SCIENTIFIC REPORTS
nature research

OPEN

Hybrid classical-quantum linear solver using Noisy Intermediate-Scale Quantum machines

Chih-Chieh Chen[1*], Shiue-Yuan Shiau[2], Ming-Feng Wu[1] & Yuh-Renn Wu[3*]

We propose a realistic hybrid classical-quantum linear solver to solve systems of linear equations of a specific type, and demonstrate its feasibility with Qiskit on IBM Q systems. This algorithm makes use of quantum random walk that runs in O(*N* log(*N*)) time on a quantum circuit made of O(log(*N*)) qubits. The input and output are classical data, and so can be easily accessed. It is robust against noise, and ready for implementation in applications such as machine learning.

Algorithms that run on quantum computers hold promise to perform important computational tasks more efficiently than what can ever be achieved on classical computers, most notably Grover's search algorithm and Shor's integer factorization[1]. One computational task indispensable for many problems in science, engineering, mathematics, finance, and machine learning, is solving systems of linear equations $\mathbf{A}\vec{x} = \vec{b}$. Classical direct and iterative algorithms take $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ time[2,3]. Interestingly, the Harrow-Hassidim-Lloyd (HHL) quantum algorithm[4–13], which is based on the quantum circuit model[14], takes only $\mathcal{O}(\log(N))$ to solve a sparse $N \times N$ system of linear equations, while for dense systems it requires $\mathcal{O}(\sqrt{N}\log(N))$[11]. Linear solvers and experimental realizations that use quantum annealing and adiabatic quantum computing machines[15–17] are also reported[18–20]. Most recently, methods[21,22] inspired by adiabatic quantum computing are proposed to be implemented on circuit-based quantum computers. Whether substantial quantum speedup exists in these algorithms remains unknown.

In practice, the applicability of quantum algorithms to classical systems are limited by the short coherence time of noisy quantum hardware in the so-called Noisy Intermediate-Scale Quantum (NISQ) era[23] and the difficulty in executing the input and output of classical data. Other roadblocks toward practical implementation include limited number of qubits, limited connectivity between qubits, and large error correction overhead. At present, experiments demonstrating the HHL linear solver on circuit quantum computers are limited to 2×2 matrices[24–29], while linear solvers inspired by adiabatic quantum computing are limited to 8×8 matrices[21,22]. For quantum annealers, the state-of-the-art linear solvers can solve up to 12×12 matrices[20].

In addition to the problems of limited available entangled qubits and short coherence time, the HHL-type algorithms for the so-called Quantum Linear Systems Problem (QLSP) are designed to work only when input and output are quantum states[30]. This condition imposes severe restriction to practical applications in the NISQ era[23,30,31]. It has been shown that the HHL algorithm can not extract information about the norm of the solution vector $\vec{x}$[4]. A state preparation algorithm for inputting a classical vector $\vec{b}$ would take $\mathcal{O}(N)$ time[30,32–34], with large overhead for current hardware. In addition, quantum state tomography is required to read out the classical solution vector $\vec{x}$, which is a demanding task[35,36], except for special cases like one-dimensional entangled qubits[37]. Inputting the matrix $\mathbf{A}$ is also a challenge that may kill the quantum speedup[1,24–29].

In this work, we propose a hybrid classical-quantum linear solver that uses circuit-based quantum computer to perform quantum random walks. In contrast to the HHL-type linear solvers, the solution vector $\vec{x}$ and the constant vector $\vec{b}$ in this hybrid algorithm stay as classical data in the classical registers. Only the matrix $\mathbf{A}$ is encoded in quantum registers. The idea is similar to that of variational quantum eigensolvers[38–41], where quantum speedup is exploited only for sampling exponentially large state Hilbert spaces, while the rest of computational

[1]Electronic and Optoelectronic System Research Laboratories, Industrial Technology Research Institute, Hsinchu, 31057, Taiwan. [2]Physics Division, National Center for Theoretical Sciences, Hsinchu, 30013, Taiwan. [3]Graduate Institute of Photonics and Optoelectronics and Department of Electrical Engineering, National Taiwan University, Taipei, 10617, Taiwan. *email: helloqworld2019@gmail.com; yrwu@ntu.edu.tw

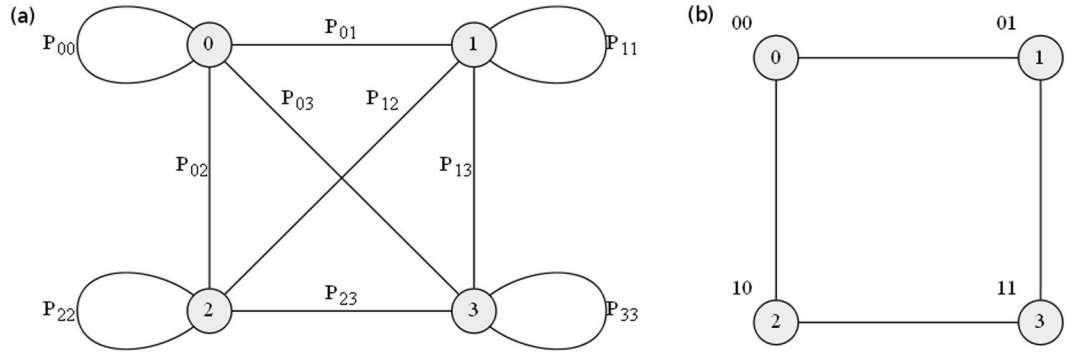

Figure 1. (**a**) Quantum (or classical) random walk on an undirected $N = 4$ graph. The transition probability of going from node I to node J or vice versa is equal to $P_{I,J}$, these elements forming a 4×4 matrix. (**b**) The four nodes on this Hamming cube are labeled by integers $(0, 1, 2, 3)$; they are encoded as four different states $|00\rangle$, $|01\rangle, |10\rangle, |11\rangle$, respectively.

task is done by classical computer. This makes it easy to perform data input and output: the $\overrightarrow{b}$ vector can be arbitrary, and the components and the norm of the $\overrightarrow{x}$ vector can be easily accessed.

We consider matrices that are useful for Markov decision problems such as in reinforcement learning[42]. We show that these matrices can be efficiently encoded by introducing the Hamming cube structure: a square matrix of size N requires $\mathcal{O}(\log(N))$ quantum bits only. The quantum random walk algorithm we here propose takes $\mathcal{O}(\log(N))$ time to obtain one component of the $\overrightarrow{x}$ vector. We also show that in the quantum random walk algorithm the matrices produced as a result of qubit-qubit correlation are inherently complex, which can be an advantage for performing difficult tasks. For the same amount of time, the matrices the classical random walk algorithm can solve are limited to factorisable ones only.

We have tested the quantum random walk algorithm using software development kit Qiskit on IBM Q systems[43,44]. Numerical results show that this linear solver works on ideal quantum computer, and most importantly, also on noisy quantum computer having a short coherence time, provided the quantum circuit that encodes the **A** matrix is not too long. The limitation due to machine errors is discussed.

Results

We consider a system of linear equations of real numbers $\mathbf{A}\overrightarrow{x} = \overrightarrow{b}$, where **A** is a $N \times N$ matrix to be solved, $N \times 1$ vectors $\overrightarrow{x}$ and $\overrightarrow{b}$ are, respectively, the solution vector and a vector of constants. Without loss of generality, we rewrite **A** as

$$\mathbf{A} = \mathbf{1} - \gamma\mathbf{P}, \tag{1}$$

where **1** is the identity matrix, and $0 < \gamma < 1$ is a real number. We take **P** as a (stochastic) Markov-chain transition matrix, such that $P_{I,J} \geq 0$ and $\sum_J P_{I,J} = 1$, where $P_{I,J}$ refers to the **P** matrix element in the J-th column of the I-th row. This type of linear systems appears in value estimation for reinforcement learning[42,45,46], and radiosity equation in computer graphics[47]. In reinforcement learning algorithms, given a fixed policy of the learning agency, the vector $\overrightarrow{x}$ is the value function that determines the long-term cumulative reward, and efficient estimation of this function is key to successful learning[42]. Note that the matrix **A** given in Eq. (1) used as model Hamiltonian matrix belongs to the so-called stoquastic Hamiltonians[48,49].

To solve $\mathbf{A}\overrightarrow{x} = \overrightarrow{b}$, we expand the solution vector as Neumann series, that is, $\overrightarrow{x} = \mathbf{A}^{-1}\overrightarrow{b} = (\mathbf{1} - \gamma\mathbf{P})^{-1}\overrightarrow{b} = \sum_{s=0}^{\infty}\gamma^s\mathbf{P}^s\overrightarrow{b}$. Let us define the I_0 component of $\overrightarrow{x}$ truncated up to γ^c terms as

$$x_{I_0}^{(c)} = \sum_{s=0}^{c}\gamma^s \sum_{I_1,\ldots,I_s=0}^{N-1} P_{I_0,I_1}\ldots P_{I_{s-1},I_s}b_{I_s}. \tag{2}$$

This expression for $x_{I_0}^{(c)}$ can be evaluated by random walks on a graph of N nodes, with the probability of going from node I and node J of the graph given by the matrix element $P_{I,J}$, which we set as symmetric (undirected), namely $P_{I,J} = P_{J,I}$. An example of a four-node graph is shown in Fig. 1(a). By performing a series of random walks starting from node I_0, walking c steps according to the transition probability matrix **P**, and ending at some node I_c, Eq. (2) can be readily calculated to get the $x_{I_0}^{(c)}$ value, which is close to the solution x_{I_0} for some large c steps. Truncating the series introduces an error $\varepsilon \sim \mathcal{O}(\gamma^c)$. So, for a given γ, the number of steps necessary to meet a given tolerance ε is equal to $c \sim \log(1/\varepsilon)/\log(1/\gamma)$.

The above procedure can be extended to general matrices **A** by setting $\mathbf{A} = \mathbf{1} - \mathbf{B}$ where $B_{I,J} = P_{I,J}v_{I,J}$ for real matrix elements $v_{I,J}$ (see Methods 0.4). The calculation converges[50,51] provided that the spectral radius $\rho(\mathbf{B}^*) < 1$ where the matrix $\mathbf{B}^*$ is defined by $B^*_{I,J} = \frac{B^2_{I,J}}{P_{I,J}} = P_{I,J}v^2_{I,J}$. The matrices we here consider is a special case where $v_{I,J} = \gamma$ is a constant, and this simplification guarantees convergence of the calculation.

For classical Monte Carlo methods to compute Eq. (2), it takes $\mathcal{O}(N)$ time to calculate the cumulative distribution function that is used to determine the next walking step. So, these linear systems can be solved by classical Monte Carlo methods within $\mathcal{O}(N^2)$ time[52–56]. Similar Monte Carlo methods have been extended to more general matrices for applications in Green's function Monte Carlo method for many-body physics[57–59].

Encoding state spaces on Hamming cubes. As for material resources, in general it takes at least $\mathcal{O}(N)$ classical bits to store a row of a stochastic transition matrix $\mathbf{P}$ (or $\mathbf{A}$). However, for the classical and quantum random walks we here consider, it is possible to reduce significantly the number of classical or quantum bits necessary to encode the corresponding transition probability matrix $\mathbf{P}$ to $\mathcal{O}(\log(N))$ by introducing the Hamming cube (HC) structure[60]. To do it, we first associate each graph node with a bit string. As shown in Fig. 1(b), the four nodes of the $N = 4$ graph are fully represented by two bits. Node states $|0\rangle, |1\rangle, |2\rangle$, and $|3\rangle$ represent binary string states $|00\rangle, |01\rangle, |10\rangle$, and $|11\rangle$, respectively. For a N-node graph, only $\log_2(N) = n$ (to base 2) bits are needed to encode the integers $J \in \{0, 1, \ldots, N-1\}$, each representing the n-bit binary string state, namely $|J\rangle = |j_{n-1}, \ldots j_1, j_0\rangle$, where j_ℓ is 0 or 1.

Classical random walk. Before we introduce our quantum random walk algorithm, let us first consider classical random walks.

To perform random walks on a N-node graph, we use a simple coin-flipping process with $\mathcal{O}(\log(N))$ time steps. The ℓ-th bit flips with probability $\sin^2(\theta_\ell/2)$ or does not flip with probability $\cos^2(\theta_\ell/2)$, the total probability being equal to 1. The transition probability matrix elements are given by

$$P_{J',J}^{classical} = \prod_{\ell=0}^{n-1} \left|\cos^2\left(\frac{\theta_\ell}{2}\right)\right|^{1-i_\ell} \left|\sin^2\left(\frac{\theta_\ell}{2}\right)\right|^{i_\ell}, \tag{3}$$

where the n-bit binary string state $|I\rangle_c = |i_{n-1}, \ldots, i_1, i_0\rangle_c$ is determined by $|J'\rangle_c = |I\rangle_c \oplus |J\rangle_c$, where $\oplus$ denotes the bitwise exclusive or (XOR) operation, and the subscript c denotes classical states. The total number of $|\sin^2(\theta_\ell/2)|$, given by $d^{classical} = \sum_{\ell=0}^{n-1} i_\ell$, is the Hamming weight of $|I\rangle_c$, and so corresponds to the Hamming distance between $|J'\rangle_c$ and $|J\rangle_c$ states. This metric measures the number of steps that a walker needs to go from $|J\rangle_c$ to $|J'\rangle_c$ on the Hamming cube.

For the four-node graph shown in Fig. 1, the transition probability matrix $\mathbf{P}$ for classical random walks reads

$$\begin{aligned}
\mathbf{P}^{classical} &= \begin{bmatrix}
\cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \cos^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) \\
 & \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) & \cos^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) \\
 & & \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) \\
 & & & \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right)
\end{bmatrix} \\
&= \begin{bmatrix}
\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_1}{2}\right) \\
\sin^2\left(\frac{\theta_1}{2}\right) & \cos^2\left(\frac{\theta_1}{2}\right)
\end{bmatrix} \otimes \begin{bmatrix}
\cos^2\left(\frac{\theta_0}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right) \\
\sin^2\left(\frac{\theta_0}{2}\right) & \cos^2\left(\frac{\theta_0}{2}\right)
\end{bmatrix},
\end{aligned} \tag{4}$$

where $\otimes$ denotes the Kronecker product. The lower triangular part of the matrix is omitted due to symmetry. This simple case demonstrates a general feature for classical transition probability matrix $\mathbf{P}^{classical}$: the probability of flipping both bits is simply a product of the probabilities of flipping the 0-th bit and the 1-th bit in arbitrary order. For instance, $P_{0,3}^{classical} = P_{|00\rangle,|11\rangle}^{classical} = \sin^2(\theta_0/2)\sin^2(\theta_1/2) = P_{|00\rangle,|01\rangle}^{classical} P_{|00\rangle,|10\rangle}^{classical}$; similarly for the other $P_{I,J}^{classical}$'s. The fact that $\mathbf{P}^{classical}$ can be factorized into a Kronecker product of the matrices of each individual bit indicates that each bit flips independently, as for a Markovian process.

Quantum random walk. We can simulate quantum walks[61–67] on a N-node graph to obtain the solution vector $\overrightarrow{x}$ from Eq. (2). To do it, we use discrete-time coined quantum walk circuit[68,69]. The circuit for the four-node graph in Fig. 1 is shown in Fig. 2. The first two qubits j_0 and j_1 are state registers that will be initialized to encode the four-node graph, while the third qubit j_2 is the coin register.

To derive the quantum transition probability matrix on a graph of N nodes, we consider the state space of the $(n+1)$-qubit circuit as spanned by $\{|i_n\rangle_\diamond \otimes |i_{n-1}, \ldots, i_1, i_0\rangle_q\}$ with $n = \log_2(N)$: the $(n+1)$-th qubit registers the coin state $|i_n\rangle_\diamond$, and the other n qubits encode the N-node graph. We take the convention that the rightmost bit is i_0. Given a n-bit string $(j_{n-1}, \ldots, j_1, j_0)$, the initialized quantum state reads

$$\begin{aligned}
|\psi_{0,J}\rangle &= |0\rangle_\diamond \otimes |j_{n-1}, j_{n-2}, \ldots, j_2, j_1, j_0\rangle_q \\
&= |0\rangle_\diamond \otimes |J\rangle_q .
\end{aligned} \tag{5}$$

Next we let the $|\psi_{0,J}\rangle$ state evolve in random walk: in each walking step, we toss the coin by rotating the coin qubit, and then flip a graph qubit by applying the CNOT gate. This process is repeated on all the n qubits in the $|j_{n-1}, j_{n-2}, \ldots, j_2, j_1, j_0\rangle_q$ state, starting with the 0-th qubit. The corresponding evolution operator reads

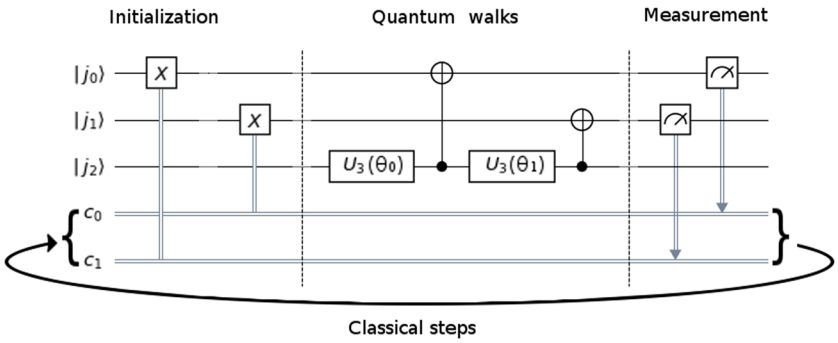

Figure 2. Discrete-time coined quantum walk circuit for the 4×4 transition matrix given in Eq. (10). Qubits j_0 and j_1 are state register qubits to represent the four-node graph in Fig. 1, first set as 0 before initialization, while the qubit j_2 is the coin register qubit. The measured registers c_0 and c_1 are fed back to initialize the next iteration. The classical-step is repeated c times to obtain the Neumann expansion up to order c.

$$\mathcal{U} = \prod_{k=0}^{n-1}{}' (|0\rangle_\Diamond\langle 0|_\Diamond \otimes 1_q + |1\rangle_\Diamond\langle 1|_\Diamond \otimes X_k) \cdot (U_3(\mathbf{u}_k) \otimes 1_q), \tag{6}$$

where the prime ($'$) on the Π denotes that the $k = 0$ operator applies first to the right, followed by the $k = 1$ operator, and so on; the 1_q operator is an identity map on the n-qubit state $|J\rangle_q$, X_k is a Pauli X gate (the Pauli matrix σ_x) that acts on the k-th qubit, and $U_3(\mathbf{u})$ is a single-qubit rotation operator

$$U_3(\mathbf{u}) = U_3(\theta, \phi, \lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{7}$$

that acts on the coin qubit state. Note that the first parentheses in Eq. (6) represents a CNOT gate. It is important to note that here we use *one* quantum coin only to decide on the Pauli X gate operation over all the n qubits, so the order of qubit operations plays a role in the determination of the transition probability matrix $\mathbf{P}$.

The first step is to project $\mathcal{U}$ on $|\psi_{0,J}\rangle$, which leads to

$$\mathcal{U}|\psi_{0,J}\rangle = \sum_{i_{n-1},\ldots,i_0=0}^{1} \prod_{\ell=0}^{n-1} U_3(\mathbf{u}_\ell)_{i_\ell, i_{\ell-1}} |i_{n-1}\rangle_\Diamond \otimes |i_{n-1} \oplus j_{n-1}, \ldots, i_1 \oplus j_1, i_0 \oplus j_0\rangle_q, \tag{8}$$

with $i_{-1} = 0$. By tracing out the coin degree of freedom, we obtain the reduced density matrix for the graph and hence the probability matrix $P_{J',J} = \langle J'|\mathrm{Tr}_\Diamond[\mathcal{U}|\psi_{0J}\rangle\langle\psi_{0J}|\mathcal{U}^\dagger]|J'\rangle$. The resulting quantum transition probability matrix elements then read

$$\begin{aligned} P^{quantum}_{J',J} &= \prod_{\ell=0}^{n-1} \left|U_3(\mathbf{u}_\ell)_{i_\ell, i_{\ell-1}}\right|^2 \\ &= \prod_{\ell=0}^{n-1} \left|\cos^2\left(\frac{\theta_\ell}{2}\right)\right|^{1-(i_\ell \oplus i_{\ell-1})} \left(\sin^2\left(\frac{\theta_\ell}{2}\right)\right)^{i_\ell \oplus i_{\ell-1}}, \end{aligned} \tag{9}$$

where $|I\rangle_q = |i_{n-1}, \ldots, i_1, i_0\rangle_q$ is determined by $|J'\rangle_q = |I\rangle_q \oplus |J\rangle_q$. For one $\mathcal{U}$ quantum evolution, the complex phase factors $e^{i\phi_\ell}$ and $e^{i\lambda_\ell}$ play no role. We will see later that these phases come into play in the case of multiple evolutions $\mathcal{U}^q$.

To understand the transition probability matrix produced by the quantum walk circuit (Fig. 2), let us again consider the four-node graph in Fig. 1, where

$$\mathbf{P}^{quantum} = \begin{bmatrix} \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) & \cos^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) \\ \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \cos^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) & \\ \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) & & \\ \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & & & \end{bmatrix}. \tag{10}$$

Unlike the above classical random walk, this matrix cannot be factorized into a Kronecker product of the matrices of each individual qubit. The probability of one qubit flipping depends on the other, indicating that the two qubits are correlated, or in quantum information theory entangled.

In comparison to Eq. (3) obtained from the classical random walk, we see that additional $\mathcal{O}(\log(N))$ XOR operations are required for classical computer to obtain the same quantum transition probability matrix, as can be seen from Eq. (9). In the case of $N = 4$, the classical and quantum transition probability matrices given by Eqs (4) and (10) are related by a permutation $\begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 2 & 1 \end{pmatrix}$. The quantum version of the Hamming distance between $|J\rangle_q$ and $|J'\rangle_q$ is given by $d^{quantum} = \sum_{\ell=0}^{n-1} i_\ell \oplus i_{\ell-1}$, which clearly shows the temporal correlation between the ℓ-th and $(\ell - 1)$-th qubits. We attribute this correlation to the fact that only *one* quantum coin is used to decide on the Pauli X gate over all the n qubits, thus creating some connection between qubits, and to the non-Markovian nature of quantum walk dynamics[70,71], in which the quantum circuit memorizes the qubit state $|i_{\ell-1}\rangle$ when it is walking in the direction that has the qubit state $|i_\ell\rangle$ in the Hamming cube.

It can be of interest to note that the circuit given in Eq. (6) is just one possible design leading to a particular correlation between qubits. In general, there are numerous ways to rearrange the walking steps to obtain different kinds of correlation, and it is possible to design the circuit for specific purposes. A simple way is to perform the walking steps in Eq. (6) in a reverse order, operating the $k = n - 1$ operator to the right first, followed by the $k = n - 2$ operator, and so on. This leads to a different metric $d^{quantum} = \sum_{\ell=0}^{n-1} i_{\ell+1} \oplus i_\ell$ with $i_N = 0$. It turns out that this $d^{quantum}$ corresponds to the Hamming distance in the Gray code representation.

The Gray code uses single-distance coding for integer sequence $0 \to 1 \to \cdots \to N - 1$, where adjacent integers differ by single bit flipping. In the case of the four-node graph in Fig. 1, the integers (0, 1, 2, 3) in the Gray code representation correspond to the $|00\rangle, |01\rangle, |11\rangle, |10\rangle$ states, respectively. It is obvious that this Gray code representation can be obtained from the natural binary code representation by a permutation $\begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 3 & 2 \end{pmatrix}$. There also exists a permutation that transforms $\mathbf{P}^{classical}$ to $\mathbf{P}^{quantum}$ in the Gray code basis. The proof of this correspondence for arbitrary N is given in Methods 0.1. Both the transform and inverse transform between the natural binary code and Gray code representations take $\mathcal{O}(\log(N))$ operations using classical computer[72]. This again shows that the quantum random walk algorithm gains $\mathcal{O}(\log(N))$ improvement over the classical one.

As the change of the Hamming distance for each walking step in the Gray code representation is $\delta d = 1$, a quantum walker in a geodesic of a Hamming cube automatically walks with the least action, that is, with the minimum change of the Hamming distance. This geodesic is a Hamiltonian path on hypercubes[73].

It is possible to increase the level of correlation in the probability matrix by performing multiple quantum evolutions, $\mathcal{U}^q$, where q is the number of quantum walk evolutions. The probability matrix produced by two quantum walk evolutions, $\mathcal{U}^2$, is given by (see Methods 0.2 for derivation)

$$P_{J',J}^{quantum} = \sum_{k=0}^{1} \left| \sum_I f(I, J' \oplus J \oplus I) \delta_{i_{n-1},k} \right|^2, \tag{11}$$

where, for $I = (i_{n-1}, \ldots, i_0)$ and $K = (k_{n-1}, \ldots, k_0)$,

$$\begin{aligned} f(I, K) \;=\; & [U_3(\mathbf{u}_{n-1})]_{i_{n-1},i_{n-2}} \cdots [U_3(\mathbf{u}_0)]_{i_0,k_{n-1}} \\ & \times [U_3(\mathbf{u}_{n-1})]_{k_{n-1},k_{n-2}} \cdots [U_3(\mathbf{u}_0)]_{k_0,0}, \end{aligned} \tag{12}$$

and

$$[U_3(\theta, \phi, \lambda)]_{\mu,\nu} = e^{i[\mu\phi+\nu\lambda]} (-1)^{(1-\mu)\nu} \left(\cos\left(\frac{\theta}{2}\right)\right)^{1-(\mu\oplus\nu)} \left(\sin\left(\frac{\theta}{2}\right)\right)^{\mu\oplus\nu}. \tag{13}$$

The fact that the summation over I in Eq. (11) runs over $\mathcal{O}(2^n)$ state configurations before the square is taken points to the complicated mixing of negative signs and complex phases ϕ_ℓ's and λ_ℓ's. The sign problem makes it difficult for pure classical Monte Carlo methods to simulate this transition.

In general, the dependence of the two-evolution quantum probability matrix on θ_ℓ's, ϕ_ℓ's and λ_ℓ's, is not trivial. Its explicit expression for the $N = 4$ graph is given in Methods 0.3. The phases ϕ_ℓ's and λ_ℓ's enter into play for graph sizes $N \geq 8$. On the other hand, the two-evolution probability matrix for classical random walk is given by

$$P_{J',J}^{classical} = \prod_{\ell=0}^{n-1} \left| \cos^4\left(\frac{\theta_\ell}{2}\right) + \sin^4\left(\frac{\theta_\ell}{2}\right) \right|^{1-i_\ell} \left| 2\cos^2\left(\frac{\theta_\ell}{2}\right) \sin^2\left(\frac{\theta_\ell}{2}\right) \right|^{i_\ell}, \tag{14}$$

which is still factorisable.

Numerical results. Figure 3 shows the performance of our hybrid quantum random walk algorithm on linear systems of dimension $N = 256$ and $N = 1024$. Their relative errors decrease with increasing sampling number. The relative error is defined as $\varepsilon = |x_I^{exact} - x_I|/|x_I^{exact}|$ for the I-th component of the solution vector $\vec{x}$, where $\vec{x}^{exact}$ is the exact result obtained with the NumPy package. To demonstrate, we use randomly generated vectors $\vec{b}$ and matrices $\mathbf{A}$ with a uniform distribution, $b_I \in [-1, 1]$ and $\theta_\ell \in [0, \pi]$. We choose γ and c such that the error introduced by the Neumann expansion is within $\mathcal{O}(10^{-4})$. See Table 1 for the relevant parameters of the

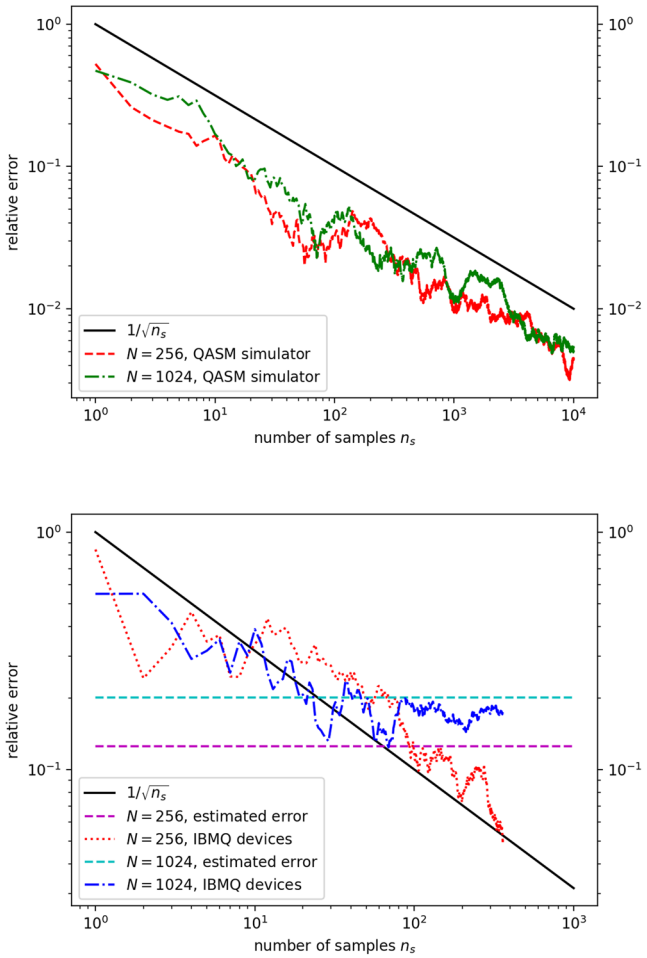

Figure 3. Relative errors $\varepsilon = |x_I^{exact} - x_I|/|x_I^{exact}|$ as a function of the sampling number n_s for $N = 256$ and $N = 1024$ matrices. The relevant parameters and estimated errors for these two matrices can be found in Table 1. Black solid lines represent the $1/\sqrt{n_s}$ error reduction expected for Monte Carlo calculations. (Upper figure) Red dashed line and green dash-dotted line are the results computed by the QASM simulator. (Lower figure) Blue dash-dotted line and red dotted line are data for the same matrices computed by the IBM Q 20 Tokyo machine or Poughkeepsie machine. Cyan and magenta horizontal dashed lines depict the estimated errors.

N	c	q	γ	Condition number	Estimated error ε_0
64	6	2	0.3	1.457	
128	6	2	0.3	1.599	
256	6	1	0.3	1.857	0.1255
1024	10	1	0.5	2.973	0.2010

Table 1. Relevant parameters for the matrices **A** of various sizes used for numerical experiments. Estimated error is defined in the text.

two matrices. The program is written and compiled with Qiskit version 0.7.2. The simulation results (upper figure) are obtained using QASM simulator[43], while the quantum machine results (lower figure) are obtained using IBM Q 20 Tokyo device or Poughkeepsie device[74,75].

The curves obtained by the QASM simulator are results averaged over ten runs. Their relative errors decrease as $1/\sqrt{n_s}$, where n_s is the number of random walk samplings. This $1/\sqrt{n_s}$ reduction is typical of Monte Carlo simulations, because the hybrid quantum walk algorithm has essentially the same structure as classical Monte Carlo methods. So, we do not gain any speedup in sampling number. Yet, this result substantiates the fact that our proposed algorithm works on ideal quantum computers.

For real IBM Q quantum devices, the accuracy stops improving after a certain number of samplings (see the plateau (blue dash-dotted curve) and oscillation (red dotted curve) in Fig. 3). This hardware limitation can be estimated using an error formula $\varepsilon_0 \sim \kappa \times E_r$, where κ is the condition number for the matrix **A** and E_r is the

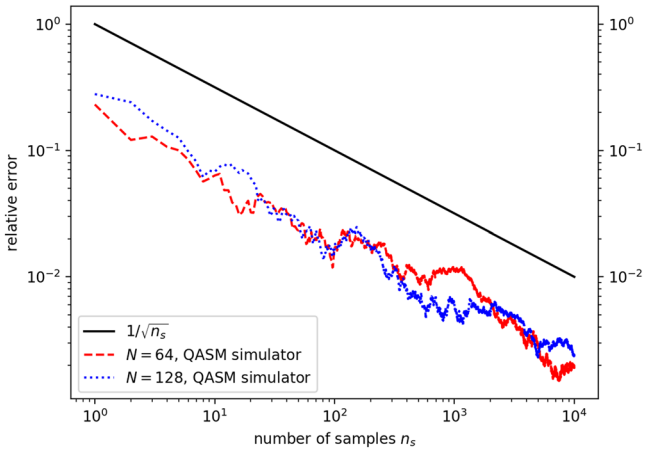

Figure 4. Relative errors $\varepsilon = |x_I^{exact} - x_I|/|x_I^{exact}|$ as a function of the sampling number n_s for $N = 64$ and $N = 128$ matrices, obtained by performing two quantum walk evolutions, $\mathcal{U}^2$. Black solid lines represent the $1/\sqrt{n_s}$ error reduction expected for Monte Carlo calculations. Red dashed line and blue dotted line are the results computed by the QASM simulator.

readout error of real machines. The condition number κ gauges the ratio of the relative error in the solution vector $\vec{x}$ to the relative error in the **A** matrix[3]: some perturbation in the matrix, $\mathbf{A} + \delta\mathbf{A}$, can cause an error in the solution vector, $\vec{x} + \delta\vec{x}$, such that $\|\delta\vec{x}\| \sim \kappa \times \|\delta\mathbf{A}\|$. By taking E_r as an estimate for $\|\delta\mathbf{A}\|$, we obtain the above error for the solution vector as $\varepsilon_0 = \|\delta\vec{x}\| \sim \kappa \times E_r$. The condition numbers given in Table 1 are computed by using Eq. (9) to construct the **A** matrices. For the average readout error of IBM Q 20 Tokyo device, we use $E_r = 6.76 \times 10^{-2}$[74]. The estimated errors ε_0 are given in Table 1. We see that the relative errors fall below the respective errors, indicating that the precision limit is due to the readout error of the current NISQ hardware. Note that the machines are calibrated several times during data collection, so the hardware error varies and the E_r value is only an estimate.

Figure 4 shows the results for linear systems of dimension $N = 64$ and $N = 128$, obtained by the QASM simulator that performs two quantum walk evolutions with uniformly distributed $(\theta_\ell, \phi_\ell, \lambda_\ell) \in [0, \pi]$. The relevant parameters for these two matrices are given in Table 1. The results again evidence that the algorithm works well, even in the presence of complex phases ϕ_ℓ's and λ_ℓ's. Note that we here take $(\theta_\ell, \phi_l, \lambda_l)$ as random variables to demonstrate the efficiency of our algorithm, but in real applications, these variables must be provided by other algorithms to generate a proper **P** matrix Fig. 4.

The communication latency between classical and quantum computer is the most time-consuming part, containing $\mathcal{O}(cn_s)$ communications. Fortunately, this number does not scale as N. For users with direct access to the quantum processors, communication bottleneck should be less severe.

Discussion

A comparison of computational resources is given in Table 2. For hybrid quantum walk algorithm, we need $1 + \log(N)$ qubits, $q\log(N)$ CNOT gates, and $q\log(N)$ U_3 gates, where q is the number of evolutions. The initialization takes $\log(N)$ X gates; but since they can be executed simultaneously, the initialization occupies one time slot only. Totally $1 + 2q\log(N)$ time slots are required for one quantum walk evolution to obtain one component of the solution vector $\vec{x}$. This can be an advantage when one is interested in partial information about $\vec{x}$. The same amount of time slots can be similarly derived for the classical random walk algorithm. Yet, we stress that these two algorithms deal with different transition probability matrices: factorisable matrices for classical random walk, and more complex correlated matrices for quantum random walk. The qubit-qubit correlation built into the correlated matrix can potentially be harnessed to perform complex tasks.

Other advantages of the algorithms we propose are:

(i) By restricting the matrices **A** to those that can be encoded in Hamming cubes, we can sample both classical and quantum random walk spaces that scale exponentially with the number of bits/qubits, and hence gain space complexity.
(ii) Classical Monte Carlo methods have time complexity of $\mathcal{O}(N)$ for general **P** matrices. For the matrices here considered, our algorithms have $\mathcal{O}(\log(N))$.
(iii) It is easier to access input and output than the HHL-type algorithm.
(iv) Random processes in a quantum computer are fundamental, and so are not plagued by various problems associated with pseudo-random number generators[76], like periods and unwanted correlations.
(v) Our quantum algorithm can run on noisy quantum computers whose coherence time is short.

We propose a hybrid quantum algorithm suitable for NISQ quantum computers to solve systems of linear equations. The solution vector $\vec{x}$ and constant vector $\vec{b}$ we consider here are classical data, so the input and

Algorithm	Time	Space for A	Input/Output
Classical Direct[2,3]	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	efficient for any **A**, $\vec{x}$, $\vec{b}$
Classical Iterative[2,3]	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	efficient for any **A**, $\vec{x}$, $\vec{b}$
Quantum HHL[4]	$\mathcal{O}(\log(N))$	$\mathcal{O}(\log(N))$ qubits	norm $\|\vec{x}\|$ not available difficult for **A**, $\vec{x}$, $\vec{b}$
Classical MC[45,53,55] (for one component x_l)	$\mathcal{O}(N)$	$\mathcal{O}(N)$	efficient for any $\vec{x}$, $\vec{b}$ limited **A** (stochastic **P**)
Classical RW on HC (for one component x_l)	$\mathcal{O}(\log(N))$	$\mathcal{O}(\log(N))$	efficient for any $\vec{x}$, $\vec{b}$ limited **A** (factorisable **P**)
Hybrid QW on HC (for one component x_l)	$\mathcal{O}(\log(N))$	$\mathcal{O}(\log(N))$ qubits	efficient for any $\vec{x}$, $\vec{b}$ limited **A** (correlated **P**)

Table 2. Comparison of various algorithms for solving $N \times N$ linear systems $\mathbf{A}\vec{x} = \vec{b}$, with respect to time and space complexities, and Input/Output issues. Note that for classical Monte Carlo (MC) method, classical random walk (RW) and hybrid quantum random walk (QW), the time complexities in the table are per sampling time. It takes $\mathcal{O}(cn_s)$ samplings to achieve the desired accuracy (see the text).

readout can be executed easily. Numerical simulations using IBM Q systems support the feasibility of this algorithm. We demonstrate that, by performing two quantum walk evolutions, the resulting probability matrix become more correlated in the parameter space. As long as the quantum circuit in this framework produces highly correlated probability matrix with a relatively short circuit depth, we can always gain quantum advantages over classical circuits.

Methods

Gray code basis. The natural binary code $B = (B_{n-1}, B_{n-2}, \ldots, B_1, B_0)$ is transformed to the Gray code basis[72] according to

$$g(B)_i = B_{i+1} \oplus B_i, \tag{A1}$$

$\forall\ i \in \{0, \ldots, n-1\}$ with $B_n = 0$. The probability matrix in the Gray code basis is given by

$$\begin{aligned} P^{quantum}_{I',J} &= \prod_{\ell=0}^{n-1} \left| U_3(\theta_\ell)_{i_\ell, i_{\ell+1}} \right|^2 \\ &= \prod_{\ell=0}^{n-1} \left| \cos^2\left(\frac{\theta_\ell}{2}\right) \right|^{1-(i_\ell \oplus i_{\ell+1})} \left| \sin^2\left(\frac{\theta_\ell}{2}\right) \right|^{i_\ell \oplus i_{\ell+1}} \end{aligned} \tag{A2}$$

with $i_N = 0$.

Lemma 1 *Let S_N be the set of all possible n-bit strings $\{(S_{n-1}, S_{n-2}, \ldots, S_1, S_0) | S_i \in \{0, 1\}\ \forall\ i \in \{0, 1, \ldots, n-1\}\}$ with $n = \log_2 N$, and π be a permutation of the set S_N. If there exists a function f: $S_N \mapsto \mathbb{R}$ such that for $A \in \mathbb{R}^{N \times N}$,*

$$A_{I \oplus J, J} = f(I) \tag{A3}$$

$\forall\ I, J \in S_N$, and if π is bitwise XOR homomorphic, then we have $A_{\pi(I \oplus J), \pi(J)} = f(\pi(I))$.

Proof 1 *Since π is bitwise XOR homomorphic*, Eq. (A.3) *leads to*

$$\begin{aligned} A_{\pi(I \oplus J), \pi(J)} &= A_{\pi(I) \oplus \pi(J), \pi(J)} \\ &= f(\pi(I)) \end{aligned} \tag{A4}$$

$\forall\ I, J \in S_N$.

Lemma 2 *Let $B \in S_N$ be represented by $(B_{n-1}, \ldots, B_0)$. Let g: $S_N \mapsto S_N$ be a function that transforms from natural bit string to Gray code according to $g(B)_i = B_{i+1} \oplus B_i$, $\forall\ i \in \{0, 1, \ldots, n-1\}$ with $B_n = 0$. Then g is a bitwise XOR homomorphism.*

Proof 2 *Let $I, J \in S_N$ be represented by bit strings $(I_{n-1}, \ldots, I_0)$ and $(J_{n-1}, \ldots, J_0)$, respectively. Using*

$$\begin{aligned} g(I)_i &= I_{i+1} \oplus I_i \\ g(J)_i &= J_{i+1} \oplus J_i \end{aligned} \tag{A5}$$

with $I_n = J_n = 0$, we get

$$\begin{aligned}[g(I)\oplus g(J)]_i &= g(I)_i \oplus g(J)_i \\ &= (I_{i+1}\oplus I_i)\oplus(J_{i+1}\oplus J_i) \\ &= (I_{i+1}\oplus J_{i+1})\oplus(I_i\oplus J_i) \\ &= g(I\oplus J)_i. \end{aligned} \tag{A6}$$

$\forall\ i \in 0,\ldots,n-1$.

Using **Lemma 1** and **Lemma 2**, the following theorem is clear.

Theorem 1 *There exists a permutation that maps the probability matrix produced by classical random walk to the probability matrix given in* Eq. (A.2) *produced by the quantum random walk circuit in a reverse order, that is, in Gray code basis.*

Derivation of Eq. (11). We use the evolution operator given in Eq. (6),

$$\mathcal{U} = \sum_{i_{n-1},\ldots,i_0,i_{-1}} \prod_{\ell=0}^{n-1} [U_3(\mathbf{u}_\ell)]_{i_\ell,i_{\ell-1}} (X_\ell)^{i_\ell} (|i_{n-1}\rangle_\diamond \langle i_{-1}|_\diamond) \tag{B1}$$

to compute the two-evolution operator

$$\mathcal{U}^2 = \sum_{\substack{i_{n-1},\ldots,i_0,i_{-1}\\ k_{n-1},\ldots,k_0,k_{-1}}} \prod_{\ell=0}^{n-1} [U_3(\mathbf{u}_\ell)]_{i_\ell,i_{\ell-1}} [U_3(\mathbf{u}_\ell)]_{k_\ell,k_{\ell-1}} (X_\ell)^{i_\ell+k_\ell} \delta_{i_{-1},k_{n-1}} (|i_{n-1}\rangle_\diamond \langle k_{-1}|_\diamond). \tag{B2}$$

Next we project the $\mathcal{U}^2$ operator on the $|\psi_{0J}\rangle$ state,

$$\begin{aligned}\mathcal{U}^2|\psi_{0J}\rangle &= \sum_{\substack{i_{n-1},\ldots,i_0,i_{-1}\\ k_{n-1},\ldots,k_0,k_{-1}}} \prod_{\ell=0}^{n-1} [U_3(\mathbf{u}_\ell)]_{i_\ell,i_{\ell-1}} [U_3(\mathbf{u}_\ell)]_{k_\ell,k_{\ell-1}} \delta_{i_{-1},k_{n-1}} \delta_{0,k_{-1}} |i_{n-1}\rangle_\diamond |I\oplus K\oplus J\rangle_q \\ &= \sum_{I,K} f(I,K)|i_{n-1}\rangle_\diamond |I\oplus K\oplus J\rangle_q, \end{aligned} \tag{B3}$$

where $f(I,K)$ is given in Eq. (12) and

$$|I\oplus K\oplus J\rangle_q = |i_{n-1}\oplus k_{n-1}\oplus j_{n-1},\ \ldots,\ i_0\oplus k_0\oplus j_0\rangle_q.$$

We then project $\mathcal{U}^2|\psi_{0J}\rangle$ on the final state $|k\rangle_\diamond |J'\rangle_q$

$$\begin{aligned}\langle J'|_q \langle k|_\diamond \mathcal{U}^2|\psi_{0J}\rangle &= \sum_{I,K} f(I,K)\delta_{k,i_{n-1}}\delta_{J',I\oplus K\oplus J} \\ &= \sum_I f(I, I\oplus J'\oplus J)\delta_{k,i_{n-1}}, \end{aligned} \tag{B4}$$

which leads to the probability matrix elements as

$$\begin{aligned}P_{J',J} &= \langle J'|\,\mathrm{Tr}_\diamond[\mathcal{U}^2|\psi_{0J}\rangle\langle\psi_{0J}|(\mathcal{U}^\dagger)^2]|J'\rangle \\ &= \sum_k |\langle J'|_q \langle k|_\diamond \mathcal{U}^2|\psi_{0J}\rangle|^2 \\ &= \sum_{k=0}^{1} \left|\sum_I f(I, J'\oplus J\oplus I)\delta_{i_{n-1},k}\right|^2. \end{aligned} \tag{B5}$$

Two-evolution quantum walk on $N=4$ graph. The probability matrix elements $P^{quantum}_{J',J}$ for two quantum evolutions $\mathcal{U}^2$ on the four-node graph read

$$\begin{aligned}P_{00} &= P_{11} = P_{22} = P_{33} \\ &= \frac{1}{4}\sin^2\theta_0 + \frac{1}{8}(1 + \cos^2\theta_1 + \cos^2\theta_0 + 4\cos\theta_1\cos\theta_0 \\ &\quad + \cos^2\theta_1\cos^2\theta_0 - \sin^2\theta_1\sin^2\theta_0), \end{aligned} \tag{C1}$$

$$P_{01} = P_{23} = \frac{1}{4}\sin^2\theta_1, \tag{C2}$$

$$P_{02} = P_{13} = \frac{1}{4}\sin^2\theta_1, \tag{C3}$$

$$P_{03} = P_{12} = \frac{1}{4}(1 - 2\cos\theta_1\cos\theta_0 + \cos^2\theta_1). \tag{C4}$$

Surprisingly, in this case the matrix elements do not depend on the (ϕ_0, ϕ_1) and (λ_0, λ_1) phases. However, the matrix elements do depend on complex phases when $N \geq 8$, as can be numerically checked. Note that $(P_{01}, P_{02}, P_{23}, P_{13})$ depend on θ_1 only: the destructive interference between configurations totally eliminates the θ_0 dependence, which is difficult to do by simple classical random walks.

Solving for general matrices. Here we discuss the applicability of our quantum random walk algorithm to general matrices[50,51,77,78]. Given an arbitrary matrix $\mathbf{A}$, we can obtain $\mathbf{B} = \mathbf{1} - \mathbf{A}$ and $B_{I,J} = P_{I,J}v_{I,J}$. Then the linear system $\mathbf{A}\vec{x} = \vec{b}$ can be solved by performing random walks according to the $P_{I,J}$ transition probabilities and by multiplying the factor $v_{I,J}$ at each walking step, provided that the linear solver converges to a solution. In classical random walk algorithms, it has been shown[50] that the convergence of the linear solver depends on the spectral radius $\rho(\mathbf{B}^*)$ of the matrix $\mathbf{B}^*$ where $B^*_{I,J} = B^2_{I,J}/P_{I,J} = P_{I,J}v^2_{I,J}$, that is, the necessary and sufficient condition for convergence is $\rho(\mathbf{B}^*) < 1$. We expect a similar condition for quantum random walk algorithms. However, one should consider the hybrid solver presented in this work as a special-purpose solver, in which the quantum circuit is designed for a specific matrix problem. The quantum circuits demonstrated in this work show that there are probability transition matrices that are easy to sample using quantum circuits but difficult using classical circuits. How to tailor a circuit design along with the relevant parameters suitable for the kind of application we are looking for is beyond the scope of this work.

Received: 20 June 2019; Accepted: 14 October 2019;
Published online: 07 November 2019

References

1. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th edn. (Cambridge University Press, New York, NY, USA, 2011).
2. Golub, G. H. & Van Loan, C. F. *Matrix Computations (3rd Ed.)*. (Johns Hopkins University Press, Baltimore, MD, USA, 1996).
3. Saad, Y. *Iterative Methods for Sparse Linear Systems*. 2nd edn. (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003).
4. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **103**, 150502, https://doi.org/10.1103/PhysRevLett.103.150502 (2009).
5. Clader, B. D., Jacobs, B. C. & Sprouse, C. R. Preconditioned quantum linear system algorithm. *Phys. Rev. Lett.* **110**, 250504, https://doi.org/10.1103/PhysRevLett.110.250504 (2013).
6. Montanaro, A. & Pallister, S. Quantum algorithms and the finite element method. *Phys. Rev. A* **93**, 032324, https://doi.org/10.1103/PhysRevA.93.032324 (2016).
7. Childs, A., Kothari, R. & Somma, R. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing* **46**, 1920–1950, https://doi.org/10.1137/16M1087072 (2017).
8. Costa, P. C. S., Jordan, S. & Ostrander, A. Quantum algorithm for simulating the wave equation. *Phys. Rev. A* **99**, 012323, https://doi.org/10.1103/PhysRevA.99.012323 (2019).
9. Berry, D. W., Childs, A. M., Ostrander, A. & Wang, G. Quantum algorithm for linear differential equations with exponentially improved dependence on precision. *Communications in Mathematical Physics* **356**, 1057–1081, https://doi.org/10.1007/s00220-017-3002-y (2017).
10. Dervovic, D. *et al*. Quantum linear systems algorithms: a primer. *arXiv e-prints* arXiv:1802.08227 (2018).
11. Wossnig, L., Zhao, Z. & Prakash, A. Quantum linear system algorithm for dense matrices. *Phys. Rev. Lett.* **120**, 050502, https://doi.org/10.1103/PhysRevLett.120.050502 (2018).
12. Biamonte, J. *et al*. Quantum machine learning. *Nature* **549**, 195–202, https://doi.org/10.1038/nature23474, 1611.09347 (2017).
13. Ciliberto, C. *et al*. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society of London Series A* **474**, 20170551, https://doi.org/10.1098/rspa.2017.0551, 1707.08561 (2018).
14. Deutsch, D. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* **400**, 97–117 (1985).
15. Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse ising model. *Phys. Rev. E* **58**, 5355–5363, https://doi.org/10.1103/PhysRevE.58.5355 (1998).
16. Farhi, E., Goldstone, J., Gutmann, S. & Sipser, M. Quantum Computation by Adiabatic Evolution. *eprint arXiv:quant-ph/0001106* (2000).
17. Aharonov, D. *et al*. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Review* **50**, 755–787, https://doi.org/10.1137/080734479 (2008).
18. O'Malley, D. & Vesselinov, V. V. Toq.jl: A high-level programming language for d-wave machines based on julia. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–7, https://doi.org/10.1109/HPEC.2016.7761616 (2016).
19. Borle, A. & Lomonaco, S. J. Analyzing the Quantum Annealing Approach for Solving Linear Least Squares Problems. *arXiv e-prints* arXiv:1809.07649 (2018).
20. Chang, C. C., Gambhir, A., Humble, T. S. & Sota, S. Quantum annealing for systems of polynomial equations. *Scientific Reports* **9**, https://doi.org/10.1038/s41598-019-46729-0 (2019).
21. Wen, J. *et al*. Experimental realization of quantum algorithms for a linear system inspired by adiabatic quantum computing. *Phys. Rev. A* **99**, 012320, https://doi.org/10.1103/PhysRevA.99.012320 (2019).
22. Subaşı, Y. B. U., Somma, R. D. & Orsucci, D. Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing. *Phys. Rev. Lett.* **122**, 060504, https://doi.org/10.1103/PhysRevLett.122.060504 (2019).
23. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* **2**, 79, https://doi.org/10.22331/q-2018-08-06-79 (2018).
24. Cao, Y., Daskin, A., Frankel, S. & Kais, S. Quantum circuit design for solving linear systems of equations. *Molecular Physics* **110**, 1675–1680, https://doi.org/10.1080/00268976.2012.668289 (2012).
25. Cai, X.-D. *et al*. Experimental quantum computing to solve systems of linear equations. *Phys. Rev. Lett.* **110**, 230501, https://doi.org/10.1103/PhysRevLett.110.230501 (2013).
26. Barz, S. *et al*. A two-qubit photonic quantum processor and its application to solving systems of linear equations. *Scientific Reports* **4**, 6115, https://doi.org/10.1038/srep06115, 1302.1210 (2014).

27. Pan, J. *et al*. Experimental realization of quantum algorithm for solving linear systems of equations. *Phys. Rev. A* **89**, 022313, https://doi.org/10.1103/PhysRevA.89.022313 (2014).
28. Zheng, Y. *et al*. Solving systems of linear equations with a superconducting quantum processor. *Phys. Rev. Lett.* **118**, 210504, https://doi.org/10.1103/PhysRevLett.118.210504 (2017).
29. Lee, Y., Joo, J. & Lee, S. Hybrid quantum linear equation algorithm and its experimental test on ibm quantum experience. *Scientific reports* **9**, 4778 (2019).
30. Aaronson, S. Read the fine print. *Nature Physics* **11**, 291–293, https://doi.org/10.1038/nphys3272 (2015).
31. Childs, A. M. Quantum algorithms: Equation solving by simulation. *Nature Physics* **5**, 861, https://doi.org/10.1038/nphys1473 (2009).
32. Möttönen, M., Vartiainen, J. J., Bergholm, V. & Salomaa, M. M. Quantum circuits for general multiqubit gates. *Phys. Rev. Lett.* **93**, 130502, https://doi.org/10.1103/PhysRevLett.93.130502 (2004).
33. Plesch, M. & Brukner, I. C. V. Quantum-state preparation with universal gate decompositions. *Phys. Rev. A* **83**, 032302, https://doi.org/10.1103/PhysRevA.83.032302 (2011).
34. Coles, P. J. *et al*. Quantum Algorithm Implementations for Beginners. *arXiv e-prints* arXiv:1804.03719 (2018).
35. James, D. F. V., Kwiat, P. G., Munro, W. J. & White, A. G. Measurement of qubits. *Phys. Rev. A* **64**, 052312, https://doi.org/10.1103/PhysRevA.64.052312 (2001).
36. Suess, D., Rudnicki, Ł., Maciel, T. O. & Gross, D. Error regions in quantum state tomography: computational complexity caused by geometry of quantum states. *New Journal of Physics* **19**, 093013, https://doi.org/10.1088/1367-2630/aa7ce9 (2017).
37. Cramer, M. *et al*. Efficient quantum state tomography. *Nature Communications* **1**, 149, https://doi.org/10.1038/ncomms1147, 1101.4366 (2010).
38. Peruzzo, A. *et al*. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications* **5**, 4213, https://doi.org/10.1038/ncomms5213, 1304.3061 (2014).
39. Wecker, D., Hastings, M. B. & Troyer, M. Progress towards practical quantum variational algorithms. *Phys. Rev. A* **92**, 042303, https://doi.org/10.1103/PhysRevA.92.042303 (2015).
40. McClean, J. R., Romero, J., Babbush, R. & Aspuru-Guzik, A. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics* **18**, 023023, https://doi.org/10.1088/1367-2630/18/2/023023, 1509.04279 (2016).
41. Kandala, A. *et al*. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* **549**, 242–246, https://doi.org/10.1038/nature23879, 1704.05018 (2017).
42. Sutton, R. S. & Barto, A. G. *Introduction to Reinforcement Learning*. 1st edn. (MIT Press, Cambridge, MA, USA, 1998).
43. Aleksandrowicz, G. *et al*. Qiskit: An open-source framework for quantum computing, https://doi.org/10.5281/zenodo.2562110 (2019).
44. IBM Q Experience, https://quantumexperience.ng.bluemix.net, Accessed: 12/01/2018 (2016).
45. Barto, A. & Duff, M. Monte carlo matrix inversion and reinforcement learning. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, 687–694 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993).
46. Paparo, G. D., Dunjko, V., Makmal, A., Martin-Delgado, M. A. & Briegel, H. J. Quantum speedup for active learning agents. *Phys. Rev. X* **4**, 031002, https://doi.org/10.1103/PhysRevX.4.031002 (2014).
47. Goral, C. M., Torrance, K. E., Greenberg, D. P. & Battaile, B. Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.* **18**, 213–222, https://doi.org/10.1145/964965.808601 (1984).
48. Bravyi, S., Divincenzo, D. P., Oliveira, R. & Terhal, B. M. The complexity of stoquastic local hamiltonian problems. *Quantum Info. Comput.* **8**, 361–385 (2008).
49. Bravyi, S. Monte carlo simulation of stoquastic hamiltonians. *Quantum Info. Comput.* **15**, 1122–1140 (2015).
50. Ji, H., Mascagni, M. & Li, Y. Convergence analysis of markov chain monte carlo linear solvers using ulam-von neumann algorithm. *SIAM Journal on Numerical Analysis* **51**, 2107–2122 (2013).
51. Dimov, I. T. & McKee, S. *Monte Carlo Methods for Applied Scientists* (World Scientific Press, 2004).
52. Metropolis, N. & Ulam, S. The monte carlo method. *Journal of the American Statistical Association* **44**, 335–341, https://doi.org/10.1080/01621459.1949.10483310, PMID: 18139350 (1949).
53. Forsythe, G. E. & Leibler, R. A. Matrix inversion by a monte carlo method. *Mathematics of Computation* **4**, 127–129 (1950).
54. Wasow, W. R. A note on the inversion of matrices by random walks. *Mathematical Tables and Other Aids to Computation* **6**, 78–81 (1952).
55. Lu, F. & Schuurmans, D. Monte carlo matrix inversion policy evaluation. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, 386–393 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003).
56. Branford, S. *et al*. Monte carlo methods for matrix computations on the grid. *Future Generation Computer Systems* **24**, 605–612, https://doi.org/10.1016/j.future.2007.07.006 (2008).
57. Ceperley, D. M. & Alder, B. J. Ground state of the electron gas by a stochastic method. *Phys. Rev. Lett.* **45**, 566–569, https://doi.org/10.1103/PhysRevLett.45.566 (1980).
58. Negele, J. W. & Orland, H. *Quantum many-particle physics* (Addison-Wesley, 1988).
59. Landau, D. & Binder, K. *A Guide to Monte Carlo Simulations in Statistical Physics*. (Cambridge University Press, New York, NY, USA, 2005).
60. Hamming, R. W. Error detecting and error correcting codes. *The Bell System Technical Journal* **29**, 147–160, https://doi.org/10.1002/j.1538-7305.1950.tb00463.x (1950).
61. Aharonov, Y., Davidovich, L. & Zagury, N. Quantum random walks. *Phys. Rev. A* **48**, 1687–1690, https://doi.org/10.1103/PhysRevA.48.1687 (1993).
62. Childs, A. M., Farhi, E. & Gutmann, S. An example of the difference between quantum and classical random walks. *eprint arXiv:quant-ph/0103020* (2001).
63. Aharonov, D., Ambainis, A., Kempe, J. & Vazirani, U. Quantum walks on graphs. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, 50–59, https://doi.org/10.1145/380752.380758 (ACM, New York, NY, USA, 2001).
64. Moore, C. & Russell, A. Quantum walks on the hypercube. In Rolim, J. D. P. & Vadhan, S. (eds) *Randomization and Approximation Techniques in Computer Science*, 164–178 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2002).
65. Szegedy, M. Quantum speed-up of markov chain based algorithms. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, 32–41, https://doi.org/10.1109/FOCS.2004.53 (IEEE Computer Society, Washington, DC, USA, 2004).
66. Kendon, V. M. A random walk approach to quantum algorithms. *Philosophical Transactions of the Royal Society of London Series A* **364**, 3407–3422, https://doi.org/10.1098/rsta.2006.1901, quant-ph/0609035 (2006).
67. Childs, A. *Lecture notes on quantum algorithms* (2017).
68. Košk, J. & Bužek, V. Scattering model for quantum random walks on a hypercube. *Phys. Rev. A* **71**, 012306, https://doi.org/10.1103/PhysRevA.71.012306 (2005).
69. Shikano, Y. & Katsura, H. Localization and fractality in inhomogeneous quantum walks with self-duality. *Phys. Rev. E* **82**, 031122, https://doi.org/10.1103/PhysRevE.82.031122 (2010).
70. Breuer, H.-P., Laine, E.-M., Piilo, J. & Vacchini, B. Colloquium: Non-markovian dynamics in open quantum systems. *Rev. Mod. Phys.* **88**, 021002, https://doi.org/10.1103/RevModPhys.88.021002 (2016).

71. de Vega, I. & Alonso, D. Dynamics of non-markovian open quantum systems. *Rev. Mod. Phys.* **89**, 015001, https://doi.org/10.1103/RevModPhys.89.015001 (2017).
72. Knuth, D. E. *The Art of Computer Programming*, *Volume 4*, *Fascicle 2*: *Generating All Tuples and Permutations* (*Art of Computer Programming*) (Addison-Wesley Professional, 2005).
73. Gilbert, E. N. Gray codes and paths on the n-cube. *The Bell System Technical Journal* **37**, 815–826, https://doi.org/10.1002/j.1538-7305.1958.tb03887.x (1958).
74. IBM Q devices and simulators, https://www.research.ibm.com/ibm-q/technology/devices/, Accessed: 2019-02-20 (2019).
75. Cramming More Power Into a Quantum Device, https://www.ibm.com/blogs/research/2019/03/power-quantum-device/, Accessed: 2019-03-21 (2019).
76. Srinivasan, A., Mascagni, M. & Ceperley, D. Testing parallel random number generators. *Parallel Computing* **29**, 69–94, https://doi.org/10.1016/S0167-8191(02)00163-1 (2003).
77. Dimov, I., Dimov, T. & Gurov, T. A new iterative monte carlo approach for inverse matrix problem. *Journal of Computational and Applied Mathematics* **92**, 15–35, https://doi.org/10.1016/S0377-0427(98)00043-0 (1998).
78. Halton, J. H. Sequential monte carlo techniques for the solution of linear systems. *J. Sci. Comput.* **9**, 213–257, https://doi.org/10.1007/BF01578388 (1994).

Acknowledgements

We thank Chia-Cheng Chang, Yu-Cheng Su, Rudy Raymond, and Tomah Sogabe for discussions. Access to IBM Q systems is provided by IBM Q Hub at National Taiwan University. This work is supported in part by Ministry of Science and Technology, Taiwan, under grant No. MOST 107-2627-E-002 -001 -MY3, MOST 106-2221-E-002 -164 -MY3, and MOST 108-2628-E-002 -010 -MY3.

Author contributions

C.-C.C. performed the calculations and simulations. S.-Y.S. provided theoretical support and revised the manuscript. M.-F.W. provided technical support. Y.-R.W. accessed IBM Q systems. All authors discussed and wrote the paper.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to C.-C.C. or Y.-R.W.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.