IEEE Transactions on Quantum Engineering

Received 24 May 2025; revised 29 September 2025; accepted 18 October 2025; date of publication 23 October 2025; date of current version 1 December 2025.

Digital Object Identifier 10.1109/TQE.2025.3624699

Binary Tree Block Encoding of Classical Matrix

ZEXIAN LI[1], XIAO-MING ZHANG[2,3], CHUNLIN YANG[4], AND GUOFENG ZHANG[1,5] (Senior Member, IEEE)

[1]Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong, China
[2]Key Laboratory of Atomic and Subatomic Structure and Quantum Control (Ministry of Education), South China Normal University, Guangzhou 510000, China
[3]Guangdong Provincial Key Laboratory of Quantum Engineering and Quantum Materials, Guangdong-Hong Kong Joint Laboratory of Quantum Matter, South China Normal University, Guangzhou 510000, China
[4]School of Mathematical Sciences, Harbin Engineering University, Harbin 150001, China
[5]Hong Kong Polytechnic University Shenzhen Research Institute, Shenzhen 518057, China

Corresponding authors: Xiao-Ming Zhang; Guofeng Zhang (e-mail: phyxmz@gmail.com; guofeng.zhang@polyu.edu.hk).

This work was supported in part by the Quantum Science and Technology-National Science and Technology Major Project under Grant 2023ZD0300600, in part by the Guangdong Provincial Quantum Science Strategic Initiative under Grant GDZX2200001, in part by the Hong Kong Research Grant Council under Grant 15213924, and in part by the National Natural Science Foundation of China under Grant 62173288, Grant 62473117, and Grant 12405013.

ABSTRACT State preparation and block encoding are essential subroutines in quantum computing. The former provides basic encoding of quantum states, while the latter transforms classical data into a matrix representation within a quantum circuit. Some quantum advantages are built on the assumption that the block-encoding subroutine has been compiled in the quantum circuit, and this derives a problem of how to efficiently compile a block encoding. The resource tradeoffs of block encoding, such as circuit size, subnormalization factor, compilation complexity (both time and space), and robustness against errors, are central to its efficiency. In this work, the binary tree block-encoding (`BITBLE`) protocol is introduced, which optimizes these tradeoffs. For a classical matrix in $\mathbb{C}^{2^n\times 2^n}$, our approach reduces the compilation time to $\mathcal{O}(n2^{2n})$ using n ancilla qubits, achieving superior resource tradeoffs compared to existing methods. Numerical experiments further reveal that the approach outlined in `BITBLE` enhances compilation efficiency, resource scalability, and robustness against single-qubit gate errors in various standard data encoding tasks. Moreover, all algorithms are available as open source.

INDEX TERMS Circuit size, quantum circuit, quantum compiling, state preparation, unitary synthesis.

I. INTRODUCTION

Quantum signal processing (QSP) [1] and quantum singular value transformation (QSVT) [2], [3] are powerful frameworks for solving high-dimensional eigenvalue and singular value problems. Under this framework, eigenvalue or singular value problems can be solved without directly solving the eigenvalue decomposition and singular value decomposition, providing an alternative way to solve numerical problems involving eigenvalues or singular values. The input model employed in these methods is based on block encoding [2]. Leveraging the block-encoding framework, several seminal quantum algorithms—such as Hamiltonian simulation [1], [4], Grover, quantum Fourier transform, and Harrow–Hassidim–Lloyd [5], [6]—can be interpreted as instances of QSVT algorithms [3]. The embedding of a matrix A is typically realized as the top-left block of a unitary matrix U, expressed as

$$U = \begin{bmatrix} A & * \\ & * \end{bmatrix}$$

where $*$ denotes other blocks in the unitary matrix U and the embedding satisfies $\|A\|_2 \leq 1$. Nevertheless, the exponential scaling of time and memory resources required for the compilation with respect to the number of qubits n presents a formidable challenge in the compilation of quantum circuits for QSP or QSVT algorithms. Should the compilation of the quantum circuit for block encoding prove to be more complex than resolving the original problems, the purported quantum advantage attributed to this input model may be negated. This suggests that the complexity of compilation is

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see http://creativecommons.org/licenses/by/4.0/

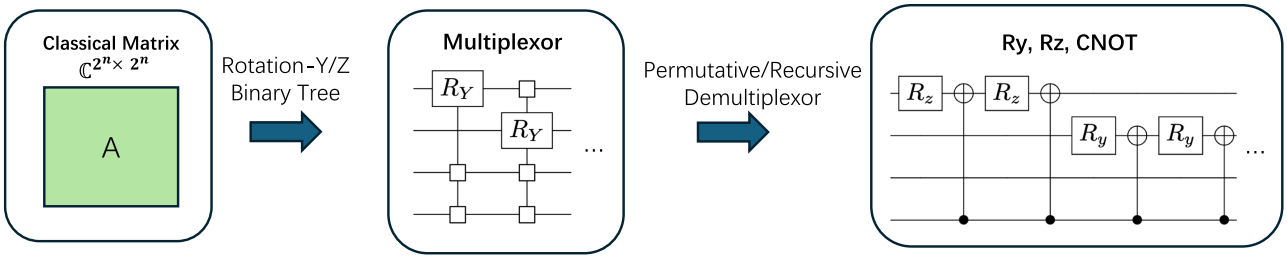


FIG. 1. Algorithm process of the BITBLE protocol. Step 1: Generate multiplexor from the matrix data based on the rotation-*Y*/*Z* binary tree data structure. Step 2: Decoupling multiplexor into single-qubit gates (Rotation-*Y*/*Z* gate) and CNOT gate using permutative or recursive demultiplexor method.

TABLE 1. Gray Permutation Mapping P_G From 3-Bit Binary Codes $b_0 - B_7$ to 3-Bit Binary Reflected Gray Codes $g_0 - G_7$

	bit3	bit2	bit1
b_0	0	0	0
b_1	0	0	1
b_2	0	1	0
b_3	0	1	1
b_4	1	0	0
b_5	1	0	1
b_6	1	1	0
b_7	1	1	1

$\xrightarrow{P_G}$

	bit3	bit2	bit1
g_0	0	0	0
g_1	0	0	1
g_2	0	1	1
g_3	0	1	0
g_4	1	1	0
g_5	1	1	1
g_6	1	0	1
g_7	1	0	0

a significant challenge to achieving comprehensive quantum advantage.

Within the domain of noisy intermediate-scale quantum architectures [7], practical quantum algorithms are limited by qubits, circuit depth, and related parameters. Similar limitations are also applied to the early-fault-tolerant setting [8]. As of 2024 [9], IBM has successfully constructed a fault-tolerant quantum memory comprising 12 logical qubits. In 2025, a bicycle architecture [10] has been proposed, which significantly reduces qubit requirements for quantum computers executing specified algorithms and thus is a promising fault-tolerant architecture. According to IBM's outlined future developments [11], the construction of a fault-tolerant quantum computer with 200 logical qubits is anticipated in 2029.

The task of block encoding also faces the aforementioned requirements. To solve the aforementioned challenges, our work bridges this gap by introducing a binary tree data structure and demultiplexor methods for block encoding and state preparation, which enable highly flexible resource tradeoffs. Our main contributions are summarized as follows: We have proposed three binary tree block-encoding protocols (BITBLE[1]–BITBLE[3]; see Table 3). Our binary tree block-encoding protocol (see Fig. 1) significantly reduces compilation time, decreases gate counts, and demonstrates greater robustness to single-qubit gate cutoff errors compared to current fast-compilation methods. Specifically, the BITBLE[1] and BITBLE[2] protocols reduce the ancilla requirement from the comparable protocol [12] (n versus $n+1$). BITBLE[3], by contrast, uses more ancillas than the BITBLE[1]–BITBLE[2] ($n+2$ versus n) but trades those qubits for a smaller error and often a lower CNOT "*size metric*" [as (1)] for some cases. To the best of our knowledge, our block-encoding protocol approach achieves the fastest known compilation efficiency (as shown in Table 4).

A. RELATED WORK

Block encoding has garnered increasing research interest, with applications spanning a wide range of matrices from structured special matrices to general unstructured ones. Special matrices, such as the density operator [13], [14], kernel matrices [15], pseudodifferential operators [16], Hamiltonians derived via Schrodingerization [17], discretized elliptic operators [18], and pairing Hamiltonians in nuclear physics [19], have all been block encoded efficiently. For encoding sparse matrices, Gilyén et al. [2] proposed an efficient implementation, including a preamplified version. Camps et al. [20] further presented the explicit construction for the block encoding of sparse matrices. When a sparse matrix processes an arithmetic structure, Sünderhauf et al. [21] proposed a novel block-encoding protocol and enhanced it using preamplification and state preparation techniques. Based on a dictionary data structure, Yang et al. [22] developed an efficient dictionary-based block encoding that achieves low subnormalization and circuit depth.

For general dense data matrices of size $2^n \times 2^n$, the block-encoding protocol in [23] provides a procedure for converting classical data into block encoding using a quantum random access memory model [24]. Building on this procedure, block-encoding protocols with near-optimal gate complexities have been extensively discussed [25]. However, these protocols require at least $\mathcal{O}(2^n/n)$ ancillary qubits [26], making it infeasible to compile large-scale block encodings. Camps and van Beeumen [12] proposed a fast approximate quantum circuit for block encodings (FABLE), designed for compilation within QSP or QSVT. This method characterizes matrices in the Walsh–Hadamard domain and can be modified to accommodate highly compressible circuits to block encode a certain subset of sparse matrices [27]. However, FABLE has a large subnormalization factor proportional to 2^n, leading to a significant computational overhead when implemented in a quantum algorithm. Furthermore, existing research has largely overlooked the analysis of tradeoffs associated with varying resource requirements, including circuit

IEEE Transactions on Quantum Engineering

TABLE 2. Comparison of Two Methods for Decoupling an *n*-Control Multiplexor

Methods	CNOT Count	Operation Count	Max Size of Walsh-Hadamard Transform
Permutative demultiplexor	2^n	$\mathcal{O}(n2^n)$	$2^n \times 2^n$
Recursive demultiplexor	$(2^{\lfloor n/2 \rfloor} + 1) \cdot 2^{\lceil n/2 \rceil}$	$\mathcal{O}(n2^n)$	$2^{\lceil n/2 \rceil} \times 2^{\lceil n/2 \rceil}$

The recursive demultiplexor method decomposes the operation into a series of smaller Walsh-Hadamard transforms, at the cost of additional CNOT gates.

TABLE 3. Comparison of Block-Encoding Protocol for a Matrix $A \in \mathbb{C}^{2^n \times 2^n}$

Protocols	Subnormalization	Decoupling Methods	Ancilla
BITBLE[1]	$\|A\|_F$	Recursive	n
BITBLE[2]	$\|A\|_F$	Permutative	n
BITBLE[3]	$\mu_p(A)$	Recursive	$n+2$
FABLE [12]	$2^n \max_{i,j} \|A_{i,j}\|$	Permutative	$n+1$

It can be observed that $\max\{\|A\|_F, \mu_p(A)\} \leq 2^n \max_{i,j} |A_{i,j}|$.

size, subnormalization factors, and the complexity of compilation in terms of both temporal and spatial dimensions.

B. NOTATION

In the rest of this article, we assume that the size of the matrix is $N \times N$ with $N = 2^n$. The notation $\beta_{:,k}$, $\beta_{k,:}$, and $\beta_{:}$ stand for vectors as $\beta_{:,k} \equiv (\beta_{1,k}, \beta_{2,k}, \ldots)^T$, $\beta_{k,:} \equiv (\beta_{k,1}, \beta_{k,2}, \ldots)$, and $\beta_{:} \equiv (\beta_1, \beta_2, \ldots)$, respectively. The symbol $(\beta_{1,k}, \beta_{2,k}, \ldots)^T$ stands for the transport of a row vector $(\beta_{1,k}, \beta_{2,k}, \ldots)$, and $(\beta_{1,k}, \beta_{2,k}, \ldots)^*$ stands for the conjugate transport of that vector. The symbol $U^\dagger$ stands for the conjugate transport of the unitary matrix U. For $j = j_n \times 2^{n-1} + \cdots + j_1 \times 2^0 \in \{0, 1, \ldots, 2^n - 1\}$, the quantum state is represented as $|j\rangle \equiv |j_n\rangle \otimes |j_{n-1}\rangle \otimes \cdots \otimes |j_1\rangle$.

C. ORGANIZATION

The rest of this article is organized as follows. Section I discusses the motivation and results of our proposed block-encoding protocols, along with relevant notation. Section II presents the definition and implementation of state preparation and block encoding. Section III presents two decomposition methods for multiplexor operations: permutative demultiplexor method and recursive demultiplexor method. Section IV introduces the BITBLE protocols and parameter-finding methods, including the computational complexity for circuit compilation, and the approximation error of the block-encoding protocol. Section V provides numerical results for circuit compilation using various examples. Finally, Section VI concludes this article. MATLAB implementations of the BITBLE protocols, developed using QCLAB [28], are publicly available online.[1]

II. STATE PREPARATION AND BLOCK ENCODINGS

Quantum state preparation generates the target state $|\psi\rangle$ from an initial state $|0\rangle^{\otimes n}$ using single- and two-qubit gates, where

$$|\psi\rangle = \sum_{k=0}^{N-1} e^{i\phi_k} |\psi_k|\, |k\rangle$$

where $N = 2^n$, $\psi_k \in \mathbb{C}$, $\sum_{k=0}^{N-1} |\psi_k|^2 = 1$, and $|k\rangle \equiv |k_n k_{n-1} \cdots k_1\rangle$ represent the computational basis vectors with bits k_j for $j = 1, 2, \ldots, n$.

Block encoding transforms classical data into a matrix representation within a quantum circuit. The formal definition is given as follows.

Definition 1 (See [2]): Let $a, n, \tilde{n} \in \mathbb{N}$ with $\tilde{n} = a + n$. Then, a unitary matrix U size of $\tilde{n}$-qubit is an (α, a, ε)-block encoding of an operator A size of n-qubit, if

$$\|A - \alpha \left(\langle 0|^{\otimes a} \otimes I_n\right) U \left(|0\rangle^{\otimes a} \otimes I_n\right)\| \leq \varepsilon.$$

The subnormalization factor α of block encoding is a crucial parameter that influences the circuit depth in quantum algorithms utilizing QSP [1] and QSVT [3], [12]. To quantify the number of operations per circuit, we consider the "*metric*" of single-qubit rotations and two-qubit CNOT gates in different protocols as

$$\text{size metric} = \text{gate count} \times \text{subnormalization}. \quad (1)$$

Unless otherwise specified, the gate count refers to the total number of CNOT gates. Sometimes, CNOT gates exhibit significantly different properties compared to single-qubit gates, so their respective size metrics are also considered separately. Note that this metric is proportional to the subnormalization factor. A special case of the aforementioned figure of the metric, in terms of T-gate counts, has been proposed by Sünderhauf et al. [21]. In contrast, in this article, we only consider the gate sets comprising CNOT gate and single-qubit gates.

Given a general classical data $A \in \mathbb{R}^{N \times N}$, the first implementation for the data structure with Frobenius norm $\|A\|_F$ for a matrix is studied by Kerenidis and Prakash [29]. The idea of quantum state representations is generalized to an alternative way [30], [31], and by that idea, the subnormalization of this block encoding is $\mu_p(A) = \sqrt{S_{2p}(A) S_{2(1-p)}(A^T)}$, where $S_q(A) = \max_{i \in N} \|A_{i,:}\|_q^q$. For some sparse matrices with s^r, s^c row and column-sparse matrix given in the sparse-access input model, subnormalization can be $\sqrt{s^r s^c}$ [2]. For an $N \times N$ random matrix, the aforementioned subnormalization is proportional to $O(N)$, which is a square improvement from other protocols. The block-encoding protocols proposed in this article are based on the implementation of multiplexor operation with subnormalization of $\|A\|_F$ and $\mu_p(A)$.

The multiplexor [32], also known as the uniformly controlled rotation, is denoted by *controlled-*$R_\alpha^{[\beta_j]_{j=0}^{2^n-1}}$. If the control nodes are placed at the top of a circuit, then it can be

[1] https://github.com/zexianLIPolyU/BITBLE

TABLE 4. Time of Circuit Synthesis for Random Complex Matrices of Size $2^n \times 2^n$ (Corresponding to n Qubits) With 90% Confidence Intervals

n	BITBLE[1]	BITBLE[2]	BITBLE[3]	FABLE [12]	Qiskit
5	0.010 ± 0.003	0.012 ± 0.003	0.017 ± 0.004	$\mathbf{0.009 \pm 0.001}$	1.763 ± 0.627
6	$\mathbf{0.029 \pm 0.002}$	0.044 ± 0.008	0.069 ± 0.015	0.040 ± 0.007	6.688 ± 0.583
7	$\mathbf{0.122 \pm 0.012}$	0.182 ± 0.036	0.237 ± 0.022	0.177 ± 0.025	34.525 ± 5.853
8	$\mathbf{0.632 \pm 0.097}$	0.802 ± 0.142	1.057 ± 0.139	0.832 ± 0.151	172.2 ± 12.6
9	$\mathbf{2.304 \pm 0.303}$	3.410 ± 0.576	4.420 ± 0.516	3.271 ± 0.520	763.1 ± 28.5
10	$\mathbf{9.848 \pm 1.433}$	14.08 ± 2.186	19.00 ± 2.488	14.01 ± 2.13	-
11	$\mathbf{42.62 \pm 6.38}$	62.23 ± 9.96	83.21 ± 11.77	60.34 ± 9.21	-
12	$\mathbf{171.8 \pm 23.5}$	249.7 ± 34.8	358.3 ± 41.6	255.0 ± 32.4	-
13	$\mathbf{761.9 \pm 95.0}$	1103.0 ± 142.5	1483.4 ± 163.0	1087.4 ± 137.7	-
14	$\mathbf{2800.0 \pm 617.9}$	4251.5 ± 981.0	5773.7 ± 1012.2	4181.9 ± 1063.4	-

BITBLE[1] demonstrates significantly faster circuit compilation compared to both FABLE and Qiskit unitary synthesis. Notably, Qiskit's unitary synthesis encounters memory limitations when processing random matrices larger than $2^{10} \times 2^{10}$.

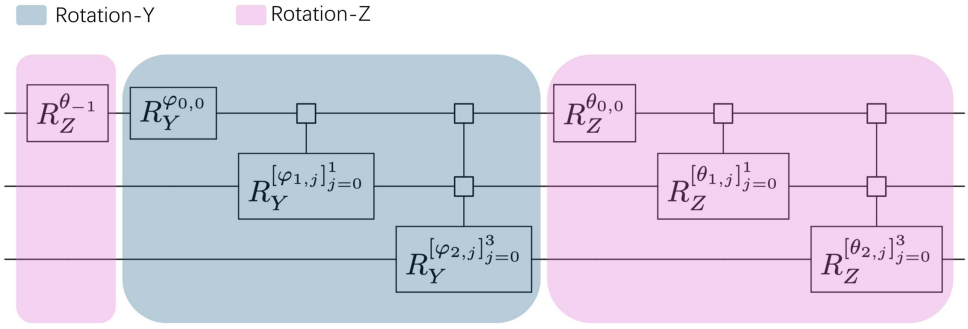


FIG. 2. Quantum circuit of state preparation for state in $\mathbb{C}^{2^n}$ with $n = 3$ using multiplexor operations, where the rotation parameters φ/θ are computed by the Rotation-Y/Z binary tree. If a quantum state is in $\mathbb{R}^{2^n}$, it can be prepared by a circuit composed solely of Y-rotation gates.

represented in a mathematical form as

$$\sum_{j=0}^{2^n-1} |j\rangle\langle j| \otimes R_\alpha^{\beta_j} = \begin{pmatrix} R_\alpha^{\beta_0} & & \\ & \ddots & \\ & & R_\alpha^{\beta_{2^n-1}} \end{pmatrix} \tag{2}$$

where the rotation matrix $R_\alpha^\beta \in \mathbb{C}^{2\times 2}$ is given by

$$R_\alpha^\beta = e^{i\alpha\cdot\sigma\beta/2} = I\cos\frac{\beta}{2} + i\alpha\cdot\sigma\sin\frac{\beta}{2}$$

with $\alpha\cdot\sigma = a_xX + a_yY + a_zZ$ involving three Pauli matrices $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$, $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, and $(a_x, a_y, a_z) \in \mathbb{R}^3$.

In the next two sections, quantum state preparation and block-encoding protocols based on multiplexor operations will be provided.

A. QUANTUM STATE PREPARATION BY MULTIPLEXOR OPERATIONS

The state preparation procedure based on precomputed amplitudes has been well established in the literature [29], [33], [34], [35], [36]. The quantum circuit for quantum state preparation by multiplexor operation is shown in Fig. 2, where the relationship between the rotation parameters of the multiplexor operations in the circuit can be visualized using the data structures of the Rotation-Y binary tree and the Rotation-Z binary tree, as shown in Fig. 3.

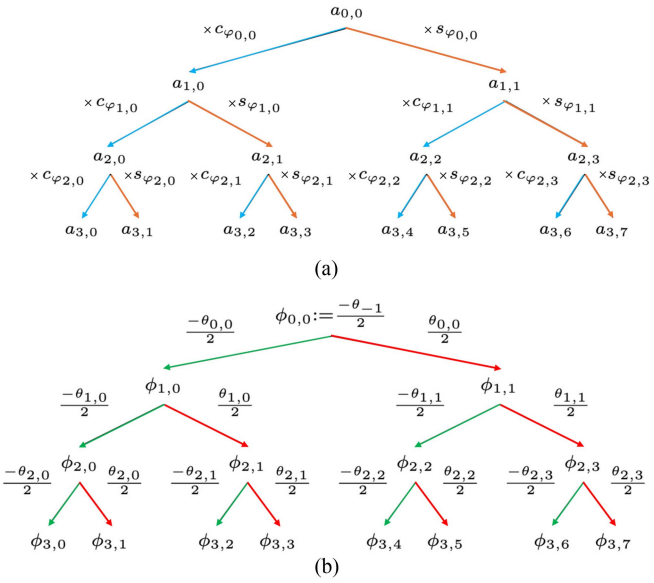


FIG. 3. Rotation-Y binary tree and Rotation-Z binary tree. Each leaf node element in Rotation-Y binary tree is the product of the elements on path of edges from the root node to that leaf node, where the value of the left-child edge is $c_\varphi = \cos(\varphi/2)$ and the value of the right-child edge is $s_\varphi = \sin(\varphi/2)$. Each leaf node element in Rotation-Z binary tree is the sum of elements on the path from the root node to that leaf node. (a) Rotation-Y binary tree. (b) Rotation-Z binary tree.

The Rotation-Y binary tree [see Fig. 3(a)] is used to generate the norms of the amplitudes $\{|\psi_k|\}_{k=0}^{2^n-1}$ (or $\{\psi_k\}_{k=0}^{2^n-1}$ for real amplitudes) under the computational basis $\{|k\rangle\}_{k=0}^{2^n-1}$.

IEEE Transactions on Quantum Engineering

Algorithm 1: Angle (Compute Angles in Nodes of Rotation-Y Binary Tree).

Require: Vectors $[a_j]_{j=0}^{2^n-1} \in \mathbb{C}^{2^n}$ with $|a_j| = 1$ for $0 \le j \le 2^n - 1$

Ensure: Vectors $[\varphi_j]_{j=0}^{2^n-1} \in \mathbb{R}^{2^n}$

1: Get φ_j such that $e^{\frac{\varphi_j}{2}i} := \mathrm{Re}(a_j) + \mathrm{Im}(a_j)i$ for all $0 \le j \le 2^n - 1$

Algorithm 2: $\mathrm{R_Y}$-Angles (Compute Rotation-Y Binary Tree Parameters).

Require: The state $[\psi_j]_{j=0}^{2^n-1} \in \mathbb{R}^{2^n}$

Ensure: Rotation Parameters $[\varphi_{k,j}]_{k=0,\ldots,n-1}^{j=0,\ldots,2^n-2}$

1: $a_{n,j} \leftarrow \psi_j$ for all $0 \le j \le 2^n - 1$
2: **for** $k = n-1, \ldots, 0$
3: **for** $j = 0, \ldots, 2^k - 1$
4: $\varphi_{k,j} \leftarrow \mathrm{angle}(a_{k+1,2j} + a_{k+1,2j+1} \cdot i)$
5: $a_{k,j} \leftarrow \sqrt{|a_{k+1,2j}|^2 + |a_{k+1,2j+1}|^2}$
6: **end for**
7: **end for**

A quantum state with 2^n amplitudes can be generated by a Rotation-Y binary tree with n layers. Let $|\psi_k|$ be the kth leaf node $\{a_{n,k}\}_{k=0}^{2^n-1}$ in the nth layer of the Rotation-Y binary tree. The value of a node in the tth layer satisfies the product of its edge and its parent node's value, i.e.,

$$a_{t,k} = \begin{cases} a_{t-1,\lfloor k/2 \rfloor} \times \cos(\varphi_{t-1,\lfloor k/2 \rfloor}/2), & \text{for even } k \\ a_{t-1,\lfloor k/2 \rfloor} \times \sin(\varphi_{t-1,\lfloor k/2 \rfloor}/2), & \text{for odd } k \end{cases} \tag{3}$$

for all $0 \le t \le n-1$ and $0 \le k \le 2^t - 1$. Equation (3) leads to the relation

$$\frac{\varphi_{t,k}}{2} = \mathrm{angle}(a_{t+1,2k-1} + a_{t+1,2k} \cdot i)$$

i.e.,

$$\begin{aligned} e^{\frac{\varphi_{t,k}}{2}i} &= \cos\left(\frac{\varphi_{t,k}}{2}\right) + \sin\left(\frac{\varphi_{t,k}}{2}\right) \cdot i \\ &= a_{t+1,2k-1} + a_{t+1,2k} \cdot i. \end{aligned}$$

The pseudocode of the Rotation-Y binary tree is given in Algorithm 2, where the angle$(\cdot)$ is stated in Algorithm 1.

The Rotation-Z binary tree [see Fig. 3(b)] is used to generate the phases $\{e^{i\phi_k}\}_{k=0}^{2^n-1}$ under the computational basis $\{|k\rangle\}_{k=0}^{2^n-1}$. A quantum state with 2^n amplitudes can be generated by a Rotation-Z binary tree with n layers. Let ϕ_k be the kth leaf node $\{\phi_{n,k}\}_{k=0}^{2^n-1}$ in the nth layer of the Rotation-Z binary tree. The value of a node in the Rotation-Z binary tree is the sum of its edge and its parent node's value, i.e.,

$$\phi_{t,k} = \phi_{t-1,\lfloor k/2 \rfloor} + \frac{(-1)^{k+1}}{2}\theta_{t-1,\lfloor k/2 \rfloor} \tag{4}$$

for $1 \le t \le n$ and $0 \le k \le 2^t - 1$.

For any pure state, the sum of the squares of its amplitudes is equal to 1. The amplitudes $\{|\psi_k|\}_{k=0}^{2^t-1}$ can be determined by

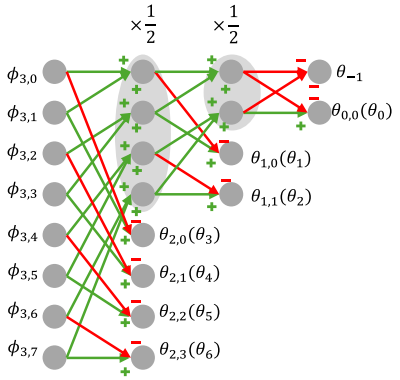


FIG. 4. Computation process of parameters in the Rotation-Z binary tree with $n = 3$ layers.

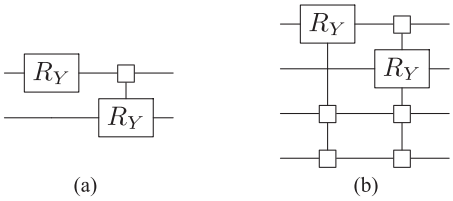


FIG. 5. State preparation U maps $|0\rangle^{\otimes 2}$ to any two-qubit real state. Controlled-state preparation U_R conditionally maps $|0\rangle^{\otimes 2}|k\rangle$ to four-qubit real state $|\psi_k\rangle|k\rangle$ for $k = 0, \ldots, 3$, where $|\psi_k\rangle$ can be any two-qubit real state. (a) State preparation U. (b) Controlled-state preparation U_R.

$2^t - 1$ degrees of freedom in a Rotation-Y binary tree with t layers; the relation can be characterized by Fig. 2. The angles θ_{-1} and $\{\theta_{t,k}\}_{0 \le t \le n, 0 \le k \le 2^t-1}$ above n layers of the Rotation-Z binary tree can be solved by the linear system

$$M_{R_Z}^t \begin{bmatrix} \theta_{-1} \\ \theta_{0,0} \\ \vdots \\ \theta_{t-1,2^{t-1}-1} \end{bmatrix} = \begin{bmatrix} \phi_{t,0} \\ \phi_{t,1} \\ \vdots \\ \phi_{t,2^t-1} \end{bmatrix}$$

for all $1 \le t \le n$, where the matrix elements of $M_{R_Z}^t$ are determined by (4). The computation of the inverse matrix $(M_{R_Z}^t)^{-1}$ follows the recursive relationship

$$\left(M_{R_Z}^1\right)^{-1} = \begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$\left(M_{R_Z}^t\right)^{-1} = \begin{bmatrix} \frac{1}{2}\left(M_{R_Z}^{t-1}\right)^{-1} & 0 \\ 0 & I_{2^{t-1}} \end{bmatrix}$$

$$\left(\begin{bmatrix} -1 & -1 & 1 & 1 \end{bmatrix} \otimes I_{2^{t-1}}\right)$$

for $t = 2, \ldots, n$. It leads to a simplified computation process of the Rotation-Z binary tree, which can be visualized in Fig. 4.

The pseudocode of the Rotation-Z binary tree is given in Algorithm 3.

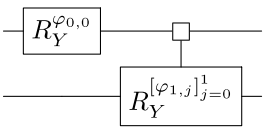
Algorithm 3: R_Z-Angles (Compute Rotation-Z Binary Tree Parameters).

Require: The phases of state $[\phi_k]_{k=0}^{2^n-1}$
Ensure: Rotation parameters and global phase $([\theta_k]_{k=0}^{2^n-2}, \theta_{-1})$
for $i = 1, \ldots, n$
  For $0 \le k \le 2^{n-i+1} - 1$
    $a_k \leftarrow \phi_k$
  end for
  for $j = 0, \ldots, 2^{n-i} - 1$
    $\phi_j \leftarrow \frac{a_{2j-1}+a_{2j}}{2}$
    $\phi_{2^{n-i}+j} \leftarrow -a_{2j-1} + a_{2j}$
  end for
end for
$\theta_{-1} \leftarrow -a_0 - a_1$
for $k = 0, \ldots, 2^n - 2$
  $\theta_k \leftarrow \phi_{k+1}$
end for

Lemma 1: Given a quantum state $|\psi\rangle = \sum_{k=0}^{2^n-1} e^{i\phi_k} |\psi_k| \, |k\rangle \in \mathbb{C}^{2^n}$, the operation count for computing the parameters $\{\varphi_j\}_{j=0}^{2^n-1}$ and $\{\phi_j\}_{j=0}^{2^n-1}$ in Rotation-Y and Rotation-Z binary tree to prepare $|\psi\rangle$ is $\Theta(2^n)$.

Proof: Since the Rotation-Y and Rotation-Z binary tree for encoding 2^n amplitudes requires n layers, parameters in the lth layer in a tree consist of 2^l edges, which require $\Theta(2^l)$ operations for computation. Therefore, it takes $\sum_{l=1}^{n} 2^l = \Theta(2^n)$ operations to compute all the rotation parameters. □

Example 1: Given a state $|\psi\rangle = \sum_{j=0}^{3} \varphi_j |j\rangle \in \mathbb{R}^4$, it can be prepared using the quantum circuit implemented by the multiplexor as follows:



By the Rotation-Y binary tree, the parameters can be calculated as

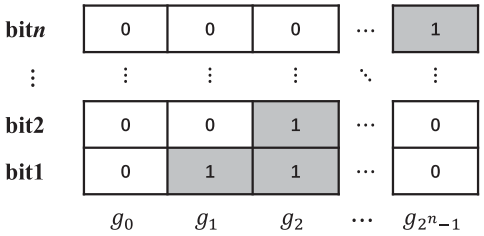
$$\varphi_{1,0} = 2 \text{ angle } (\varphi_0 + \varphi_1 i)$$
$$\varphi_{1,1} = 2 \text{ angle } (\varphi_2 + \varphi_3 i)$$
$$\varphi_{0,0} = 2 \arccos \left(\frac{\sqrt{|\varphi_0|^2 + |\varphi_1|^2}}{\sqrt{\sum_{j=0}^{3} |\varphi_j|^2}} \right).$$

B. BLOCK ENCODING BY STATE PREPARATION

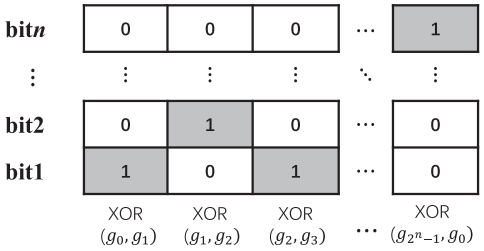
The block-encoding protocol in this article follows the prescription laid out in [2], [23], [29], and [37]. The block-encoding unitary U_A comprises the (controlled-) state preparation operators U_R and U_L and swap gates, i.e.,

$$U_A = U_L^{\dagger} \text{SWAP}_{n_1,n_2} U_R \tag{5}$$

Step 1. Initialize Gray codes g_0-g_{2^n-1}:



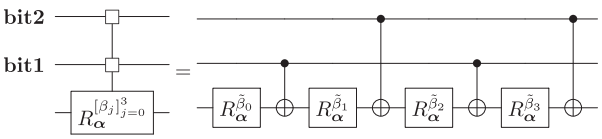
Step 2. Implement Exclusive-Or (XOR) operations on the n-bit cyclic Gray codes.



Step 3. The order of CNOT sequences are determined by the order of '1' bit in the transformed Gray codes generated in Step 2:

The order of control sequences of CNOT:
bit1 -> bit2 -> bit1 -> ⋯ -> bit$n$

(a)



(b)

FIG. 6. Quantum circuit of the uniformly controlled rotation with two control nodes. (a) Process of CNOT position determination based on the operations on the 2-bit cyclic binary Gray codes. (b) Quantum circuit of the uniformly controlled rotation $\sum_{j=0}^{3} |j\rangle\langle j| \otimes R_\alpha^{\beta_j}$ with $\alpha_x = 0$.

where SWAP_{n_1,n_2} denotes the swap gates implemented in the register n_1 and n_2. There are two different subnormalization factors in this block-encoding protocol using different state preparation based on (5).

On the one hand, if we define

$$|0\rangle_{n_1} |j\rangle_{n_2} \xrightarrow{U_R} \sum_{k=0}^{2^n-1} |A_j\rangle_{n_1} |j\rangle_{n_2}$$
$$|0\rangle_{n_1} |k\rangle_{n_2} \xrightarrow{U_L} |A_F\rangle_{n_1} |k\rangle_{n_2} \tag{6}$$

where the states $|A_j\rangle_{n_1} = \sum_{k=0}^{2^n-1} \frac{A_{k,j}}{\|A_{\cdot,j}\|} |k\rangle_{n_1}$, $|A_F\rangle = \sum_{j=0}^{2^n-1} \frac{\|A_{\cdot,j}\|}{\|A\|_F} |j\rangle_{n_1}$, and $\|A_{\cdot,j}\|$ denote the Euclidean norm of the jth column of A, then

$$\langle 0|_{n_1} \langle k|_{n_2} U_A |0\rangle_{n_1} |j\rangle_{n_2} = A_{k,j}/\|A\|_F.$$

On the other hand, if we define U_R and U_L as [23], [29]

$$|0\rangle_{n_1+2}\,|j\rangle_{n_2} \xrightarrow{U_R} |A_j^{(p)}\rangle_{n_1+2}\,|j\rangle_{n_2}$$

$$|0\rangle_{n_1+2}\,|k\rangle_{n_2} \xrightarrow{U_L} |\tilde{A}_k^{(p)}\rangle_{n_1+2}\,|k\rangle_{n_2}$$

for $p \in [0, 1]$, we have

$$|A_j^{(p)}\rangle = \sum_{k\in[N]} \frac{e^{i\theta_{k,j}}|A_{k,j}|^p}{\sqrt{\|A_{\cdot,j}\|_{2p}^{2p}}}\,|k\rangle_{n_1}\left[\cos\chi_j\,|0\rangle + \sin\chi_j\,|1\rangle\right]|0\rangle$$

$$|\tilde{A}_k^{(p)}\rangle = \sum_{j\in[N]} \frac{|A_{k,j}|^{1-p}}{\sqrt{\|A_{k,\cdot}\|_{2(1-p)}^{2(1-p)}}}\,|j\rangle_{n_1}\,|0\rangle\,[\cos\chi_k\,|0\rangle + \sin\chi_k\,|1\rangle]$$

where $\cos\chi_j = \sqrt{\frac{\|A_{\cdot,j}\|_{2p}^{2p}}{S_{2p}(A^T)}}$, $\cos\chi_k = \sqrt{\frac{\|A_{k,\cdot}\|_{2(1-p)}^{2(1-p)}}{S_{2(1-p)}(A)}}$, and $S_q(A) = \max_k \|A_{k,\cdot}\|_q^q$ are the qth power of the maximum q-norm of any row of A. Then

$$\langle 0|_{n_1+2}\,\langle k|_{n_2}\,U_A\,|0\rangle_{n_1+2}\,|j\rangle_{n_2} = A_{k,j}/\mu_p(A^T),$$

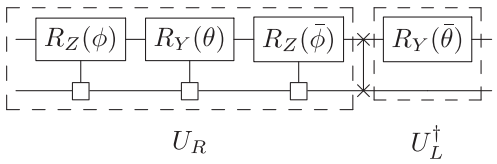
where $\mu_p(A^T) = \sqrt{S_{2p}(A^T)S_{2(1-p)}(A)}$.

The controlled-state preparation can be implemented by multiplexor operation. Suppose that $A \in \mathbb{R}^{4\times4}$ and $\sum_{k,j=0,\ldots,3}|A_{j,k}|^2 = 1$, and U_R maps $|0\rangle^{\otimes2}\,|j\rangle$ to $\sum_{k=0,\ldots,3}A_{k,j}\,|k\rangle\,|j\rangle$ for $j \in \{0, 1, 2, 3\}$; then, the quantum circuit of controlled-state preparation U_R is in the form of Fig. 5(b).

Example 2: In case that $A \in \mathbb{C}^{2\times2}$, we encode the matrix elements as

$$\begin{cases} A_{0,0} = e^{-(\bar{\phi}_0+\phi_0)i}\cos\bar{\theta}_0\cos\theta_0 \\ A_{0,1} = -e^{-(\bar{\phi}_1+\phi_1)i}\sin\bar{\theta}_0\cos\theta_1 \\ A_{1,0} = e^{(\bar{\phi}_0-\phi_0)i}\cos\bar{\theta}_0\sin\theta_0 \\ A_{1,1} = -e^{(\bar{\phi}_1-\phi_1)i}\sin\bar{\theta}_0\sin\theta_1. \end{cases}$$

Fast parameter computation methods for circuit synthesis for this block-encoding protocol will be provided in Section IV. The quantum circuit encoding of the aforementioned matrix using multiplexor operations follows the form



Since

$$U_L = \begin{bmatrix} \cos\bar{\theta}_0 & & * & \\ & \cos\bar{\theta}_0 & & * \\ \sin\bar{\theta}_0 & & * & \\ & \sin\bar{\theta}_0 & & * \end{bmatrix}$$

$$U_R = \begin{bmatrix} e^{-(\bar{\phi}_0+\phi_0)i}\cos\theta_0 & & * & \\ & e^{-(\bar{\phi}_1+\phi_1)i}\cos\theta_1 & & * \\ e^{(\bar{\phi}_0-\phi_0)i}\sin\theta_0 & & * & \\ & e^{(\bar{\phi}_1-\phi_1)i}\sin\theta_1 & & * \end{bmatrix}$$

it follows that

$$U_L^\dagger \cdot \mathrm{SWAP} \cdot U_R =$$

$$\begin{bmatrix} e^{-(\bar{\phi}_0+\phi_0)i}\cos\bar{\theta}_0\cos\theta_0 & -e^{-(\bar{\phi}_1+\phi_1)i}\sin\bar{\theta}_0\cos\theta_1 & * & * \\ e^{(\bar{\phi}_0-\phi_0)i}\cos\bar{\theta}_0\sin\theta_0 & -e^{(\bar{\phi}_1-\phi_1)i}\sin\bar{\theta}_0\sin\theta_1 & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$

where the top-left corner of the aforementioned form is the objective matrix in the block encoding.

III. DECOUPLE MULTIPLEXOR OPERATION

In this section, we will introduce some efficient compilation methods for multiplexor operations based on Gray code. The demultiplexor method comprises two components: single-qubit rotation parameters calculation and CNOT position determination.

A. PARAMETER CALCULATION

Given a multiplexor operation $\sum_{j=0}^{2^n-1}|j\rangle\langle j|\otimes R_{\boldsymbol{\alpha}}^{\beta_j}$, where $R_{\boldsymbol{\alpha}}^{\beta_j} = e^{i\boldsymbol{\alpha}\cdot\boldsymbol{\sigma}\beta_j/2}$ and $\boldsymbol{\alpha}\cdot\boldsymbol{\sigma} = a_xX + a_yY + a_zZ$, it follows that

$$XR_{\boldsymbol{\alpha}}^{\beta}X = R_{\boldsymbol{\alpha}}^{-\beta} \tag{7}$$

when $a_x = 0$. Furthermore, subsequent rotations about any single axis $\boldsymbol{\alpha}$ are additive as

$$R_{\boldsymbol{\alpha}}^{\beta_1}R_{\boldsymbol{\alpha}}^{\beta_2} = R_{\boldsymbol{\alpha}}^{\beta_1+\beta_2}. \tag{8}$$

Based on (7) and (8), the single-qubit rotation parameters $[\tilde{\beta}_j]_{j=0}^{2^n-1}$ can be derived from the multiplexor operation parameters $[\beta_j]_{j=0}^{2^n-1}$ by solving a linear system

$$M^n\begin{pmatrix}\tilde{\beta}_0\\ \vdots \\ \tilde{\beta}_{2^n-1}\end{pmatrix} = \begin{pmatrix}\beta_0\\ \vdots \\ \beta_{2^n-1}\end{pmatrix} \tag{9}$$

where the linear system involves a $2^n \times 2^n$ matrix $M^n \equiv H^{\otimes n}P_G$ [12], where $H^{\otimes n} = \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}^{\otimes n}$ is the Walsh–Hadamard transformation, and P_G is the Gray permutation matrix [38] that maps n-bit binary code ordering to n-bit binary reflected Gray code ordering.

The binary reflected Gray code [39] is a scheme for listing all n-bit binary numbers so that successive numbers differ in exactly one bit.

Table 1 gives an example of the Gray permutation mapping. Based on this table, the 3-bit Gray permutation mapping is in the form of

$$P_G = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & & 1 & & & & \\ & & 1 & & & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & 1 & & & \\ & & & 1 & & & & \end{bmatrix}.$$

B. CNOT POSITION DETERMINATION

The order of control nodes within the sequence of CNOT gates [38] is ascertainable through the transformed binary reflected Gray code, as the control nodes are delineated by the "1" resultant from the EXCLUSIVE-OR (XOR) operations on the cyclic k-bit binary Gray codes. Fig. 6(a) illustrates the methodology by which the ordering of control nodes of CNOT gates is established, alongside the quantum circuit of the uniformly controlled rotation with two controlled qubits predicated upon that specific CNOT order.

Above all, the uniformly controlled rotation with $a_x = 0$ can be implemented using single-qubit and CNOT gates [38], as illustrated in Fig. 6(b).

The following lemma derived from [12] and [40] demonstrates the computational complexity of decoupling uniformly controlled rotation.

Lemma 2: The single-qubit and CNOT gate's decomposition of $\sum_j |j\rangle\langle j| \otimes R_{\boldsymbol{\alpha}}^{[\beta_j]_{j=0}^{2^n-1}}$ with $a_x = 0$ can be calculated in an operation count of $\mathcal{O}(n2^n)$.

Proof: Since the single-qubit gates' rotation parameters of uniformly controlled gates in (9) can be calculated by the fast Walsh–Hadamard transform [40], [41] with an operation count of $\frac{23}{24}n2^n + \mathcal{O}(2^n)$ [42], the Gray code permutation [12], [40] and the position of CNOT gates can be calculated with $\Theta(2^n)$ operations. To sum up, it takes $\mathcal{O}(n2^n)$ operations. □

The extension of the uniformly controlled rotation as described in (2), originally applicable when the target node is situated at the circuit's bottom, is herein generalized to scenarios where the target node may assume any position. The mathematical representation of the multiplexor, with the control node positioned at bit t, is provided as follows:

$$\sum_{j=0}^{2^n-1} \left(\otimes_{k=n,\ldots,t} |j_k\rangle\langle j_k|\right) \otimes R_{\boldsymbol{\alpha}}^{\beta_j} \left(\otimes_{k=t-1,\ldots,1} |j_k\rangle\langle j_k|\right) \quad (10)$$

where $|j\rangle \equiv |j_n\rangle \otimes |j_{n-1}\rangle \otimes \cdots \otimes |j_1\rangle$.

To decouple the aforementioned multiplexor, we introduce the permutative demultiplexor and recursive demultiplexor methods in the subsequent subsections. These two methods are summarized by Table 2.

C. PERMUTATIVE DEMULTIPLEXOR

The first decomposition method, termed as the "permutative demultiplexor" method, is derived from the permutation of indices in the uniformly controlled rotations [38]. In this method, all single-qubit rotation parameters are computable via (9). The primary focus is on the configuration of the CNOT sequence.

CNOT position determination: Fig. 6(a) demonstrates that the order of control nodes in the multiplexor associated with (10) is established through the ensuing procedures.

1) Initialize n-bit reflected Gray codes (ranged from g_0 to g_{2^n-1}).
2) Apply the EXCLUSIVE OR logical operation to these n-bit sequences.
3) The control positions of the CNOT sequences are determined by the "1" position (highlighted in Gray) in that transformed sequence.

Based on the aforementioned control node ordering, the multiplexor $\sum_{j=0}^{2^n-1}(\otimes_{k=n,\ldots,t} |j_k\rangle\langle j_k|) \otimes R_{\boldsymbol{\alpha}}^{\beta_j}(\otimes_{k=t-1,\ldots,1} |j_k\rangle\langle j_k|)$ can be implemented using the same control node ordering by positioning the control node between bit t and bit $t-1$, as shown in Fig. 7(b). The arrangement of control nodes in Fig. 7(b) has undergone a permutation. Consequently, this configuration is designated as the "permutative demultiplexor" method.

D. RECURSIVE DEMULTIPLEXOR

The second decomposition strategy for decoupling (10), known as the "recursive demultiplexor" method, leverages the recursive decomposition characteristics of the multiplexor operation.

Similar to the recursive cancellation of uniformly controlled rotation proposed in the article [32], a multiplexor operation can be decomposed into simpler multiplexor operations interleaved with CNOT gates, which can be visualized in Fig. 8.

Consider a multiplexor operation with $n-t+1$ control nodes in the upper part of the circuit and $t-1$ control nodes in the lower part. Parameter calculation and CNOT sequence arrangement in the recursive demultiplexor method are summarized as follows.

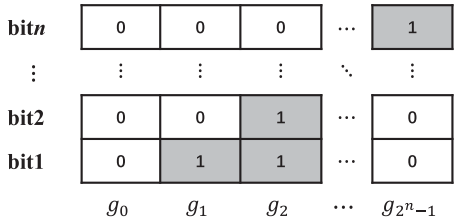
1) PARAMETER CALCULATION

The single-qubit rotation parameters in the recursive demultiplexor method can be calculated in two steps.

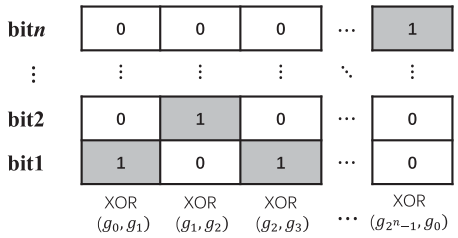
1) The intermediate variables $\tilde{\beta}_{i,j}^{(1)}$ can be calculated by the linear system as

$$M^{n-t} \begin{bmatrix} \tilde{\beta}_{1,0}^{(1)} & \cdots & \tilde{\beta}_{1,2^t-1}^{(1)} \\ \vdots & \ddots & \vdots \\ \tilde{\beta}_{2^{n-t},0}^{(1)} & \cdots & \tilde{\beta}_{2^{n-t},2^t-1}^{(1)} \end{bmatrix} = \text{reshape}\left(\left[\beta_j\right]_{j=0}^{2^n-1}, [2^{n-t}, 2^{t-1}]\right) \quad (11)$$

Step 1. Initialize Gray codes g_0-g_{2^n-1}:



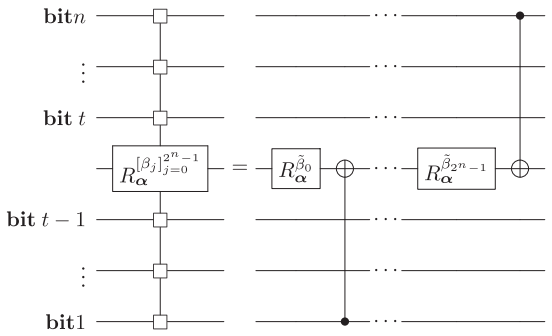
Step 2. Implement Exclusive-Or (XOR) operations on the n-bit cyclic Gray codes.



Step 3. The order of CNOT sequences are determined by the order of '1' bit in the transformed Gray codes generated in Step 2:

The order of control sequences of CNOT:
bit1 -> bit2 -> bit1 -> ⋯ -> bit$n$

(a)



(b)

FIG. 7. Using permutative demultiplexor method to decouple $\sum_{j=0}^{2^n-1}(\otimes_{k=n,\ldots,t}|j_k\rangle\langle j_k|)\otimes R_\alpha^{\beta_j}(\otimes_{k=t-1,\ldots,1}|j_k\rangle\langle j_k|)$ with $\alpha_x = 0$. (a) Process of CNOT position determination of $\sum_{j=0}^{2^n-1}(\otimes_{k=n,\ldots,t}|j_k\rangle\langle j_k|) \otimes R_\alpha^{\beta_j}(\otimes_{k=t-1,\ldots,1}|j_k\rangle\langle j_k|)$ using the permutative demultiplexor method. (b) Quantum circuit of $\sum_{j=0}^{2^n-1}(\otimes_{k=n,\ldots,t}|j_k\rangle\langle j_k|)\otimes R_\alpha^{\beta_j}(\otimes_{k=t-1,\ldots,1}|j_k\rangle\langle j_k|)$ with $\alpha_x = 0$, whose control nodes ordering is determined in (a) and parameters $\tilde{\beta}_j$ are computed via (9).

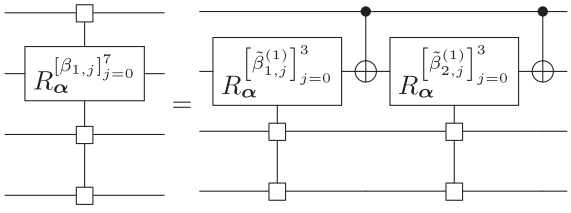


FIG. 8. Decouple the top node of the $\sum_{j=0}^{7}|j_3\rangle\langle j_3|\otimes R_\alpha^{\beta_j}(\otimes_{k=2,1}|j_k\rangle\langle j_k|)$ into two multiplexor operations.

where the matrix M^{n-t} is a 2^{n-t}-by-2^{n-t}-dimensional matrix defined in (9), and reshape$(\beta, [n_1, n_2])$ reshapes β into an n_1-by-n_2 matrix.

2) Parameters $\tilde{\beta}_{i,j}^{(2)}$ can be calculated by the linear system as

$$M^t \begin{bmatrix} \tilde{\beta}_{1,0}^{(2)} & \cdots & \tilde{\beta}_{2^{n-t},0}^{(2)} \\ \vdots & \ddots & \vdots \\ \tilde{\beta}_{1,2^t-1}^{(2)} & \cdots & \tilde{\beta}_{2^{n-t},2^t-1}^{(2)} \end{bmatrix} = \begin{bmatrix} \tilde{\beta}_{1,0}^{(1)} & \cdots & \tilde{\beta}_{2^{n-t},0}^{(1)} \\ \vdots & \ddots & \vdots \\ \tilde{\beta}_{1,2^t-1}^{(1)} & \cdots & \tilde{\beta}_{2^{n-t},2^t-1}^{(1)} \end{bmatrix}. \quad (12)$$

2) CNOT POSITION DETERMINATION

The control node ordering of the CNOT gates is established through the ensuing procedures.

1) Initialize the n-bit codes into two sequences: $(n-t)$-bit top Gray codes sequence **u-bit**, the t-bit bottom sequence Gray codes sequence **l-bit**.
2) Apply the XOR logical operation to the Gray codes **u-bit** and **l-bit** sequences separately.
3) Broadcast the bottom sequence for each Gray code in the top sequence.
4) The control position of CNOT sequences can be determined by the bit "1" (highlighted by Gray) of the transformed Gray code sequences.

The recursive demultiplexer method introduces periodic idle single-qubit gates between consecutive CNOTs. Specifically, a block of 2^{n-t} idle gates repeats after every 2^t active single-qubit rotations. Fig. 9 demonstrates the decoupling of a multiplexor operation with the control node located between the bit2 and bit3 positions using the recursive demultiplexor method.

Lemma 3: The single-qubit and CNOT gate decomposition of a multiplexor operation with n control nodes, using the recursive demultiplexor method, can be calculated to an operation count of $\mathcal{O}(n2^n)$.

Proof: The linear equation

$$M^k \begin{bmatrix} \tilde{\beta}_0 \\ \vdots \\ \tilde{\beta}_{2^k-1} \end{bmatrix} = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_{2^k-1} \end{bmatrix}$$

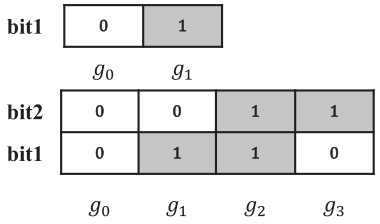
can be solved in $\mathcal{O}(k2^k)$ operations [40], [41]. To solve the parameters $\{\tilde{\beta}_{i,j}^{(2)}\}_{j=0,\ldots,2^t-1}^{i=1,\ldots,2^{n-t}}$ with $t = \lceil n/2 \rceil$ in (11) and (12), it takes $\mathcal{O}(n2^n)$ operations.

□

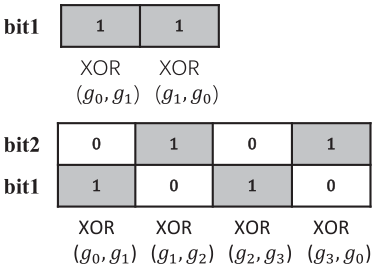
IV. BINARY TREE BLOCK-ENCODING PROTOCOL

The protocol for block encoding, denoted as binary tree block encoding (BITBLE), is founded on (5), utilizing quantum

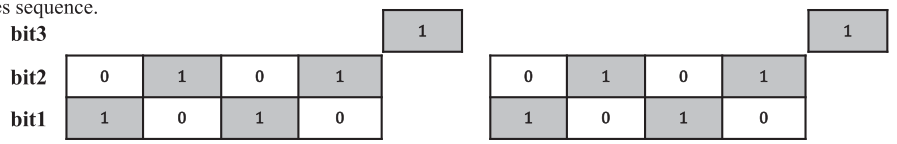
Step 1. Initialize two Gray code sequences g_0-g_{2^n-t-1} (g_0-g_1) and g_0-g_{2^t-1} (g_0-g_3):



Step 2. Implement Exclusive-Or (XOR) operations on these two cyclic Gray codes respectively.



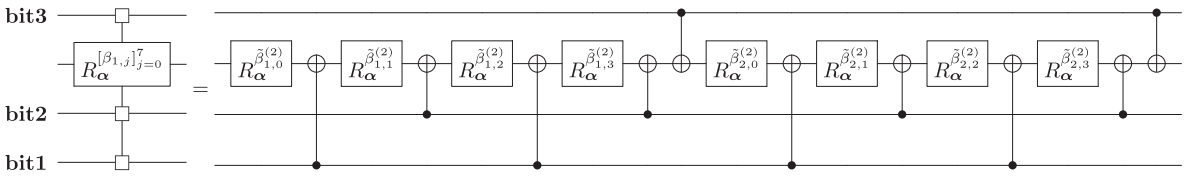
Step 3. Broadcast the $(n-t)$-bit codes in the upper part of Step 2 based on the t-bit codes in the lower part of Step 2 to generate a new n-bit codes sequence.



Step 4. The order of CNOT sequences are determined by the order of '1' bit in the transformed Gray codes generated in Step 3:

The order of control sequences of CNOT:
bit1 -> bit2 -> bit1 -> bit2 -> bit3 -> bit1 -> bit2 -> bit1 -> bit2 -> bit3

(a)



(b)

FIG. 9. **Using recursive demultiplexor method to decouple $\sum_{j=0}^{7} |j_3\rangle\langle j_3| \otimes R_\alpha^{\beta_j} \otimes |j_2\rangle\langle j_2| \otimes |j_1\rangle\langle j_1|$ with $a_x = 0$. (a) Process of determining the order of CNOT's control sequence for implementing $\sum_{j=0}^{2^n-1} (\otimes_{k=n,\ldots,t} |j_k\rangle\langle j_k|) \otimes R_\alpha^{\beta_j} (\otimes_{k=t-1,\ldots,1} |j_k\rangle\langle j_k|)$ using the recursive demultiplexor method with $n = 3$ and $t = 2$. (b) Quantum circuit of $\sum_{j=0}^{7} |j_3\rangle\langle j_3| \otimes R_\alpha^{\beta_j} (\otimes_{k=2,1} |j_k\rangle\langle j_k|)$ operations based on the recursive demultiplexor method, whose control nodes ordering is determined in (a) and parameters $\tilde{\beta}_{\cdot}^{(2)}$ are computed via (11) and (12).**

state preparation facilitated by multiplexor operations. The complete parameters' computation follows three processes.

Process 1: From classical data to multiplexor operations' parameters. Compute the multiplexor operation parameters in BITBLE[1] and BITBLE[2] protocols by Algorithm 4, or BITBLE[3] by Algorithm 5 in Appendix A.

Process 2: From multiplexor operations' parameters to single-qubit rotations' parameters. Decouple the multiplexor operations using the "permutative demultiplexor" or "recursive demultiplexor" methods stated in Section III.

Process 3: Circuit compression. Similar to the compression function described in [12], given a predefined threshold δ_c, any rotation angle satisfying $|\tilde{\beta}_{k,j}| \leq \delta_c$ is considered negligible, allowing the corresponding single-qubit gate to be removed. Furthermore, consecutive CNOT gates that mutually commute and share the same target qubit can be canceled out via a parity check, as outlined in [12].

The protocol for block encoding, which integrates a subnormalization factor $\|A\|_F$, is illustrated in Fig. 10. Within this framework, both the controlled-state preparation U_R and the state preparation $U_L^\dagger$ are associated with (6).

Since block-encoding protocols are decoupled through the Rotation-Y/Z binary tree, this approach is designated as "binary tree block encoding." In order to elucidate the distinctions among the different demultiplexor methods and subnormalization factors, the methodology delineated in BITBLE[1]–BITBLE[3] is comprehensively detailed in Table 3. To facilitate a comparative analysis, the protocol outlined in FABLE [12] is also taken into account.

A. COMPUTATIONAL COMPLEXITY OF PARAMETER COMPILATION

In this section, we will provide two theorems to demonstrate the computational complexity of parameter compilation for quantum state preparation and block encoding.

Theorem 1 (Computational Complexity of Circuit Compilation for State Preparation): The time complexity to compile the state preparation circuit in a size of 2^n is $\mathcal{O}(n2^n)$.

IEEE Transactions on Quantum Engineering

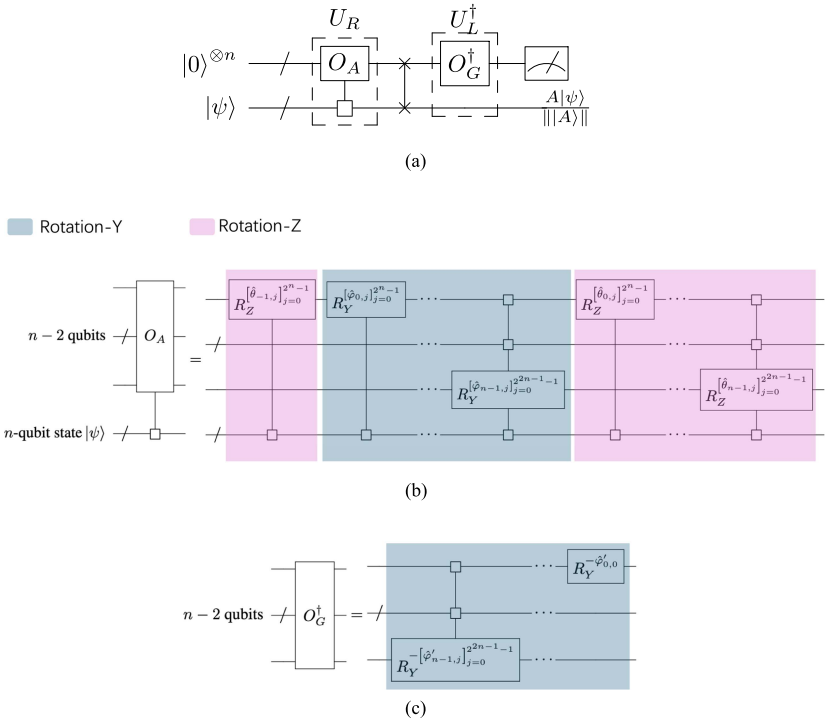


FIG. 10. **Quantum circuits of binary tree block encoding (BITBLE) with subnormalization $\|A\|_F$. (a) Circuit frameworks of BITBLE. (b) State preparation unitary U_R. (c) Inverse state preparation unitary $O_G^\dagger$.**

Proof: On the one hand, it takes $\Theta(2^n)$ time to compute the multiplex or operation parameters using the Rotation-Y and Rotation-Z binary tree by Lemma 1. On the other hand, it takes $\mathcal{O}(n2^n)$ time to decompose single- and two-qubit gates using the permutative demultiplexor method by Lemma 2 or using the recursive demultiplexor method by Lemma 3. Overall, it takes $\mathcal{O}(n2^n)$ time to decouple a preparation circuit with binary tree data structures. □

Theorem 2 (Computational Complexity of Circuit Compilation for BITBLE Protocol) The computational complexity to compile BITBLE protocol of a $2^n \times 2^n$ matrix is $\mathcal{O}(n2^{2n})$.

Proof: We calculate the time complexity step by step.

1) Multiplexor operations' parameters: By Lemma 1, $2^n \times 2^n$ multiplexor operations' parameters can be computed with $\Theta(2^{2n})$ operations. Therefore, the multiplexor operations' parameters in the BITBLE protocol can be generated in $\Theta(2^{2n})$ operations.
2) Decoupling multiplexor operations' parameters: Both the recursive demultiplexor method and the permutative demultiplexor method require $\mathcal{O}(n2^{2n})$ operations.
3) Circuit compression: It requires $\Theta(2^{2n})$ operations to compress $\Theta(2^{2n})$ parameters and parity check at most $\Theta(2^{2n})$ on consecutive CNOT gates.

Summing up, it requires $\mathcal{O}(n2^{2n})$ operations to calculate all single-qubit parameters and CNOT gates in the BITBLE protocol. □

B. APPROXIMATE ERROR

Based on the circuit compression method stated in [12], some single-qubit gates whose rotation angles are less than a threshold value will be removed. The approximation leads to some errors. The case of real-valued data is considered by the following theorem.

Theorem 3: For an n-qubit matrix $A \in \mathbb{R}^{2^n \times 2^n}$, $\|A\|_F = 1$, and the BITBLE circuit with a cutoff threshold $\delta_c \in \mathbb{R}^+$ gives an $(1, n, n2^{3n}\delta_c)$-block encoding of A up to third order in δ_c.

Proof: Consider the parameters $\hat{\boldsymbol{\theta}}$ with cutoff δ_c, and the error of the angles decoupled by uniformly controlled rotations is bounded by $\|\boldsymbol{\delta\theta}\|_2 \leq \|\hat{H}^{\otimes 2n}\|_2 \|P_G\|_2 \|\boldsymbol{\delta\hat{\theta}}\|_2 \leq 2^{2n}\delta_c$ [12]. Thus, the elementwise error is bounded by $|\delta\theta_i| = |\theta_i - \tilde{\theta}_i|$.

By the binary tree data structure, the element of $A_{k,j} = \prod_{t=1}^{2n} \cos\theta_t \cdot \delta_{t,k,j} + \sin\theta_t \cdot (1 - \delta_{t,k,j})$, where $\delta_{t,k,j} = 0$ or 1. The cutoff error can be bounded by

$$\|A_{k,j} - \tilde{A}_{k,j}\| = \left\| \prod_{t=1}^{n} \left(\cos\theta_t - \cos\tilde{\theta}_t\right) \cdot \delta_{t,k,j} + \left(\sin\theta_t - \sin\tilde{\theta}_t\right) \cdot (1 - \delta_{t,k,j}) \right\|.$$

Let $a_t = \cos\theta_t \cdot \delta_{t,k,j} + \sin\theta_t \cdot (1 - \delta_{t,k,j})$ and $b_t = \cos\tilde{\theta}_t \cdot \delta_{t,k,j} + \sin\tilde{\theta}_t \cdot (1 - \delta_{t,k,j})$ for $t = 1, \ldots, n$. Then

$$\prod_{t=1}^{n} a_t - \prod_{t=1}^{n} b_t = \sum_{t=1}^{n} (a_t - b_t) \left(\prod_{m=1}^{t-1} a_t\right) \left(\prod_{m=t}^{n} b_t\right).$$

Applying the triangle inequality, we have

$$\|A_{k,j} - \tilde{A}_{k,j}\| \leq \sum_{t=1}^{n} \left| \left(\cos\theta_t - \cos\tilde{\theta}_t\right) \cdot \delta_{t,k,j} + \left(\sin\theta_t - \sin\tilde{\theta}_t\right) \cdot (1 - \delta_{t,k,j}) \right|$$

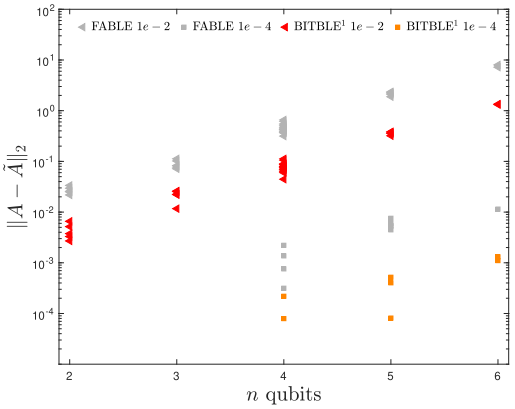


FIG. 11. Error on sample data for generated random real matrix by BITBLE[1] and FABLE [12] with threshold values $\delta_c = 1e-2$ and $\delta_c = 1e-4$.

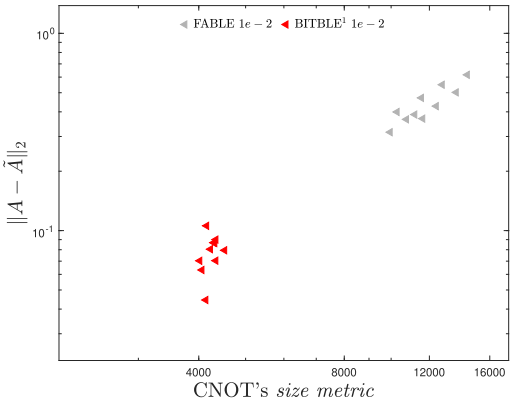


FIG. 12. Approximation error and CNOT's "size metric" of BITBLE[1] and FABLE [12] protocol encoding generated random matrix in $\mathbb{R}^{16\times16}$ with a threshold value $\delta_c = 1e-2$.

$$\times \prod_{m=1}^{t-1} \left|\cos\theta_t \cdot \delta_{t,k,j} + \sin\theta_t \cdot (1-\delta_{t,k,j})\right|$$
$$\times \prod_{m=t}^{n} \left|\cos\tilde{\theta}_t \cdot \delta_{t,k,j} + \sin\tilde{\theta}_t \cdot (1-\delta_{t,k,j})\right|$$
$$\leq \sum_{t=1}^{2n} s \Big| \left(\cos\theta_t - \cos\tilde{\theta}_t\right) \cdot \delta_{t,k,j} + \left(\sin\theta_t - \sin\tilde{\theta}_t\right) \cdot (1-\delta_{t,k,j}) \Big|$$
$$\leq 2n \max_t |\sin(\delta\theta_t/2)| \approx n2^{2n}\delta_c + \mathcal{O}(n\delta_c^3).$$

Finally, $\|A - \tilde{A}\|_2 \leq \|\delta A\|_F \approx n2^{3n}\delta_c + \mathcal{O}(n2^n\delta_c^3)$. □

Fig. 11 illustrates the relationship between the approximation error and matrices' dimensions for the BITBLE[1] and FABLE [12] protocols, and Fig. 12 illustrates the relationship between the approximation error and the CNOT's "size metric" within the same cutoff threshold value setting. Both diagrams show that BITBLE[1] exhibits greater robustness to single-qubit gate cutoff errors compared to FABLE.

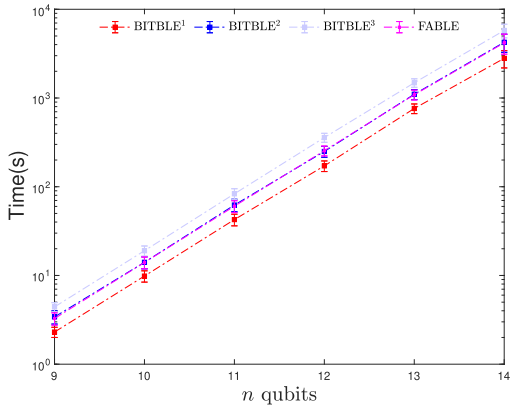


FIG. 13. Compilation time of encoding generated random real matrices using different protocols with 90% confidence intervals. Less time is preferable.

V. NUMERICAL EXPERIMENTS

In this section, some numerical experiments will be provided to demonstrate the efficiency and compiling performance. The experiments in this section explore the time of compilation and "size metric" of BITBLE protocol with different decoupling methods and subnormalization factors.

The numerical experiments were conducted on a Windows personal computer equipped with an Intel i7-12700 processor and 32 GB of RAM. In the following experiments, BITBLE[3] protocol set parameter $p = 0.5$.

A. COMPILING TIME OF RANDOM MATRICES

In the numerical experiments, we compare the compilation time of encoding random matrices using the following methods:

1) BITBLE[1]-BITBLE[3] (our proposed protocols);
2) FABLE [12];
3) Qiskit's unitary synthesis [32], [43].

Since Qiskit [43] does not provide a direct command for block encoding. To benchmark the compilation time for encoding a $2^n \times 2^n$ random matrix by Qiskit, we generate a random unitary matrix of size $2^{n+1} \times 2^{n+1}$ and the top-left $2^n \times 2^n$ submatrix of that unitary matrix serves as our target random matrix. In our Qiskit implementation, we measure the compilation time required to encode the full $2^{n+1} \times 2^{n+1}$ unitary matrix using Qiskit's unitary synthesis functionality.

Numerical experiments of the compilation time for encoding random matrices are presented in Table 4 and Fig. 13. Among these compiling methods, BITBLE[1] achieves the fastest compilation time to encode a $2^n \times 2^n$ random matrix. Although decoupling multiplexor operations can be implemented by parallel computation in BITBLE[1], our experiments were conducted in serial computing. Even under serial computation, the recursive demultiplexor method demonstrates significant advantages.

IEEE Transactions on Quantum Engineering

B. SIZE METRIC OF BLOCK ENCODING

In order to assess the efficacy of various block-encoding protocols, a series of numerical experiments were performed in the following domains: 1) single-channel images; 2) discretized Laplacian operators as presented in 1-D/2-D stemming from partial differential equations; 3) Hamiltonians of Heisenberg-type spin chains; and 4) the Fermi–Hubbard model. The metric parameter referred to as "size metric" in (1) is

$$\text{size metric} = \text{gate}'\textit{scount} \times \text{subnormalization}.$$

1) ENCODING IMAGES

A comparative analysis of CNOT count "size metric" and circuit compilation time across three quantum compilation techniques: the BITBLE protocols, the FABLE protocol (the QPIXL framework [40]), and Qiskit's unitary synthesis is conducted. To assess the encoding efficacy of Qiskit [43] concerning image and discrete operator encoding, the unitary synthesis [32] of Qiskit is employed to encode a unitary. In this approach, the target matrix, denoted as A, is integrated into the top-left block as follows:

$$\begin{bmatrix} A & \sqrt{I - AA^*} \\ -\sqrt{I - A^*A} & A^* \end{bmatrix}.$$

We explore the following two distinct scenarios:

1) encoding a "Peppers.png" color image, where each channel (Red, Green, and Blue) is encoded independently. The CNOT gate's "size metric" and compilation time for this image are presented in Fig. 14;
2) encoding grayscale images from the FASHION MNIST dataset [44]. We evaluate the same performance index for composite images with sizes ranging from 10×10 to 80×80 images, each image has 28×28 pixels, as shown in Fig. 15.

In the context of encoding the "Peppers.png" color image, the unitary synthesis method implemented by Qiskit achieves a lower CNOT's "size metric" for the Red channel than BITBLE[1]; however, it entails a 300× longer compilation time relative to BITBLE[1]. Conversely, the BITBLE[1] method surpasses Qiskit and FABLE in both compilation time and "size metric" when encoding the Green and Blue channels. When considering the encoding of the composite Fashion MNIST dataset, BITBLE[1] exhibits considerable advantages over FABLE, notably requiring 35% less compilation time and reducing the CNOT count by more than 75%. It is important to note that the unitary synthesis by Qiskit faces memory constraints when handling unitary matrices larger than 1028×1028 on a 32-GB RAM machine; consequently, data related to Qiskit's unitary synthesis for encoding FASHION MNIST images have not been presented.

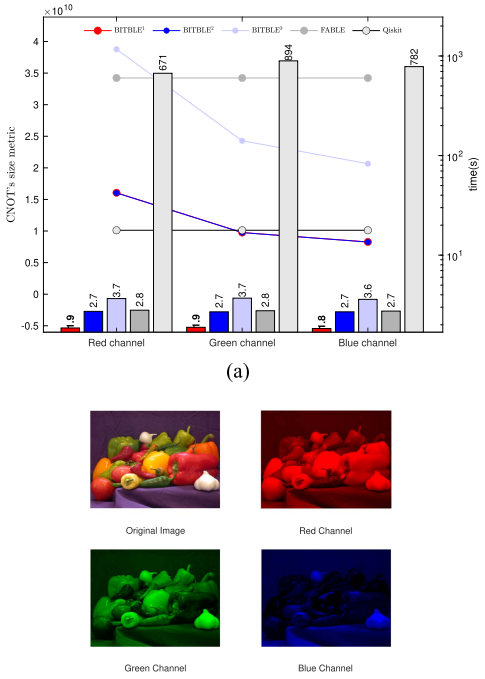


FIG. 14. The CNOT's "size metric" and the average compiling time for encoding RGB channels of "Peppers.png" using the BITBLE[1] protocol. (a) Lines show the CNOT's "size metric" and the bars show the average compiling time for encoding RGB channels. Lower values for both the "size metric" and time are preferable. (b) RGB channels of "Peppers.png."

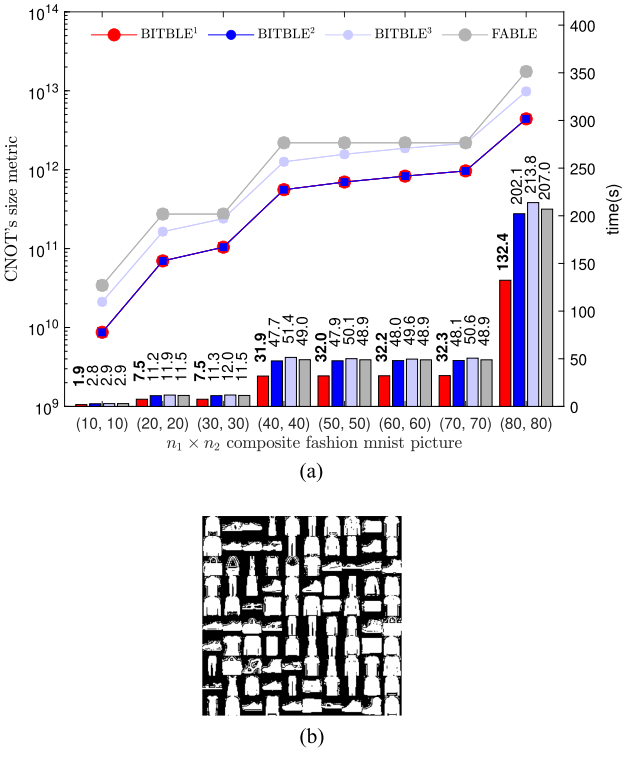


FIG. 15. CNOT's "size metric" and the compiling time for encoding composite FASHION MNIST images using the BITBLE[1] protocol. (a) CNOT's "size metric" and the average compiling time for encoding composite FASHION MNIST images. (b) 10 × 10 composite FASHION MNIST images.

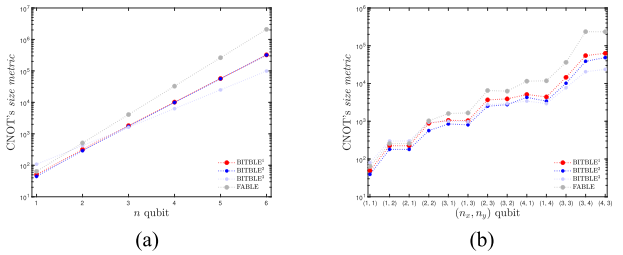


FIG. 16. CNOT's "size metric" for encoding the Laplacian in 1-D and 2-D elliptic partial differential equations using BITBLE and FABLE protocols. Lower "size metric" is preferable. (a) 1-D nonperiodic boundary condition. (b) 2-D periodic boundary condition.

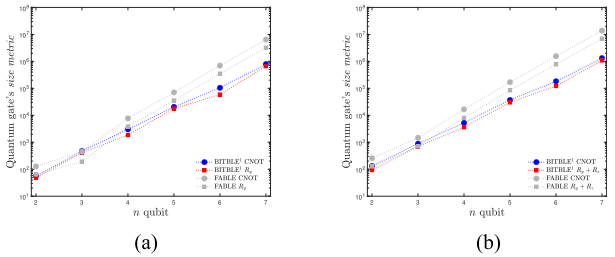


FIG. 17. Quantum gate's "size metric" for encoding Heisenberg-type spin chain Hamiltonians. (a) $Jx = Jy = Jz = 1$, $h_z = 0$. (b) $Jx = Jy = Jz = 1i$, $h_z = 0$.

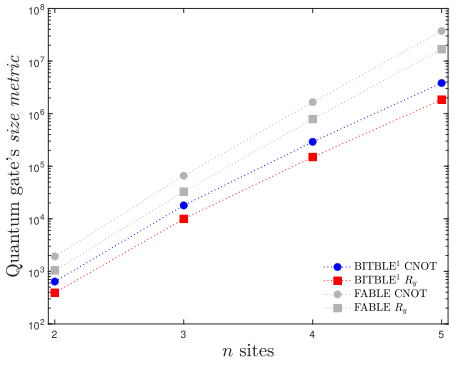


FIG. 18. Quantum gate's "size metric" for encoding the Fermi–Hubbard model.

2) ENCODING ELLIPTIC PARTIAL DIFFERENTIAL OPERATOR

The 1-D discretized Laplace operator is of the form

$$L = \begin{bmatrix} 2 & -1 & 0 & \cdots & * \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ & \cdots & 0 & -1 & 2 \end{bmatrix}.$$

For nonperiodic boundary conditions, the entries referred to as $*$ within the operator are set to 0. On the other hand, for periodic boundary conditions, the entries labeled as $*$ are set to -1. In situations involving 2-D case, an (n_x, n_y)-qubit discretized Laplace operator [4], [12] can be defined as follows:

$$L = L_{xx} \oplus L_{yy} = L_{xx} \otimes I_{n_y} + I_{n_x} \otimes L_{yy}$$

where L_{xx} and L_{yy} denote n_x-qubit and n_y-qubit operators, respectively. Fig. 16 illustrates the "size metric" of CNOT and single-qubit rotation gates of BITBLE and FABLE circuits for both 1-D Laplace operators with nonperiodic boundary conditions and 2-D Laplace operators with periodic boundary conditions.

For encoding 1-D and 2-D discretized Laplace operators, the proposed BITBLE protocol achieves a smaller CNOT's "size metric" compared to the FABLE protocol. In particular, the BITBLE^3 protocol with subnormalization factor $\mu_p(A)$ exhibits the lowest CNOT's "size metric"—outperforming BITBLE^1, BITBLE^2, and FABLE. This protocol reduces the "size metric" in terms of CNOT gates by over 90% compared to the FABLE protocol [12] in systems with more than four qubits.

3) ENCODING HEISENBERG-TYPE SPIN CHAINS HAMILTONIANS

The Heisenberg-type spin chain Hamiltonians are localized Hamiltonians, described mathematically as follows:

$$\sum_{i=1}^{n-1} J_x X^{(i)} X^{(i+1)} + J_y Y^{(i)} Y^{(i+1)} + J_z Z^{(i)} Z^{(i+1)} + \sum_{i=1}^{n} h_z Z^{(i)}$$

where $X^{(i)}$, $Y^{(i)}$, and $Z^{(i)}$ represent the Pauli-X, Pauli-Y, and Pauli-Z matrices acting on the ith qubit. The numerical experiments consider two cases

$$J_x = J_y = J_z = 1,\ h_z = 0$$
$$J_x = J_y = J_z = 1i,\ h_z = 0.$$

The "size metric" of CNOT and single-qubit rotation gates of the above two Hamiltonians for systems of $2, \ldots, 7$ qubits is shown in Fig. 17.

4) ENCODING FERMI–HUBBARD MODEL

The Hamiltonian in the Fermi–Hubbard model is given by

$$H = -t \sum_{ij} \sum_{\sigma \in \{\uparrow,\downarrow\}} c_{i\sigma}^\dagger c_{j\sigma} + U \sum_i c_{i\uparrow}^\dagger c_{j\uparrow} c_{i\downarrow}^\dagger c_{j\downarrow}$$

where $c_{i\sigma}^\dagger c_{j\sigma}$ is the creation (annihilation) operator for site i and spin σ, which can be generated by a software—OpenFermion [45]. We consider the Hamiltonian generated by the Bravyi–Kitaev transformation for the case of $t = 1$, $U = 0$, which corresponds to 1-D and 2-D lattices with two spin orbitals per site and nonperiodic boundary conditions. The n-site 1-D Hubbard model corresponds to a $2n$-qubit Hamiltonian. The CNOT's "size metric" of that model with four to ten qubits is given by Fig. 18.

Based on the aforementioned experiments, the BITBLE protocol achieves a faster time of quantum compiling and better "size metric" of CNOT gates compared to FABLE and Qiskit's unitary synthesis for encoding high-dimensional physical Hamiltonians and natural images in many cases.

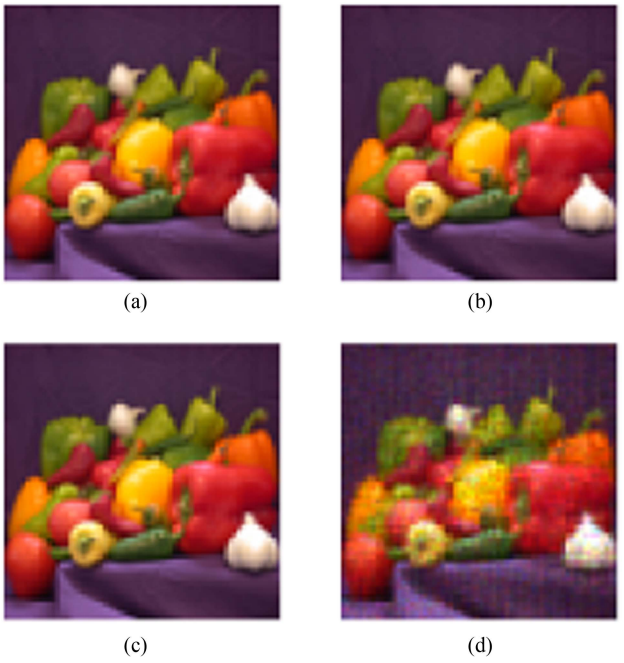


FIG. 19. Sample outputs of the picture encoding from BITBLE. (a) Original figure size of 64 × 64 × 3. (b) Threshold value: $1e-6$, average PSNR: 293.9 dB. (c) Threshold value $1e-4$, average PSNR: 82.5 dB. (d) Threshold value $1e-2$, average PSNR: 27.5 dB.

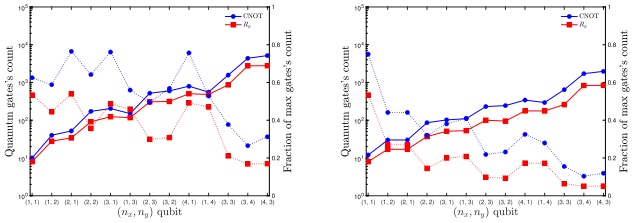


FIG. 20. Compression performance of the BITBLE[1] protocol. The number of gates in the full line is shown on the left axis, while the fraction of the maximum CNOT gates in the dotted line is shown on the right axis. The maximum gate count for the BITBLE[1] protocol to encode a $2^n \times 2^n$ real matrix is $(2^{2n}-1)$ R_y gates and $(2^{2n}+2^n-4)$ CNOT gates. (a) 2-D nonperiodic boundary condition. (b) 2-D periodic boundary condition.

C. COMPRESSION PERFORMANCE

The compression function of the BITBLE protocol resembles FABLE, eliminating single-qubit rotations with parameters below a threshold and reducible CNOT gates. The compression performance of BITBLE is evaluated by encoding the pepper figure and Laplace operators as shown in Figs. 19 and 20. For encoding the pepper figure, the original size of 384 × 512 × 3 uint8 is resized to 64 × 64 × 3 double, and rotation angles below the threshold are removed for each color channel. The fraction of max single-qubit and CNOT gates, average peak signal-to-noise ratio (PSNR), and the output of different threshold values by BITBLE[1] are shown in Table 5 and Fig. 19.

For encoding Laplace operators, the BITBLE protocol achieves excellent results for 2-D discretized Laplace operators with periodic boundary conditions, requiring only

TABLE 5. Fraction of Max Single-Qubit Gate (1.0 Means No Compression), Fraction of Max CNOT Gate, and PNSR of Encoding Pepper's Figure With Different Cutoff Threshold Values

Threshold Value	Single-qubit Gate	CNOT	PNSR (dB)
$1e-6$	1.0	1.0	293.88
$1e-4$	0.9876	0.9998	82.45
$1e-2$	0.2751	0.4535	27.38

10–40% fraction of the maximum CNOT's count for systems with more than five qubits ($n_x + n_y \geq 5$), as shown in Fig. 20(b).

VI. CONCLUSION

In this article, we introduced a block-encoding protocol—BITBLE—for generating quantum circuits that encode arbitrary target matrices. We proposed two decoupling multiplexor operation methods, permutative demultiplexor and recursive demultiplexor, to reduce the compilation time of decoupling multiplexor operations, which serve as a subroutine in certain block-encoding protocols. Based on these two decoupling methods, the time of circuit synthesis and the subnormalization factor for encoding classical matrices and state preparation have been optimized.

We evaluated the effectiveness of BITBLE protocols using the "*size metric*," compilation time, and approximation error. Our proposed protocol, BITBLE[1], demonstrated superior performance, achieving both the fastest compilation time and lowest "size metric" (for some cases) compared to FABLE and Qiskit's unitary synthesis. This advantage was particularly pronounced for high-dimensional matrices, including structured matrices arising in partial differential equations, physical Hamiltonians, and natural image encoding. Notably, even in serial implementations of the *recursive demultiplexor*, BITBLE[1] showed promising results, with the potential for further speedup through parallel computation.

A key future direction involves developing fast compilation methods for large-scale quantum circuits, particularly for state preparation, unitary synthesis, and block-encoding protocols.

APPENDIX A
PARAMETERS' COMPUTING ALGORITHM

The algorithms of R_Y-angles() and R_Z-angles() are stated in Algorithms 2 and 3, respectively. The complete algorithms for computing parameters in the BITBLE protocol with subnormalization $\|A\|_F$ (BITBLE[1] and BITBLE[2]) are stated in Algorithm 4. Besides, the protocol with subnormalization $\mu_p(A)$ (BITBLE[3]) is stated in Algorithm 5.

APPENDIX B
DEMO OF BITBLE PROGRAM

All codes are open source and can be found online.[2] Some circuit demonstrations are given in Figs. 21 and 22. The following is a demonstration of the code.

1) Quantum state preparation:

[2]https://github.com/zexianLIPolyU/BITBLE

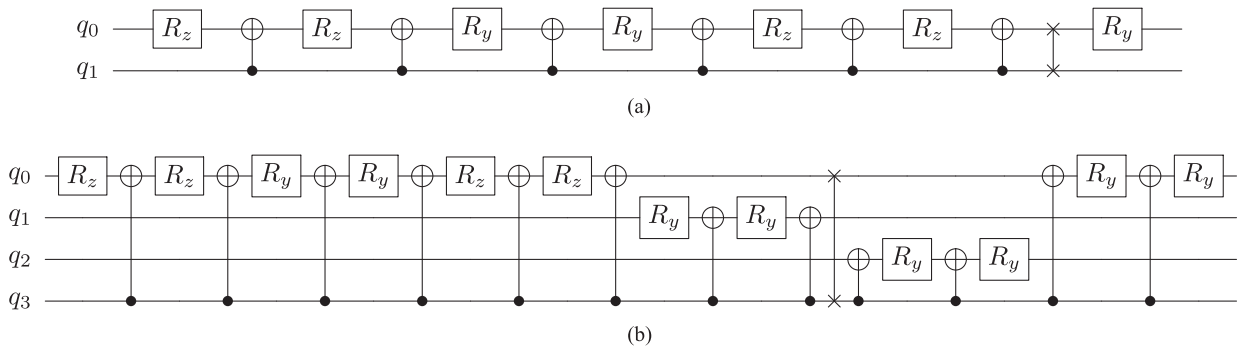


FIG. 21. Quantum circuit realizes BITBLE protocols encoding 2 × 2 complex matrix. (a) Output of $\texttt{BITBLE}^1$. (b) Output of $\texttt{BITBLE}^3$.

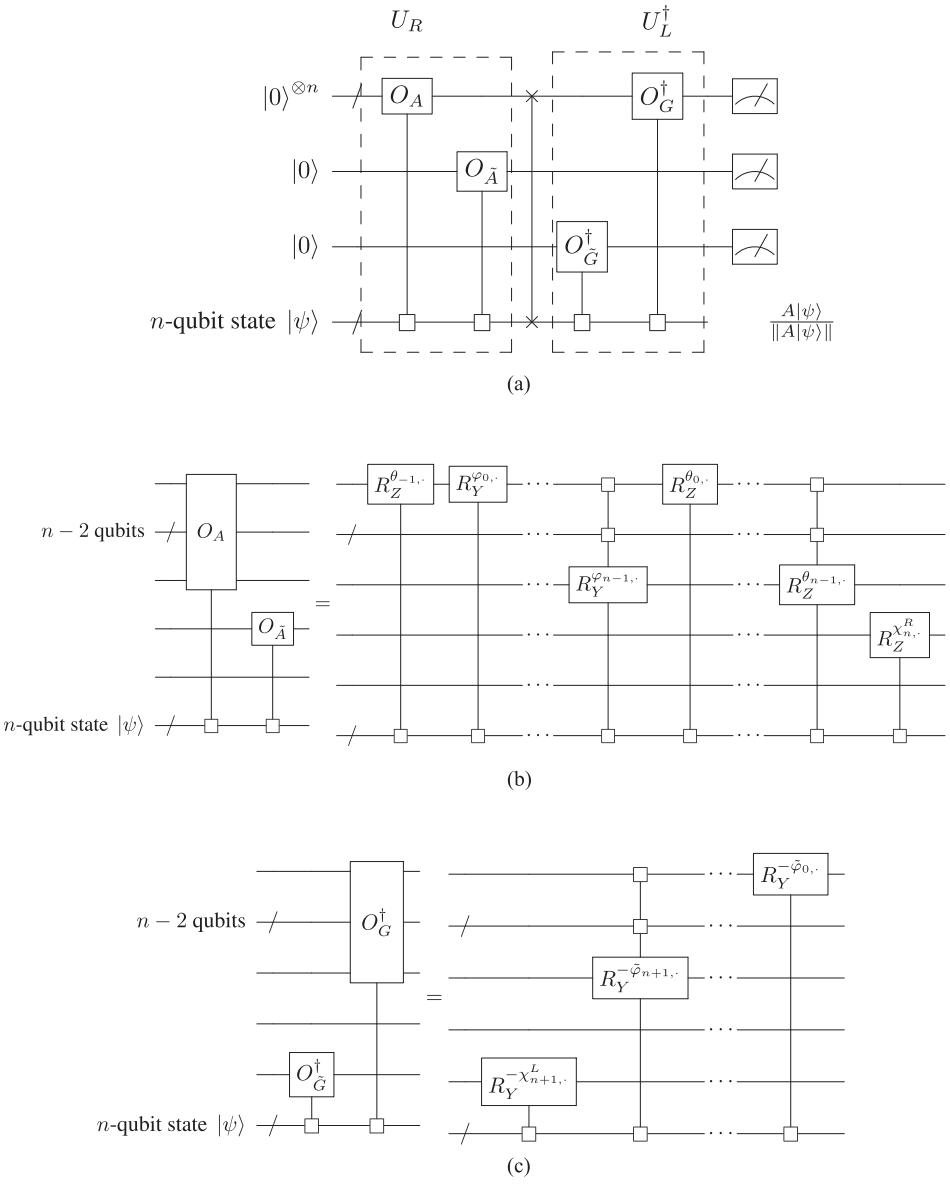


FIG. 22. Quantum circuit realizes block encoding with subnormalization factor $\mu_P(A)$ ($\texttt{BITBLE}^3$). (a) Circuit frameworks of $\texttt{BITBLE}^3$. (b) State preparation unitary U_R. (c) Inverse state preparation unitary ($U_L^\dagger$).

Algorithm 4: Compute Multiplexor Operation Parameters With Subnormalization $\|A\|_F$ (BITBLE[1] and BITBLE[2]).

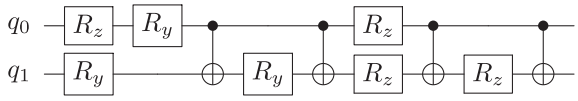
Require $[A_{k,j} := e^{i\phi_{k,j}}|A_{k,j}|]_{0\le k,j\le 2^n-1}$
Ensure Compute parameters of the BITBLE protocol with a subnormalization factor $\|A\|_F$ in Fig. 10.
1: **for** $j = 1, \ldots, 2^n$
2: $[\varphi_{k,j}]_{k=0}^{2^n-2} \leftarrow$ R$_Y$-angles$([|A_{k,j}|]_{k=0}^{2^n-1})$
3: **if** A is complex
4: $([\theta_{k,j}]_{k=0}^{2^n-2}, \theta_{-1,j}) \leftarrow$ R$_Z$-angles$([\phi_{k,j}]_{k=0}^{2^n-1})$
5: **end if**
6: **end for**
7: $[\varphi'_j]_{j=0}^{2^n-2} \leftarrow$ R$_Y$-angles$(\{\|A_{\cdot,j}\|_2/\|A\|_F\}_{j=0}^{2^n-1})$
8: $\hat{\varphi}'_{k,j} \leftarrow \varphi'_{2^k+j-1}$ for all $0 \le k \le n-1$, $0 \le j \le 2^k - 1$

Algorithm 5: Compute Multiplexor Operation Parameters With a Subnormalization Factor $\mu_p(A)$ (BITBLE[3]).

Require $[A_{k,j} := e^{i\phi_{k,j}}|A_{k,j}|]_{0\le k,j\le 2^n-1}$ and $p \in [0, 1]$
Ensure Compute parameters of the BITBLE[3] protocol with subnormalization $\mu_p(A)$.
1: **for** $j = 1, \ldots, 2^n$
2: $[\varphi_{k,j}]_{k=0}^{2^n-2} \leftarrow$ R$_Y$-angles$([\frac{|A_{k,j}|^p}{\sqrt{\|A_{\cdot,j}\|_{2p}^{2p}}}]_{k=0}^{2^n-1})$
3: $[\tilde{\varphi}_{k,j}]_{k=0}^{2^n-2} \leftarrow$ R$_Y$-angles$([\frac{|A_{j,k}|^{1-p}}{\sqrt{\|A_{k,\cdot}\|_{2(1-p)}^{2(1-p)}}}]_{k=0}^{2^n-1})$
4: **if** A is complex
5: $([\theta_{k,j}]_{k=0}^{2^n-2}, \theta_{-1,j}) \leftarrow$ R$_Z$-angles$([\phi_{k,j}]_{k=0}^{2^n-1})$
6: **end if**
7: **end for**
8: $[\chi_{n,j}^R]_{j=0}^{2^n-1} \leftarrow$ R$_Y$-angles$([\arccos(\sqrt{\frac{\|A_{\cdot,j}\|_{2p}^{2p}}{S_{2p}(A^T)}})]_{j=0}^{2^n-1})$
9: $[\chi_{n+1,j}^L]_{j=0}^{2^n-1} \leftarrow$ R$_Y$-angles$([\arccos(\sqrt{\frac{\|A_{k,\cdot}\|_{2(1-p)}^{2(1-p)}}{S_{2(1-p)}(A)}})]_{k=0}^{2^n-1})$

```
cd("state_preparation");
addpath('QCLAB');
logging=true; %r e c o r d
circuit_sim=true;
N=4;
state_complex=randn(N,1)+randn(N,1).*1i;
n=log2(N);
state_complex=state_complex . . .
./norm(state_complex);
[circuit0,info0]= . . .
binary_tree_statepreparation( . . .
state_complex,logging,circuit_sim);
circuit0.draw();
```

Output:



2) Binary tree block encoding (BITBLE[1] *) using recursive demultiplexor:*

```
cd("bitble-qclab");
addpath('QCLAB');
n=1;
A=randn(pow2(n),pow2(n))+ . . .
randn(pow2(n),pow2(n)).*1j;
offset=0;
logging=true;
compr_type='\hl{cutoff}';
compr_val=1e-8;
circuit_sim=true;
[circuit1,normalization_factor1,info1] . . .
=bitble(A,compr_type,compr_val, . . .
logging,offset,circuit_sim);
circuit1.draw();
```

3) Binary tree block encoding (BITBLE[2] *) using recursive demultiplexor:*

```
cd("bitble-qclab");
addpath('QCLAB');
n=1;
A=randn(pow2(n),pow2(n))+ . . .
randn(pow2(n),pow2(n)).*1j;
offset=0;
logging=true;
compr_type='\hl{cutoff}';
compr_val=1e-8;
circuit_sim=true;
[circuit2,normalization_factor2,info2] . . .
=bitble2(A,compr_type,compr_val, . . .
logging,offset,circuit_sim);
circuit2.draw();
```

4) Binary tree block encoding (BITBLE[3] *) using permutative demultiplexor with subnormalization factor $\mu_p(A)$*

```
cd("bitble-qclab");
addpath('QCLAB');
n=1;
A=randn(pow2(n),pow2(n))+ . . .
randn(pow2(n),pow2(n)).*1j;
offset=0;
logging=true;
compr_type='\hl{cutoff}';
compr_val=1e-8;
circuit_sim=true;
[circuit3,normalization_factor3,info3] . . .
=bitble3(A,compr_type,compr_val, . . .
logging,offset,circuit_sim);
circuit3.draw();
```

REFERENCES

[1] G. H. Low and I. L. Chuang, "Optimal Hamiltonian simulation by quantum signal processing," *Phys. Rev. Lett.*, vol. 118, Jan. 2017, Art. no. 010501, doi: 10.1103/PhysRevLett.118.010501.

[2] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, "Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics," in *Proc. 51st Annu. ACM SIGACT Symp. Theory Comput.*, 2019, pp. 193–204, doi: 10.1145/3313276.3316366.

[3] J. M. Martyn, Z. M. Rossi, A. K. Tan, and I. L. Chuang, "Grand unification of quantum algorithms," *PRX Quantum*, vol. 2, Dec. 2021, Art. no. 040203, doi: 10.1103/PRXQuantum.2.040203.

[4] N. Klco and M. J. Savage, "Digitization of scalar fields for quantum computing," *Phys. Rev. A*, vol. 99, no. 5, 2019, Art. no. 052335, doi: 10.1103/PhysRevA.99.052335.

[5] L. Wossnig, Z. Zhao, and A. Prakash, "Quantum linear system algorithm for dense matrices," *Phys. Rev. Lett.*, vol. 120, Jan. 2018, Art. no. 050502, doi: 10.1103/PhysRevLett.120.050502.

[6] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, Oct. 2009, Art. no. 150502, doi: 10.1103/PhysRevLett.103.150502.

[7] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, Aug. 2018, Art. no. 79, doi: 10.22331/q-2018-08-06-79.

[8] Y. Dong, L. Lin, and Y. Tong, "Ground-state preparation and energy estimation on early fault-tolerant quantum computers via quantum eigenvalue transformation of unitary matrices," *PRX Quantum*, vol. 3, Oct. 2022, Art. no. 040305, doi: 10.1103/PRXQuantum.3.040305.

[9] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, "High-threshold and low-overhead fault-tolerant quantum memory," *Nature*, vol. 627, no. 8005, pp. 778–782, Mar. 2024, doi: 10.1038/s41586-024-07107-7.

[10] T. J. Yoder et al., "Tour de gross: A modular quantum computer based on bivariate bicycle codes," 2025, *arXiv:2506.03094*, doi: 10.48550/arXiv.2506.03094.

[11] M. Ryan et al., "How IBM will build the World's first large-scale, fault-tolerant quantum computer," 2025. [Online]. Available: https://www.ibm.com/quantum/blog/large-scale-ftqc#fn-2

[12] D. Camps and R. Van Beeumen, "FABLE: Fast approximate quantum circuits for block-encodings," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, Sep. 2022, pp. 104–113, doi: 10.1109/QCE53715.2022.00029.

[13] G. H. Low and I. L. Chuang, "Hamiltonian simulation by qubitization," *Quantum*, vol. 3, Jul. 2019, Art. no. 163, doi: 10.22331/q-2019-07-12-163.

[14] J. van Apeldoorn and A. Gilyén, "Improvements in quantum SDP-Solving with applications," in *Proc. 46th Int. Colloq. Automata, Lang., Programming/Leibniz Int. Proc. Informat.*, 2019, pp. 1–15, doi: 10.4230/LIPIcs.ICALP.2019.99.

[15] Q. T. Nguyen, B. T. Kiani, and S. Lloyd, "Block-encoding dense and full-rank kernels using hierarchical matrices: Applications in quantum numerical linear algebra," *Quantum*, vol. 6, Dec. 2022, Art. no. 876, doi: 10.22331/q-2022-12-13-876.

[16] H. Li, H. Ni, and L. Ying, "On efficient quantum block encoding of pseudo-differential operators," *Quantum*, vol. 7, Jun. 2023, Art. no. 1031, doi: 10.22331/q-2023-06-02-1031.

[17] N. Guseynov, X. Huang, and N. Liu, "Gate construction of block-encoding for Hamiltonians needed for simulating partial differential equations," *Phys. Rev. Res.*, vol. 7, Jul. 2025, Art. no. 033100, doi: 10.1103/xlpd-fb1g.

[18] T. Kharazi, A. M. Alkadri, J.-P. Liu, K. K. Mandadapu, and K. B. Whaley, "Explicit block encodings of boundary value problems for many-body elliptic operators," *Quantum*, vol. 9, Jun. 2025, Art. no. 1764, doi: 10.22331/q-2025-06-04-1764.

[19] D. Liu, W. Du, L. Lin, J. P. Vary, and C. Yang, "An efficient quantum circuit for block encoding a pairing Hamiltonian," *J. Comput. Sci.*, vol. 85, 2025, Art. no. 102480, doi: 10.1016/j.jocs.2024.102480.

[20] D. Camps, L. Lin, R. Van Beeumen, and C. Yang, "Explicit quantum circuits for block encodings of certain sparse matrices," *SIAM J. Matrix Anal. Appl.*, vol. 45, no. 1, pp. 801–827, 2024, doi: 10.1137/22M1484298.

[21] C. Sünderhauf, E. Campbell, and J. Camps, "Block-encoding structured matrices for data input in quantum computing," *Quantum*, vol. 8, Jan. 2024, Art. no. 1226, doi: 10.22331/q-2024-01-11-1226.

[22] C. Yang, Z. Li, H. Yao, Z. Fan, G. Zhang, and J. Liu, "Dictionary-based block encoding of sparse matrices with low subnormalization and circuit depth," *Quantum*, vol. 9, Jul. 2025, Art. no. 1805, doi: 10.22331/q-2025-07-22-1805.

[23] B. D. Clader, A. M. Dalzell, N. Stamatopoulos, G. Salton, M. Berta, and W. J. Zeng, "Quantum resources required to block-encode a matrix of classical data," *IEEE Trans. Quantum Eng.*, vol. 3, no. 1, 2022, Art. no. 3103323, doi: 10.1109/TQE.2022.3231194.

[24] C. T. Hann, G. Lee, S. Girvin, and L. Jiang, "Resilience of quantum random access memory to generic noise," *PRX Quantum*, vol. 2, Apr. 2021, Art. no. 020311, doi: 10.1103/PRXQuantum.2.020311.

[25] X.-M. Zhang and X. Yuan, "Circuit complexity of quantum access models for encoding classical data," *npj Quantum Inf.*, vol. 10, no. 1, Apr. 2024, Art. no. 42, doi: 10.1038/s41534-024-00835-8.

[26] P. Yuan and S. Zhang, "Optimal (controlled) quantum state preparation and improved unitary synthesis by quantum circuits with any number of ancillary qubits," *Quantum*, vol. 7, Mar. 2023, Art. no. 956, doi: 10.22331/q-2023-03-20-956.

[27] P. Kuklinski and B. Rempfer, "S-FABLE and LS-FABLE: Fast approximate block-encoding algorithms for unstructured sparse matrices," 2024, *arXiv:2401.04234*, doi: 10.48550/arXiv.2401.04234.

[28] D. Camps and R. Van Beeumen, "QCLAB," 2021, version 0.1.3. [Online]. Available: https://github.com/QuantumComputingLab/qclab

[29] I. Kerenidis and A. Prakash, "Quantum recommendation systems," in *Proc. 8th Innov. Theor. Comput. Sci. Conf.*, 2017, vol. 67, pp. 1–21, doi: 10.4230/LIPIcs.ITCS.2017.49.

[30] S. Chakraborty, A. Gilyén, and S. Jeffery, "The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation," in *Proc. 46th Int. Colloq. Automata, Lang., Program.*, 2019, vol. 132, pp. 33:1–33:14, doi: 10.4230/LIPIcs.ICALP.2019.33.

[31] I. Kerenidis and A. Prakash, "Quantum gradient descent for linear systems and least squares," *Phys. Rev. A*, vol. 101, Feb. 2020, Art. no. 022316, doi: 10.1103/PhysRevA.101.022316.

[32] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum logic circuits," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2005, pp. 272–275, doi: 10.1145/1120725.1120847.

[33] L. Grover and T. Rudolph, "Creating superpositions that correspond to efficiently integrable probability distributions," 2002, *arXiv:quant-ph/0208112*, doi: 10.48550/arXiv.quant-ph/0208112.

[34] I. F. Araujo, D. K. Park, T. B. Ludermir, W. R. Oliveira, F. Petruccione, and A. J. da Silva, "Configurable sublinear circuits for quantum state preparation," *Quantum Inf. Process.*, vol. 22, no. 2, Feb. 2023, Art. no. 123, doi: 10.1007/s11128-023-03869-7.

[35] Z. Li, X.-M. Zhang, C. Yang, and G. Zhang, "Binary tree block encoding of classical matrix," 2025, *arXiv:2504.05624*, doi: 10.48550/arXiv.2504.05624.

[36] J. A. de Carvalho, C. Batista, T. de Veras, I. Araujo, and A. J. da Silva, "Quantum multiplexer simplification for state preparation," *ACM Trans. Quantum Comput.*, vol. 6, no. 4, Aug. 2025, Art. no. 26, doi: 10.1145/3748260.

[37] S. Chakrabarti, A. M. Childs, T. Li, and X. Wu, "Quantum algorithms and lower bounds for convex optimization," *Quantum*, vol. 4, Jan. 2020, Art. no. 221, doi: 10.22331/q-2020-01-13-221.

[38] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Quantum circuits for general multiqubit gates," *Phys. Rev. Lett.*, vol. 93, Sep. 2004, Art. no. 130502, doi: 10.1103/PhysRevLett.93.130502.

[39] C. Savage, "A survey of combinatorial Gray codes," *SIAM Rev.*, vol. 39, no. 4, pp. 605–629, 1997, doi: 10.1137/S0036144595295272.

[40] M. G. Amankwah, D. Camps, E. W. Bethel, R. Van Beeumen, and T. Perciano, "Quantum pixel representations and compression for n-dimensional images," *Sci. Rep.*, vol. 12, no. 1, May 2022, Art. no. 7712, doi: 10.1038/s41598-022-11024-y.

[41] Fino and Algazi, "Unified matrix treatment of the fast Walsh-Hadamard transform," *IEEE Trans. Comput.*, vol. C- 25, no. 11, pp. 1142–1146, Nov. 1976, doi: 10.1109/TC.1976.1674569.

[42] J. Alman and K. Rao, "Faster Walsh-Hadamard and discrete Fourier transforms from matrix non-rigidity," in *Proc. 55th Annu. ACM Symp. Theory Comput.*, 2023, pp. 455–462, doi: 10.1145/3564246.3585188.

[43] A. Javadi-Abhari et al., "Quantum computing with qiskit," 2024, *arXiv:2405.08810*, doi: 10.48550/arXiv.2405.08810.

[44] Z. Research, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017. [Online]. Available: https://github.com/zalandoresearch/fashion-mnist

[45] J. R. McClean et al., "OpenFermion: The electronic structure package for quantum computers," 2019, *arXiv:1710.07629*,doi: 10.48550/arXiv.1710.07629.