to estimate the inner product between $|X_k\rangle$ and $|0\rangle^{\otimes k}|\mathbf{c}\rangle$. This returns an ϵ' approximate of $\|\mathbf{x}_k\|\langle\mathbf{x}_k|\mathbf{c}\rangle/\nu_k$ in time $O(k(\log n)/\epsilon')$. Thus, by setting $\epsilon' = \epsilon/\nu_k\|\mathbf{c}\|$, we will obtain an ϵ approximate of $\mathbf{x}_k \cdot \mathbf{c}$ in time $O(k(\log n)\nu_k\|\mathbf{c}\|/\epsilon)$. We can also use amplitude estimation to estimate the norm $\|\mathbf{x}_k\|$ of the solution. This will cost $O(k(\log n)\nu_k/\epsilon)$.

III. QUANTUM COLUMN ITERATION ALGORITHM

In this section, we apply similar techniques to Algorithm 1 to implement the column iteration algorithm on a quantum computer. Denote the jth column of A as $\mathbf{c}_j$ and assume that it is normalized for simplicity. Suppose the quantum state of $\mathbf{c}_j$ can be efficiently prepared. That is, for each j, there exists an efficient unitary operator S_j such that $S_j^{\dagger}|j\rangle = |\mathbf{c}_j\rangle$.

The column iteration method can be implemented as follows. (1) Randomly choose an initial guess $\mathbf{x}_0$ and set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$. (2) Randomly choose an index parameter $t_k \in \{1, \ldots, n\}$ and update

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{c}_{t_k}^T\mathbf{r}_k\mathbf{e}_{t_k}, \quad \mathbf{r}_{k+1} = \left(I_n - \mathbf{c}_{t_k}\mathbf{c}_{t_k}^T\right)\mathbf{r}_k. \tag{14}$$

Using quantum states, we can rewrite Eq. (14) as

$$|\mathbf{x}_{k+1}\rangle \propto \|\mathbf{x}_k\|\,|\mathbf{x}_k\rangle + \|\mathbf{r}_k\|\,|t_k\rangle\langle \mathbf{c}_{t_k}|\mathbf{r}_k\rangle,$$
$$|\mathbf{r}_{k+1}\rangle \propto \|\mathbf{r}_k\|\,(I_n - |\mathbf{c}_{t_k}\rangle\langle \mathbf{c}_{t_k}|)|\mathbf{r}_k\rangle. \tag{15}$$

Before implementing the procedure (15) on the quantum computer, we explain the main idea below. First, we consider the update of the residual. The basic idea is the same as Algorithm 1. Suppose that the residual of the kth step is encoded in the state

$$|R_k\rangle = \|\mathbf{r}_k\|\,|0\rangle^{\otimes k}|\mathbf{r}_k\rangle + |0^\perp\rangle^{\otimes k}|\cdots\rangle. \tag{16}$$

Apply $(I_2^{\otimes k}\otimes U_{t_k})\text{SWAP}_{1,k+1}$ to $|0\rangle|R_k\rangle$, then

$$\begin{aligned}&\left(I_2^{\otimes k}\otimes U_{t_k}\right)\text{SWAP}_{1,k+1}|0\rangle|R_k\rangle\\ &= |0\rangle^{\otimes(k+1)}\|\mathbf{r}_k\|(I_n - |\mathbf{c}_{t_k}\rangle\langle \mathbf{c}_{t_k}|)|\mathbf{r}_k\rangle + |0^\perp\rangle^{\otimes(k+1)}|\cdots\rangle\\ &= \|\mathbf{r}_{k+1}\|\,|0\rangle^{\otimes(k+1)}|\mathbf{r}_{k+1}\rangle + |0^\perp\rangle^{\otimes(k+1)}|\cdots\rangle\\ &= |R_{k+1}\rangle.\end{aligned} \tag{17}$$

This is in fact the Algorithm 1 with initial vector $\mathbf{r}_0$ and $\nu_k = 1$ for all k.

Second, we consider the update of the approximate solution. Since $\langle t|S_t = \langle \mathbf{c}_t|$, we have

$$|\mathbf{x}_{k+1}\rangle \propto \|\mathbf{x}_k\|\,|\mathbf{x}_k\rangle + \|\mathbf{r}_k\|\,|t_k\rangle\langle t_k|S_{t_k}|\mathbf{r}_k\rangle. \tag{18}$$

Algorithm 1 is not applicable to the above procedure directly. Some modifications are required.

Suppose that the approximate solution $\mathbf{x}_k$ and the corresponding residual $\mathbf{r}_k$ are encoded in the following states, respectively,

$$|\widetilde{X}_k\rangle = \frac{\|\mathbf{x}_k\|}{\nu}|0\rangle|\mathbf{x}_k\rangle + |0^\perp\rangle|\cdots\rangle, \tag{19}$$

and

$$|\widetilde{R}_k\rangle = \|\mathbf{r}_k\|\,|0\rangle S_{t_k}|\mathbf{r}_k\rangle + |0^\perp\rangle|\cdots\rangle. \tag{20}$$

We then need to combine them to generate a state that contains $|\mathbf{x}_{k+1}\rangle$.

First, we introduce two auxilla qubits to prepare the superposition of the two states (19) and (20),

$$|\phi_1\rangle = \beta|00\rangle|\widetilde{X}_k\rangle + \gamma|10\rangle|\widetilde{R}_k\rangle, \tag{21}$$

where $\beta^2+\gamma^2 = 1$. The state $|\phi_1\rangle$ is obtained in a similar way to $|Y_k\rangle$ defined in step 2 of Algorithm 1. For any t, define

$$W_t = \begin{bmatrix} I_n & 0 & 0\\ 0 & I_n - |t\rangle\langle t| & |t\rangle\langle t|\\ 0 & |t\rangle\langle t| & I_n - |t\rangle\langle t|\end{bmatrix}. \tag{22}$$

Apply $\text{SWAP}_{2,3}W_{t_k}$ to $|\phi_1\rangle$ to prepare

$$\begin{aligned}|\phi_2\rangle &= \text{SWAP}_{2,3}W_{t_k}|\phi_1\rangle\\ &= \text{SWAP}_{2,3}(\beta|00\rangle|\widetilde{X}_k\rangle + \gamma|01\rangle|t_k\rangle\langle t_k|\widetilde{R}_k\rangle + |10\rangle|\cdots\rangle)\\ &= |00\rangle\left(\frac{\beta\|\mathbf{x}_k\|}{\nu}|0\rangle|\mathbf{x}_k\rangle + \gamma\|\mathbf{r}_k\|\,|1\rangle|t_k\rangle\langle t_k|S_{t_k}|\mathbf{r}_k\rangle\right)\\ &\quad + |0^\perp\rangle^{\otimes 2}|\cdots\rangle.\end{aligned} \tag{23}$$

The operator W_{t_k} helps us to compute $\langle t_k|S_{t_k}|\mathbf{r}_k\rangle$ without using the swap test.

To obtain the linear combinations of the right-hand side of Eq. (18), the next step is to apply a single qubit operator to the first register. Here we choose a rotation $G_k = \left[\begin{smallmatrix} c & s\\ -s & c\end{smallmatrix}\right]$, where $c^2+s^2 = 1$. Apply $I_4\otimes G_k\otimes I_n$ to $|\phi_2\rangle$, then we obtain

$$\begin{aligned}|\phi_3\rangle &= (I_4\otimes G_k\otimes I_n)|\phi_2\rangle\\ &= |000\rangle\left(\frac{c\beta\|\mathbf{x}_k\|}{\nu}|\mathbf{x}_k\rangle + s\gamma\|\mathbf{r}_k\|\,|t_k\rangle\langle t_k|S_{t_k}|\mathbf{r}_k\rangle\right)\\ &\quad + |0^\perp\rangle^{\otimes 3}|\cdots\rangle.\end{aligned} \tag{24}$$

We can properly choose the parameters c, s, β, and γ such that the first term of $|\phi_3\rangle$ is proportional to

$$\|\mathbf{x}_k\|\,|\mathbf{x}_k\rangle + \|\mathbf{r}_k\|\,|t_k\rangle\langle t_k|S_{t_k}|\mathbf{r}_k\rangle = \|\mathbf{x}_{k+1}\|\,|\mathbf{x}_{k+1}\rangle. \tag{25}$$

With the preparations above, we can present the quantum column iteration algorithm as follows, where the rotation G_k is defined by

$$G_k = \frac{1}{\sqrt{k+2}}\begin{bmatrix}\sqrt{k+1} & 1\\ -1 & \sqrt{k+1}\end{bmatrix}. \tag{26}$$

Algorithm 2. The quantum column iteration.

1: Randomly choose a unit vector $\mathbf{x}_0$ such that its quantum state can be prepared in time $O(\log n)$. Assume that $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ has unit norm and its quantum state is prepared in time $O(\log n)$. Set $k = 0$. Denote

$$|X_k\rangle = \frac{\|\mathbf{x}_k\|}{k+1}|0\rangle^{\otimes 2k}\otimes|\mathbf{x}_k\rangle + |0^\perp\rangle^{\otimes 2k}|\cdots\rangle,$$
$$|R_k\rangle = \|\mathbf{r}_k\|\,|0\rangle^{\otimes k}|\mathbf{r}_k\rangle + |0^\perp\rangle^{\otimes k}|\cdots\rangle. \tag{27}$$

2: Randomly choose t_k from the set $\{1,\ldots,n\}$, and prepare the state

$$\sqrt{\frac{k+1}{k+2}}\,|00\rangle|X_k\rangle + \sqrt{\frac{1}{k+2}}\,|10\rangle(I^{\otimes 2k}\otimes S_{t_k})|0\rangle^{\otimes k}|R_k\rangle, \tag{28}$$

3: Apply $(I_2^{\otimes(2k+1)}\otimes G_k\otimes I_n)(I_2^{\otimes 2k}\otimes W_{t_k})\text{SWAP}_{2,2k+2}\,\text{SWAP}_{1,2k+1}$ to the state (28), then we obtain $|X_{k+1}\rangle$.

4: Apply $(I_2^{\otimes k}\otimes U_{t_k})\text{SWAP}_{1,k+1}$ to $|0\rangle|R_k\rangle$ to generate $|R_{k+1}\rangle$.

5: Set $k = k+1$, and go to step 2 until convergence.

In the algorithm above, the choices with $\beta = c = \sqrt{(k+1)/(k+2)}$ and $\gamma = s = \sqrt{1/(k+2)}$ are optimal. The analysis is easy since we need to keep $c\beta/(k+1) = s\gamma = 1/(k+2)$ in Eq. (24), where $\nu = k+1$ comes from Eq. (27).

We explain the update of approximate solution in details. The state (28) in step 2 is denoted as $|\psi_0\rangle$, that is,

$$\begin{aligned}|\psi_0\rangle &= \sqrt{\frac{k+1}{k+2}}\,|00\rangle\left(\frac{\|\mathbf{x}_k\|}{k+1}\,|0\rangle^{\otimes 2k}|\mathbf{x}_k\rangle + |0^\perp\rangle^{\otimes 2k}|\cdots\rangle\right)\\ &\quad + \sqrt{\frac{1}{k+2}}\,|10\rangle\left(\|\mathbf{r}_k\|\,|0\rangle^{\otimes 2k}S_{t_k}|\mathbf{r}_k\rangle + |0^\perp\rangle^{\otimes 2k}|\cdots\rangle\right).\end{aligned} \tag{29}$$

If we apply $\mathrm{SWAP}_{2,2k+2}\mathrm{SWAP}_{1,2k+1}$ to $|\psi_0\rangle$, then we obtain

$$|\psi_1\rangle = \sqrt{\frac{k+1}{k+2}}\left(\frac{\|\mathbf{x}_k\|}{k+1}|0\rangle^{\otimes 2k}|00\rangle|\mathbf{x}_k\rangle + |0^\perp\rangle^{\otimes 2k}|00\rangle|\cdots\rangle\right) + \sqrt{\frac{1}{k+2}}\left(\|\mathbf{r}_k\||0\rangle^{\otimes 2k}|10\rangle S_{t_k}|\mathbf{r}_k\rangle + |0^\perp\rangle^{\otimes 2k}|10\rangle|\cdots\rangle\right)$$
$$= |0\rangle^{\otimes 2k}\otimes\left(\sqrt{\frac{k+1}{k+2}}\frac{\|\mathbf{x}_k\|}{k+1}|00\rangle|\mathbf{x}_k\rangle + \sqrt{\frac{1}{k+2}}\|\mathbf{r}_k\||10\rangle S_{t_k}|\mathbf{r}_k\rangle\right) + |0^\perp\rangle^{\otimes 2k}|\cdots\rangle. \tag{30}$$

Apply $I_2^{\otimes 2k}\otimes W_{t_k}$ to $|\psi_1\rangle$ to generate the state

$$|\psi_2\rangle = |0\rangle^{\otimes(2k+1)}\otimes\left(\sqrt{\frac{k+1}{k+2}}\frac{\|\mathbf{x}_k\|}{k+1}|0\rangle|\mathbf{x}_k\rangle + \frac{\|\mathbf{r}_k\|}{\sqrt{k+2}}|1\rangle|t_k\rangle\langle t_k|S_{t_k}|\mathbf{r}_k\rangle\right) + |0^\perp\rangle^{\otimes(2k+1)}|\cdots\rangle. \tag{31}$$

Finally, apply $I_2^{\otimes(2k+1)}\otimes G_k\otimes I_n$ to $|\psi_2\rangle$, then we obtain

$$|\psi_3\rangle = |0\rangle^{\otimes 2(k+1)}\otimes\left(\frac{k+1}{k+2}\frac{\|\mathbf{x}_k\|}{k+1}|\mathbf{x}_k\rangle + \frac{\|\mathbf{r}_k\|}{k+2}|t_k\rangle\langle t_k|S_{t_k}|\mathbf{r}_k\rangle\right) + \left|0^\perp\right\rangle^{\otimes 2(k+1)}|\cdots\rangle$$
$$= \frac{\|\mathbf{x}_{k+1}\|}{k+2}|0\rangle^{\otimes 2(k+1)}\otimes|\mathbf{x}_{k+1}\rangle + |0^\perp\rangle^{\otimes 2(k+1)}|\cdots\rangle$$
$$= |X_{k+1}\rangle. \tag{32}$$

Similar to the proof of Theorem 1, we have the following result.

Theorem 2. In Algorithm 2 the complexity to prepare $|X_k\rangle$ is $O(k\log n)$.

Remark 1. Since $\|\mathbf{x}_0\| = 1$, now assume that $\|\mathbf{x}_k\| \leqslant k+1$ and $\|\mathbf{r}_0\| = 1$, then by Eq. (15),

$$\|\mathbf{x}_{k+1}\|^2 = \|\mathbf{x}_k\|^2 + 2\|\mathbf{r}_k\|\|\mathbf{x}_k\|\langle\mathbf{c}_{t_k}|\mathbf{r}_k\rangle\langle t_k|\mathbf{x}_k\rangle + \|\mathbf{r}_k\|^2\langle\mathbf{c}_{t_k}|\mathbf{r}_k\rangle^2$$
$$\leqslant (k+1)^2 + 2(k+1) + 1 = (k+2)^2.$$

Therefore, by induction the Eq. (27) is well defined.

Remark 2. By definition, $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, with a suitable choice of $\mathbf{x}_0$ we can make sure that it has unit norm. Even if it does not have unit norm, Algorithm 2 still works. Let ρ be a parameter such that $\rho\|\mathbf{r}_0\| \leqslant 1$. In Algorithm 2, it suffices to change $|R_0\rangle$ into $|\widehat{R}_0\rangle = \rho\|\mathbf{r}_0\||0\rangle|\mathbf{r}_0\rangle + |0^\perp\rangle|\cdots\rangle$. By Eq. (15), $\|\mathbf{r}_{k+1}\||\mathbf{r}_{k+1}\rangle = \|\mathbf{r}_k\|(I_n - |\mathbf{c}_{t_k}\rangle\langle\mathbf{c}_{t_k}|)|\mathbf{r}_k\rangle$. Therefore, $|R_k\rangle$ used in Algorithm 2 is simply changed into $|\widehat{R}_k\rangle = \rho\|\mathbf{r}_k\||0\rangle^{\otimes(k+1)}|\mathbf{r}_k\rangle + |0^\perp\rangle^{\otimes(k+1)}|\cdots\rangle$, where $k+1$ ancilla qubits are used due to the extra one ancilla qubit introduced in $|\widehat{R}_0\rangle$. Since $\rho\|\mathbf{r}_0\| \leqslant 1$, we have $\rho\|\mathbf{r}_k\| \leqslant 1$ for all k.

Remark 3. For the update of residual we can use the linear combinations of unitaries (LCU) [21]. It contains many applications in quantum computing, such as quantum simulation [21,22], quantum linear solver [19]. Let $U_0,\ldots,U_{m-1}$ be m unitary operators, and $\alpha_0,\ldots,\alpha_{m-1}$ be m positive real numbers. Set $s = \sum_j \alpha_j$. Assume that V is a unitary operator that maps $|0\rangle^{\otimes\log m}$ to $\frac{1}{\sqrt{s}}\sum_j\sqrt{\alpha_j}|j\rangle$. Given a quantum state $|\psi\rangle$, the technique of LCU can compute $\sum_j\alpha_j U_j|\psi\rangle$. In our case, if we choose $U_0 = I_n$, $U_1 = I_n - 2|\mathbf{c}_{t_k}\rangle\langle\mathbf{c}_{t_k}|$, and $\alpha_0 = \alpha_1 = 1/2$, then we can prepare $(I_n - |\mathbf{c}_{t_k}\rangle\langle\mathbf{c}_{t_k}|)|\psi\rangle$. However, by using LCU, we will need two times the ancilla qubits in preparing $|X_k\rangle$ in Algorithm 1, and $|R_k\rangle$ in Algorithm 2.

Denote the number of required iterations by T. From Theorems 1 and 2, we find that the complexity of quantum Kaczmarz and column iteration method is $O(T\log n)$. However, the classical counterparts have complexity $O(Tn)$. Therefore, in comparison, the quantum iterative algorithms achieve an exponential speedup in the problem size n while keep the same dependence on number of steps T.

IV. CONCLUSION

In this paper, we investigate the row (Kaczmarz) method and the column (coordinate descent) method, and show their simple implementations on a quantum computer based on the idea of block-encoding. With the QRAM, the efficiency is exponentially better in the problem size than their classical counterparts. Both quantum and classical Kaczmarz and column iterations have linear dependence on the number of iteration steps.

Our quantum iterative linear solvers are different from previous quantum linear solvers [18–20,23]. For solving linear systems, our algorithms may not be superior to other quantum linear solvers (see Table II). But the methods proposed in this paper are independent of Hamiltonian simulation and quantum phase estimation. They also work for the dense linear systems. The only assumption on our algorithms is the availability of QRAM, by which we can efficiently extract the quantum states of the rows or columns of the coefficient matrix. One drawback of our quantum iterative linear solvers is the increasing number of ancilla qubits. Since the iteration procedure is generally not unitary, we need to introduce ancilla qubits to transform it into a unitary one. It remains to be an unsolved problem to find better ways to reduce the number of ancilla qubits, such as using the idea of the authors of [24] to implement nonunitary operations.

ACKNOWLEDGMENTS

H.X. is supported by the Natural Science Foundation of China under Grants No. 11571265 and NSFC-RGC No. 11661161017. C.S. was supported by the QuantERA ERA-NET Cofund in Quantum Technologies implemented within

the European Union's Horizon 2020 Programme (QuantAlgo project), and EPSRC Grants No. EP/L021005/1 and No. EP/R043957/1.

APPENDIX: DERIVATION OF THE ITERATIVE SCHEMES

The row and column iteration methods can be derived in the following aspects: (1) the Gauss-Seidel method on $AA^T\mathbf{y} = \mathbf{b}$ and $A^TA\mathbf{x} = A^T\mathbf{b}$ respectively; (2) the minimization of quadratic functions $\psi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^TAA^T\mathbf{y} - \mathbf{y}^T\mathbf{b}$ and $\varphi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^TA^TA\mathbf{x} - \mathbf{x}^TA^T\mathbf{b}$ in one coordinate direction at each iteration step.

(1) The Gauss-Seidel method. For $A\mathbf{x} = \mathbf{b}$ where $A \in \mathbb{R}^{n\times n}$, the componentwise Gauss-Seidel scheme reads

$$x_i^{(l+1)} = x_i^{(l)} + \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(l+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(l)}\right),$$

where the subscript i stands for the component ($i = 1, \ldots, n$), and the subscript (l) denotes the iteration step. The term in the bracket is the ith component of the current residual $\mathbf{r} = \mathbf{b} - A\mathbf{x}$, where the approximate solution $\mathbf{x} = (x_1^{(l+1)}, \ldots, x_{i-1}^{(l+1)}, x_i^{(l)}, \ldots, x_n^{(l)})^T$. For simplicity, we rewrite it in a compact vector form

$$\mathbf{x} \leftarrow \mathbf{x} + \frac{1}{a_{ii}}\mathbf{e}_i^T(\mathbf{b} - A\mathbf{x})\mathbf{e}_i, \quad (i = 1, \ldots, n).$$

Note that the component index i can be chosen randomly from $\{1, \ldots, n\}$ with certain probability.

(i) For $AA^T\mathbf{y} = \mathbf{b}$, the diagonal element of AA^T is $\mathbf{a}_i^T\mathbf{a}_i$, where $\mathbf{a}_i$ is the ith column of A^T. Then the Gauss-Seidel scheme for updating $\mathbf{y}$ reads

$$\mathbf{y} \leftarrow \mathbf{y} + \frac{\mathbf{e}_i^T(\mathbf{b} - AA^T\mathbf{y})}{\mathbf{a}_i^T\mathbf{a}_i}\mathbf{e}_i.$$

Left multiplying A^T, setting $\mathbf{x} = A^T\mathbf{y}$ and $A^T\mathbf{e}_i = \mathbf{a}_i$, we have

$$\mathbf{x} \leftarrow \mathbf{x} + \frac{\mathbf{e}_i^T(\mathbf{b} - A\mathbf{x})}{\mathbf{a}_i^T\mathbf{a}_i}\mathbf{a}_i.$$

Thus, we have the Kaczmarz scheme (1).

(ii) For $A^TA\mathbf{x} = A^T\mathbf{b}$, the diagonal element of A^TA is $\mathbf{c}_i^T\mathbf{c}_i$, where $\mathbf{c}_i$ is the ith column of A. The Gauss-Seidel method updates $\mathbf{x}$ as follows:

$$\mathbf{x} \leftarrow \mathbf{x} + \frac{\mathbf{e}_i^T(A^T\mathbf{b} - A^TA\mathbf{x})}{\mathbf{c}_i^T\mathbf{c}_i}\mathbf{e}_i.$$

Noticing that $\mathbf{e}_i^TA^T = \mathbf{c}_i^T$, we have

$$\mathbf{x} \leftarrow \mathbf{x} + \frac{\mathbf{c}_i^T(\mathbf{b} - A\mathbf{x})}{\|\mathbf{c}_i\|^2}\mathbf{e}_i.$$

Thus, we have the column iteration scheme (2). Recalling that $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$, it can be rewritten as

$$\alpha_i = \frac{\mathbf{c}_i^T\mathbf{r}_k}{\|\mathbf{c}_i\|^2},$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_i\mathbf{e}_i,$$
$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_i\mathbf{c}_i.$$

(2) The minimization of quadratic functions. We can derive the iterative scheme by casting the solution to a quadratic semidifinite programming (SDP) and minimizing in one coordinate direction at each step.

(i) Minimization of $\psi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^TAA^T\mathbf{y} - \mathbf{y}^T\mathbf{b}$. Let us start from the approximate solution $\mathbf{y}$ and minimize it along the direct $\mathbf{e}_i$. That is, we seek the next approximate solution $\mathbf{y} + \gamma\mathbf{e}_i$, by minimizing

$$\tfrac{1}{2}(\mathbf{y} + \gamma\mathbf{e}_i)^TAA^T(\mathbf{y} + \gamma\mathbf{e}_i) - (\mathbf{y} + \gamma\mathbf{e}_i)^T\mathbf{b},$$

which is equivalent to minimizing

$$\tfrac{1}{2}\gamma^2\mathbf{e}_i^TAA^T\mathbf{e}_i - \gamma\mathbf{e}_i^T(\mathbf{b} - AA^T\mathbf{y}).$$

Therefore,

$$\gamma = \frac{\mathbf{e}_i^T(\mathbf{b} - AA^T\mathbf{y})}{\mathbf{e}_i^TAA^T\mathbf{e}_i} = \frac{\mathbf{e}_i^T(\mathbf{b} - A\mathbf{x})}{\mathbf{a}_i^T\mathbf{a}_i}.$$

The iterative scheme $\mathbf{y} \leftarrow \mathbf{y} + \gamma\mathbf{e}_i$, by using $A^T\mathbf{y} = \mathbf{x}$ and $A^T\mathbf{e}_i = \mathbf{a}_i$, can be reformulated as

$$\mathbf{x} \leftarrow \mathbf{x} + \gamma\mathbf{a}_i = \mathbf{x} + \frac{\mathbf{e}_i^T(\mathbf{b} - A\mathbf{x})}{\mathbf{a}_i^T\mathbf{a}_i}\mathbf{a}_i.$$

(ii) Minimization of $\varphi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^TA^TA\mathbf{x} - \mathbf{x}^TA^T\mathbf{b}$, which is equivalent to minimizing $\|\mathbf{b} - A\mathbf{x}\|^2$. Assume that the current approximate solution is $\mathbf{x}$, and we minimize the function along the direction $\mathbf{e}_i$. Set the next approximate solution as $\mathbf{x} + \gamma\mathbf{e}_i$, then we need to minimize

$$\tfrac{1}{2}(\mathbf{x} + \gamma\mathbf{e}_i)^TA^TA(\mathbf{x} + \gamma\mathbf{e}_i) - (\mathbf{x} + \gamma\mathbf{e}_i)^TA^T\mathbf{b},$$

which is equivalent to minimizing

$$\tfrac{1}{2}\gamma^2\mathbf{e}_i^TA^TA\mathbf{e}_i - \gamma\mathbf{e}_i^T(A^T\mathbf{b} - A^TA\mathbf{x}).$$

Therefore,

$$\gamma = \frac{\mathbf{e}_i^TA^T(\mathbf{b} - A\mathbf{x})}{\mathbf{e}_i^TA^TA\mathbf{e}_i} = \frac{\mathbf{c}_i^T(\mathbf{b} - A\mathbf{x})}{\mathbf{c}_i^T\mathbf{c}_i},$$

and the iterative scheme reads

$$\mathbf{x} \leftarrow \mathbf{x} + \gamma\mathbf{e}_i = \mathbf{x} + \frac{\mathbf{c}_i^T(\mathbf{b} - A\mathbf{x})}{\mathbf{c}_i^T\mathbf{c}_i}\mathbf{e}_i.$$

[1] S. Kaczmarz, Classe des Sciences Mathématiques et Naturelles. Série A, Sciences Mathématiques **35**, 355 (1937).
[2] R. Gordon, R. Bender, and G. Herman, J. Theor. Biol. **29**, 471 (1970).
[3] D. Leventhal and A. S. Lewis, Math. Oper. Res. **35**, 641 (2010).
[4] T. Elfving, P. C. Hansen, and T. Nikazad, Numer. Algor. **74**, 905 (2017).
[5] R. M. Gower and P. Richtárik, SIAM J. Matrix Anal. Appl. **36**, 1660 (2015).
[6] H. Xiang and L. Zhang, arXiv:1708.09845v1 (2017).

[7] Y. Censor, SIAM Rev. **23**, 444 (1981).
[8] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections* (Springer Science & Business Media, 2009).
[9] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd, New J. Phys. **21**, 073023 (2019).
[10] I. Kerenidis and A. Prakash, arXiv:1808.09266v1 (2018).
[11] I. Kerenidis and A. Prakash, arXiv:1704.04992v4 (2017).
[12] S. Lloyd, M. Mohseni, and P. Rebentrost, Nat. Phys. **10**, 631 (2014).
[13] T. Strohmer and R. Vershynin, J. Fourier Anal. Appl. **15**, 262 (2009).
[14] S. Chakraborty, A. Gilyén, and S. Jeffery, in *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, Leibniz International Proceedings in Informatics (LIPIcs), edited by C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2019), Vol. 132, pp. 33:1–33:14.
[15] V. Giovannetti, S. Lloyd, and L. Maccone, Phys. Rev. Lett. **100**, 160501 (2008).
[16] I. Kerenidis and A. Prakash, in *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), edited by C. H. Papadimitriou (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017), Vol. 67, pp. 49:1–49:21.
[17] L. Grover and T. Rudolph, arXiv:quant-ph/0208112 (2002).
[18] A. W. Harrow, A. Hassidim, and S. Lloyd, Phys. Rev. Lett. **103**, 150502 (2009).
[19] A. M. Childs, R. Kothari, and R. D. Somma, SIAM J. Comput. **46**, 1920 (2017).
[20] L. Wossnig, Z. Zhao, and A. Prakash, Phys. Rev. Lett. **120**, 050502 (2018).
[21] A. M. Childs and N. Wiebe, Quantum Inf. Comput. **12**, 901 (2012).
[22] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, Phys. Rev. Lett. **114**, 090502 (2015).
[23] Y. Subasi, R. D. Somma, and D. Orsucci, Phys. Rev. Lett. **122**, 060504 (2019).
[24] M. Motta, C. Sun, A. T. K. Tan, M. J. Rourke, E. Ye, A. J. Minnich, F. G. Brandao, and G. K. Chan, Nat. Phys. **16**, 231 (2020).