



**General quantum matrix exponential dimensionality-reduction framework based on block encoding**Yong-Mei Li, Hai-Ling Liu, Shi-Jie Pan, Su-Juan Qin <sup>\*</sup>, Fei Gao,<sup>†</sup> and Qiao-Yan Wen*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China* (Received 29 May 2023; revised 1 September 2023; accepted 26 September 2023; published 19 October 2023)

As a general framework, matrix exponential dimensionality reduction (MEDR) deals with the small-sample-size problem that appears in linear dimensionality-reduction (DR) algorithms. High complexity is the bottleneck in this type of DR algorithm because one has to solve a large-scale matrix exponential generalized eigenproblem. To address it, here we design a general quantum algorithm framework for MEDR based on the block-encoding technique. This framework is configurable, that is, by selecting suitable methods to design the block encodings of the data matrices, a series of efficient quantum algorithms can be derived from this framework. Specifically, by constructing the block encodings of the data matrix exponentials, we solve the generalized eigenproblem and then obtain the digital-encoded quantum state corresponding to the compressed low-dimensional data set, which can be directly utilized as the input state for other quantum machine learning tasks to overcome the curse of dimensionality. As applications, we apply this framework to four linear DR algorithms and design their quantum algorithms, which all achieve a polynomial speedup in the dimension of the sample over their classical counterparts.

DOI: [10.1103/PhysRevA.108.042418](https://doi.org/10.1103/PhysRevA.108.042418)**I. INTRODUCTION**

The power of quantum computing is illustrated by quantum algorithms that solve specific problems, such as factoring [1], unstructured data search [2], linear systems [3], and cryptanalysis [4], much more efficiently than classical algorithms. In recent years, a series of quantum algorithms for solving machine learning problems has been proposed and attracted the attention of the scientific community, such as clustering [5–7], dimensionality reduction [8–11], and matrix computation [12–14]. Quantum machine learning (QML) [15] has appeared as a remarkable emerging direction with great potential in quantum computing.

In the era of big data, dimensionality reduction (DR) has gained significant importance in machine learning and statistical analysis, which is a powerful technique to reveal the intrinsic structure of data and mitigate the effects of the curse of dimensionality [16,17]. The essential task of DR is to find a mapping function  $F : \mathbf{x} \mapsto \mathbf{y}$  that transforms  $\mathbf{x} \in \mathbb{R}^M$  into the desired low-dimensional representation  $\mathbf{y} \in \mathbb{R}^m$ , where typically  $m \ll M$ . Most of the DR algorithms, such as locality preserving projections (LPP) [18], unsupervised discriminant projection (UDP) [19], neighborhood preserving embedding (NPE) [20], and linear discriminant analysis (LDA) [21], were unified into a general graph embedding framework proposed by Yan *et al.* [22]. The framework provides a unified perspective for the understanding and comparison of many popular DR algorithms and facilitates the design of new algorithms. Unfortunately, almost all linear DR (i.e.,  $F$  is a linear function) algorithms in the graph embedding framework

encounter the well-known small-sample-size (SSS) problem that stems from the generalized eigenproblems with singular matrices. To tackle it, a general matrix exponential dimensionality reduction (MEDR) framework [23] was proposed. In the framework, the SSS problem is solved by transforming the original generalized eigenproblem into  $e^{S_1} \mathbf{v} = \lambda e^{S_2} \mathbf{v}$ , where the data matrices  $S_1$  and  $S_2$  have different forms for different algorithms. However, it involves the solution of the large-scale matrix exponential generalized eigenproblem, resulting in the matrix exponential-based DR algorithms being time consuming for a large number of high-dimensional samples. Therefore, it would be of great significance to seek new strategies to speed up this type of DR algorithm.

There exists some work on quantum DR algorithms, which achieve different degrees of acceleration compared with classical algorithms [8–11,24–32]. In the early stages, the work focused on linear DR. Lloyd *et al.* [8] proposed the first quantum DR algorithm, quantum principal component analysis (PCA), which provided an important reference for many subsequent quantum algorithms. Later, the quantum nonlinear DR algorithms gradually emerged based on manifold learning [25,29] and kernel method [28]. However, there is no quantum algorithm that efficiently realizes the matrix exponential-based DR. An interesting question is whether we can design quantum algorithms for this type of DR algorithm in a unified way, which will provide computational benefits and facilitate the design of alternative quantum DR algorithms. We answer the question in the affirmative by using the method of block encoding [33–36].

Block encoding is a good framework for implementing matrix arithmetic on quantum computers. The block encoding  $U$  of a matrix  $A$  is a unitary matrix whose top left block is proportional to  $A$ . Given  $U$ , one can produce the state  $A|\phi\rangle/\|A|\phi\rangle\|$  by applying  $U$  to an initial state  $|0\rangle|\phi\rangle$ . Low

<sup>\*</sup>qsjuan@bupt.edu.cn<sup>†</sup>gaof@bupt.edu.cn

and Chuang [34] showed how to perform optimal Hamiltonian simulation given a block-encoded Hamiltonian  $H$ . Based on this, Chakraborty *et al.* [36] developed several tools within the block-encoding framework, such as singular value estimation and quantum linear system solvers. Moreover, the block-encoding technique has been used to design QML algorithms, such as quantum classification [37] and quantum DR [31].

In this paper we apply the block-encoding technique to MEDR and design a general quantum MEDR (QMEDR) framework. This framework is configurable, that is, for a matrix exponential-based DR algorithm, one can take suitable methods to design the block encodings of  $S_1$  and  $S_2$  and then construct the corresponding quantum algorithm to obtain the compressed low-dimensional data set. Once these two block encodings are implemented efficiently, the quantum algorithm will have a better running time compared to its classical counterpart. More specifically, the main contributions of this paper are as follows.

(a) Given a block-encoded Hermitian  $H$ , we present a method to implement the block encoding of  $e^H$  or  $e^{-H}$  and derive the error upper bound. It provides a way for the computation of the matrix exponential, which is one of the most important tasks in linear algebra [38]. To be specific, depending on the application, the computation of the matrix exponential may be to compute  $e^A$  for a given square matrix  $A$ , to apply  $e^A$  to a vector, and so on [38]. They are very time consuming when  $A$  is an exponentially large matrix. With the help of block encodings, these tasks can be performed much faster on a quantum computer.

(b) By combining the block-encoding technique and quantum phase estimation [39], we solve the generalized eigenproblem  $e^{S_1} \mathbf{v} = \lambda e^{S_2} \mathbf{v}$  and then construct the compressed digital-encoded state [40], i.e., directly encode the compressed low-dimensional data set into qubit strings in quantum parallel. The compressed state can be further utilized as the input state for a variety of QML tasks, such as quantum  $k$ -medoids clustering [41], to overcome the curse of dimensionality. This builds a bridge between quantum DR algorithms and other QML algorithms. In addition, the proposed method for constructing the compressed state can be extended to other linear DR algorithms.

(c) As applications, we apply the QMEDR framework to four matrix exponential-based DR algorithms, i.e., exponential LPP (ELPP) [42], exponential UDP (EUDP) [23], exponential NPE (ENPE) [43], and exponential discriminant analysis (EDA) [44], and design their quantum algorithms. The results show that all of these quantum algorithms achieve a polynomial speedup in the dimension of the sample over the classical algorithms. Moreover, for the number of samples, the quantum ELPP and quantum ENPE algorithms achieve a polynomial speedup over their classical counterparts, and the quantum EDA algorithm provides an exponential speedup.

The remainder of the paper proceeds as follows. In Sec. II we review the MEDR framework in Sec. II A and the block-encoding framework in Sec. II B. In Sec. III we propose the QMEDR framework in Sec. III A and analyze its complexity in Sec. III B. In Sec. IV we present several applications of the QMEDR framework. In Sec. V we discuss the main idea of the QMEDR framework and compare the framework with the related work. A summary is given in Sec. VI.

## II. PRELIMINARIES

### A. MEDR framework

In this section we first review the linear DR in regard to graph embedding and then review the general MEDR framework and analyze its complexity.

Let  $X = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{N-1}]^T \in \mathbb{R}^{N \times M}$  denote the original data matrix. Dimensionality reduction aims to seek an optimal transformation to map the  $M$ -dimensional sample  $\mathbf{x}_i$  onto an  $m$ -dimensional ( $m \ll M$ ) sample  $\mathbf{y}_i$ ,  $i = 0, 1, \dots, N-1$ . Many DR algorithms were unified into a general graph embedding framework [22]. In the framework, an undirected weighted graph  $G = (\mathcal{X}, S)$  with vertex set  $\mathcal{X} = \{\mathbf{x}_i\}_{i=0}^{N-1}$  and similarity matrix  $S = (S_{ij}) \in \mathbb{R}^{N \times N}$  is defined to characterize the original data set, where  $S_{ij}$  measures the similarity of a pair of vertices  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . For the case  $m = 1$ , the graph-preserving criterion is

$$\mathbf{y}^* = \arg \min_{\mathbf{y}^T B \mathbf{y} = d} \sum_{i \neq j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 S_{ij} = \arg \min_{\mathbf{y}^T B \mathbf{y} = d} \mathbf{y}^T L \mathbf{y}, \quad (1)$$

where  $\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T$  is the low-dimensional data matrix (now it is a vector),  $B$  is a constraint matrix (it is Hermitian in general),  $d$  is a constant,  $L = D - S$  is the Laplacian matrix,  $D$  is a diagonal matrix, and  $D_{ii} = \sum_{j \neq i} S_{ij}$ . Note that  $\|\cdot\|$  denotes the  $l_2$ -norm in this paper. Assuming that the low-dimensional vector representations of the vertices can be obtained from a linear projection as  $y_i = \mathbf{w}^T \mathbf{x}_i$ , where  $\mathbf{w}$  is the unitary projection vector, the objective function in Eq. (1) becomes

$$\begin{aligned} \mathbf{w}^* &= \arg \min_{\substack{\mathbf{w}^T X^T B X \mathbf{w} = d \\ \text{or } \mathbf{w}^T \mathbf{w} = d}} \sum_{i \neq j} \|\mathbf{w}^T \mathbf{x}_i - \mathbf{w}^T \mathbf{x}_j\|^2 S_{ij} \\ &= \arg \min_{\substack{\mathbf{w}^T X^T B X \mathbf{w} = d \\ \text{or } \mathbf{w}^T \mathbf{w} = d}} \mathbf{w}^T X^T L X \mathbf{w}. \end{aligned} \quad (2)$$

The solutions of Eqs. (1) and (2) are obtained by solving the generalized eigenproblem

$$S_1 \mathbf{v} = \lambda S_2 \mathbf{v}, \quad (3)$$

where  $S_1 = L$  or  $X^T L X$  and  $S_2 = I$ ,  $B$ , or  $X^T B X$  [22]. Then the optimal projection vector is the generalized eigenvector corresponding to the minimum generalized eigenvalue. Generalized to the case  $m \geq 2$ , let  $\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{m-1}$  be the generalized eigenvectors corresponding to the first  $m$  smallest generalized eigenvalues. Then  $W = [\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{m-1}]$  is the optimal transformation matrix and the desired low-dimensional data matrix  $Y = XW = [y_0, y_1, \dots, y_{N-1}]^T$ .

However, in many cases of real life,  $N < M$ , resulting in the matrices  $S_1$  and  $S_2$  being singular and then Eq. (3) is unsolvable. This is a so-called SSS problem. To solve it, the general MEDR framework is proposed [23]. It replaced the matrices  $S_1$  and  $S_2$  in Eq. (3) with their respective matrix exponentials [38]. In other words, the MEDR framework should solve the generalized eigenproblem

$$e^{S_1} \mathbf{v} = \lambda e^{S_2} \mathbf{v} \quad (4)$$

and take the first  $m$  principal generalized eigenvectors to obtain the optimal transformation matrix. The flowchart of the MEDR framework is shown in Fig. 1.

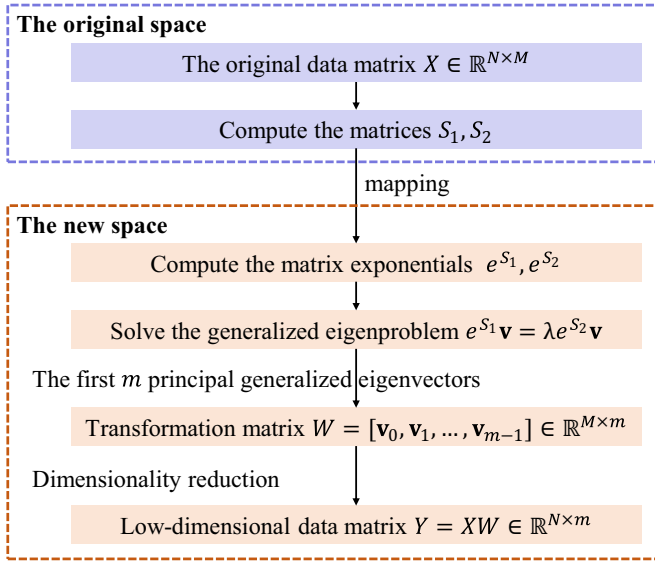


FIG. 1. Flowchart of the general MEDR framework.

The matrix exponential method solves the SSS problem well. Nevertheless, its high complexity constitutes a bottleneck in this type of matrix exponential-based algorithm because one has to compute the multiples and the exponentials of matrices and to solve a matrix exponential generalized eigenproblem. First, for a matrix of the type  $X^T F X$ , the time complexity of computing it is  $O(N^2 M + N M^2)$ , where  $F$  is a square matrix of order  $N$ . Note that although  $N$  is less than  $M$  and can be ignored, we keep  $N$  here for a clearer representation of the parameter relationship. Second, in general, for a given  $M \times M$  matrix, the complexity of computing its matrix exponential is  $O(M^3)$  [38,45]. Third, the complexity of solving the generalized eigenproblem is  $O(M^3)$  for a square matrix of order  $M$ . Adding them up, the total time complexity is  $O(N^2 M + M^3)$ , which introduces difficulty to the practical applications of this type of algorithm when dealing with a larger number of high-dimensional data. Therefore, it is necessary to seek alternative strategies to speedup the matrix exponential-based DR algorithms.

**B. Block-encoding framework**

In this section we briefly review the framework of block encoding.

*Definition 1 (block encoding [35]).* Suppose that  $A$  is an  $s$ -qubit operator,  $\alpha, \epsilon \in \mathbb{R}_+$ , and  $a \in \mathbb{N}$ . Then we say that the  $(s + a)$ -qubit unitary  $U$  is an  $(\alpha, a, \epsilon)$  block encoding of  $A$  if

$$\|A - \alpha(|0\rangle^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)\| \leq \epsilon. \tag{5}$$

Note that since  $\|U\| = 1$ , we necessarily have  $\|A\| \leq \alpha + \epsilon$ . Moreover, the above definition is not restricted to only square matrices. When  $A$  is not a square matrix, we can define an embedding square matrix  $A_e$  such that the top left block of  $A_e$  is  $A$  and all other elements are 0.

There are several methods to implement the block encodings for specific matrices, such as sparse matrices [34,35], density operators [33–35], Gram matrices [35], and matri-

ces stored in structured quantum random access memories (QRAMs) [36,47]. The main motivation for using block encodings is to perform optimal Hamiltonian simulation for a given block-encoded Hamiltonian.

*Theorem 1 (optimal block Hamiltonian simulation [34,46,48]).* Suppose that  $U$  is an  $(\alpha, a, \frac{\epsilon}{|2I|})$  block encoding of the Hamiltonian  $H$ . Then we can implement an  $\epsilon$ -precise Hamiltonian simulation unitary  $V$  which is a  $(1, a + 2, \epsilon)$ -block encoding of  $e^{iHt}$ , with  $O(|\alpha t| + \frac{\log_2(1/\epsilon)}{\log_2 \log_2(1/\epsilon)})$  uses of controlled  $U$  or its inverse and with  $O(a|\alpha t| + a \frac{\log_2(1/\epsilon)}{\log_2 \log_2(1/\epsilon)})$  two-qubit gates.

The block-encoding technique has been applied to many quantum algorithms, such as quantum linear systems [13,34,36] and quantum mean centering [49]. In the following section, we will apply the block-encoding framework to MEDR and design a general quantum algorithm framework for MEDR to speed up the matrix exponential-based DR algorithms.

**III. GENERAL QMEDR FRAMEWORK BASED ON BLOCK ENCODING**

In this section we will detail the general quantum algorithm framework that performs matrix exponential-based DR more efficiently than what is achievable classically under certain conditions.

Assume that the original data matrix  $X = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{N-1}]^T \in \mathbb{R}^{N \times M}$  is stored in a structured QRAM [36,50,51]. Then there exists a quantum algorithm that can perform the mapping

$$\mathbf{O}_1 : |i\rangle|0\rangle \mapsto |i\rangle\|\mathbf{x}_i\| \tag{6}$$

in time  $O(\text{polylog}(NM))$  and perform the mappings with  $\epsilon_x$  precision in time  $O(\text{polylog}(\frac{NM}{\epsilon_x}))$ ,

$$\begin{aligned} \mathbf{O}_2 : |i\rangle|0\rangle &\mapsto |i\rangle\|\mathbf{x}_i\rangle = |i\rangle \frac{1}{\|\mathbf{x}_i\|} \sum_{j=0}^{M-1} x_{ij}|j\rangle, \\ \mathbf{O}_3 : |0\rangle|j\rangle &\mapsto \frac{1}{\|X\|_F} \sum_{i=0}^{N-1} \|\mathbf{x}_i\| |i\rangle|j\rangle, \end{aligned} \tag{7}$$

where  $x_{ij}$  denotes the  $(i, j)$  entry of  $X$  and  $\|\cdot\|_F$  is the Frobenius norm for a matrix. Based on this assumption, the framework can be summarized as the following theorem.

*Theorem 2 (QMEDR framework).* For an algorithm belonging to the MEDR framework, let  $\kappa_1, \kappa_2 \geq 2$  such that  $\frac{1}{\kappa_1} \leq S_1 \leq I$  and  $\frac{1}{\kappa_2} \leq S_2 \leq I$ . Suppose that  $U_1$  is an  $(\alpha, a, \epsilon)$  block encoding of  $S_1$  and  $U_2$  is a  $(\beta, b, \delta)$  block encoding of  $S_2$ , which can be implemented in times  $T_1$  and  $T_2$ , respectively. Then there exists a quantum algorithm to produce the compressed digital-encoded state

$$\frac{1}{\sqrt{Nm}} \sum_{i=0}^{N-1} \sum_{j=0}^{m-1} |i\rangle|j\rangle|y_{ij}\rangle := |\psi\rangle \tag{8}$$

in time  $\tilde{O}(\frac{T \max_i \|\mathbf{x}_i\|^2 m \sqrt{M}}{\epsilon})$ , where  $T = \tilde{O}(\alpha \kappa_1 (a + T_1) + \beta \kappa_2 (b + T_2))$ ,  $y_{ij}$  is the  $(i, j)$  entry of the low-dimensional data matrix  $Y$ , and its error is  $\epsilon$ .

Note that with  $\tilde{O}(\cdot)$  we hide the polylogarithmic factors. To prove Theorem 2, we give the framework details in Sec. III A and analyze its complexity in Sec. III B.

### A. QMEDR framework

The core of the QMEDR framework is to solve the generalized eigenproblem in Eq. (4) and then construct the compressed digital-encoded state. Since the matrices  $e^{S_1}$  and  $e^{S_2}$  are nonsingular, the generalized eigenproblem can be transformed into the standard eigenvalue problem and then solved by quantum phase estimation [39]. One could rewrite Eq. (4) as  $e^{-S_2}e^{S_1}\mathbf{v} = \lambda\mathbf{v}$  (see Appendix D); however, one runs into the immediate problem that the matrix  $e^{-S_2}e^{S_1}$  is generally not Hermitian. This will make the quantum phase estimation fail. Instead here we consider the more general case and use the fact that a positive-definite matrix  $e^{S_1}$  has a unique positive-definite square root  $e^{S_1/2}$ . Letting  $\mathbf{u} := e^{S_1/2}\mathbf{v}$ , Eq. (4) can be transformed into

$$e^{S_1/2}e^{-S_2}e^{S_1/2}\mathbf{u} = \lambda\mathbf{u}. \quad (9)$$

Obviously, it is a Hermitian eigenvalue problem. Based on this, we first use the block-encoding technique to simulate  $e^{S_1/2}e^{-S_2}e^{S_1/2}$ , which can be done by constructing its block encoding. With it, we extract the eigenvalue  $\lambda$  and the associated eigenvector  $\mathbf{u}$  by quantum phase estimation [39] and then use the quantum minimum-finding algorithm [52] to find the first  $m$  smallest eigenvalues. Next we apply  $e^{-S_1/2}$  to  $\mathbf{u}$  to get the corresponding generalized eigenvector  $\mathbf{v} = e^{-S_1/2}\mathbf{u}$ . After that, we construct the compressed digital-encoded state by quantum amplitude amplification [53] and inner product estimation [54]. The QMEDR framework consists of four steps, which we will now detail one by one.

(1) *Construct the block encoding of  $e^{S_1/2}e^{-S_2}e^{S_1/2}$ .* We first give the following lemma, which is inspired by [46] and allows us to construct the block encodings of  $e^{S_1}$  and  $e^{-S_2}$ .

*Lemma 1 (implementing the block encodings of matrix exponential and its inverse).* Let  $c \in \{1, -1\}$ ,  $\epsilon \in (0, \frac{1}{2}]$ ,  $\kappa \geq 2$ , and  $H$  be a Hermitian matrix such that  $\frac{1}{\kappa} \leq H \leq I$ . If for some fixed  $\omega \in \mathbb{R}_+$  we have  $\delta \leq \omega\epsilon/\kappa \log_2(\frac{1}{\epsilon}) \log_2^2[\kappa \log_2(\frac{1}{\epsilon})]$  and  $U$  is an  $(\alpha, a, \delta)$  block encoding of  $H$  which can be implemented in time  $T_U$ , then we can implement a unitary  $\tilde{U}$  that is an  $(e^2, a + O(\log_2[\kappa \log_2(\frac{1}{\epsilon})]), e^2\epsilon)$  block encoding of  $e^{cH}$  in cost  $O(\alpha\kappa \log_2(\frac{1}{\epsilon})(a + T_U) + \kappa \log_2(\frac{\kappa}{\epsilon}) \log_2(\frac{1}{\epsilon}))$ .

For the proof of Lemma 1, see Appendix A.

In general,  $S_1$  and  $S_2$  are semipositive-definite Hermitian. Since they may involve large numbers, the normalization is needed in most cases [44]. For simplicity, here we assume that  $\frac{1}{\kappa_1} \leq S_1 \leq I$  and  $\frac{1}{\kappa_2} \leq S_2 \leq I$ , where  $\kappa_1, \kappa_2 \geq 2$ . Suppose that  $U_1$  is an  $(\alpha, a, \epsilon)$  block encoding of  $S_1$  and  $U_2$  is a  $(\beta, b, \delta)$  block encoding of  $S_2$ . We implement an  $(e^2, a + O(\log_2[\kappa_1 \log_2(\frac{1}{\epsilon})]), e^2\epsilon_1)$  block encoding of  $e^{S_1}$  and an  $(e^2, b + O(\log_2[\kappa_2 \log_2(\frac{1}{\delta})]), e^2\epsilon_2)$  block encoding of  $e^{-S_2}$  by Lemma 1, where  $\epsilon_1, \epsilon_2 \in (0, \frac{1}{2}]$ . We can also obtain an  $(\frac{\alpha}{2}, a, \frac{\epsilon}{2})$  block encoding of  $\frac{S_1}{2}$  [55] and then construct an  $(e^2, a + O(\log_2[\kappa_1 \log_2(\frac{1}{\epsilon})]), e^2\epsilon_1)$  block encoding of  $e^{S_1/2}$ . Then an  $(e^6, 2a + b + O(\log_2[\kappa_2 \log_2(\frac{1}{\delta})]) + \log_2[\kappa_1 \log_2(\frac{1}{\epsilon})], 2e^6\epsilon_1 + e^6\epsilon_2)$  block encoding of

$e^{S_1/2}e^{-S_2}e^{S_1/2}$  is implemented by the product of block-encoded matrices [35,56]. Based on this, we can simulate  $e^{S_1/2}e^{-S_2}e^{S_1/2}$  by Theorem 1.

(2) *Extract the first  $m$  principal generalized eigenvalues.*

Here we first use quantum phase estimation [39] to reveal the eigenvalues and eigenvectors of  $e^{S_1/2}e^{-S_2}e^{S_1/2}$  and then use the quantum minimum-finding algorithm [52] to search for the first  $m$  smallest eigenvalues, i.e., the first  $m$  smallest generalized eigenvalues of Eq. (4). The details are as follows.

(2.1) *Reveal the eigenvalues and eigenvectors of  $e^{S_1/2}e^{-S_2}e^{S_1/2}$ .* Given the block encoding of  $e^{S_1/2}e^{-S_2}e^{S_1/2}$ , the unitary  $e^{i(e^{S_1/2}e^{-S_2}e^{S_1/2})^t}$  can be implemented according to Theorem 1, where  $t^2 = -1$ . By using it, we apply quantum phase estimation on the first two registers of the state

$$|0\rangle^{\otimes q_1} \frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |i\rangle|i\rangle \quad (10)$$

and then we obtain the state

$$\frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |\lambda_i\rangle|\mathbf{u}_i\rangle|\mathbf{u}_i\rangle := |\Psi\rangle, \quad (11)$$

where  $\lambda_i$  and  $|\mathbf{u}_i\rangle$  are the eigenvalue and the corresponding eigenvector of  $e^{S_1/2}e^{-S_2}e^{S_1/2}$ , respectively, and the value of  $q_1$  determines the accuracy of phase estimation, which we discuss in Sec. III B. Such an entangled state in Eq. (10) can be efficiently constructed by using Hadamard gates and controlled-NOT (CNOT) gates and its partial trace over the first two registers is proportional to the identity matrix.

(2.2) *Find the first  $m$  smallest eigenvalues.* To invoke the quantum minimum-finding algorithm, we should be able to construct an oracle  $\mathbf{O}_\chi$  to mark the item  $\lambda_i \leq \gamma$ , where  $\gamma$  is a threshold parameter which can be determined by measuring the eigenvalue register. We define a classical Boolean function  $\chi$  on the eigenvalue register satisfying

$$\chi(\lambda) = \begin{cases} 1, & \lambda \leq \gamma \\ 0, & \lambda > \gamma. \end{cases} \quad (12)$$

Based on  $\chi$ , the corresponding quantum oracle  $\mathbf{O}_\chi$  can be constructed, satisfying

$$\mathbf{O}_\chi |\lambda_i\rangle|\mathbf{v}_i\rangle|\mathbf{v}_i\rangle = (-1)^{\chi(\lambda_i)} |\lambda_i\rangle|\mathbf{v}_i\rangle|\mathbf{v}_i\rangle. \quad (13)$$

Then we apply quantum minimum-finding algorithm on  $|\Psi\rangle$  to find the minimum eigenvalue, in which the Grover iteration operator is  $(2|\Psi\rangle\langle\Psi| - I)\mathbf{O}_\chi$ .

Without loss of generality, we assume that the eigenvalues have been arranged in ascending order, that is,  $\lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{M-1}$ . Suppose that we have obtained the first  $s$  smallest eigenvalues. We modify  $\chi$  as

$$\chi(\lambda) = \begin{cases} 1, & \lambda \leq \gamma, \lambda \notin \{\lambda_j\}_{j=0}^{s-1} \\ 0, & \lambda > \gamma \end{cases} \quad (14)$$

and construct the new oracle  $\mathbf{O}_\chi$  and the new Grover operator. Based on this, we invoke the quantum minimum-finding algorithm again to find  $\lambda_s$ . The first  $m$  smallest eigenvalues  $\{\lambda_j\}_{j=0}^{m-1}$  can be obtained in this way. Note that when we get an eigenvalue  $\lambda_j$ , we also get the corresponding quantum state  $|\mathbf{u}_j\rangle|\mathbf{u}_j\rangle$ .



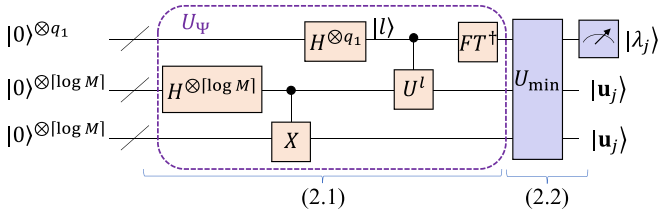


FIG. 2. Quantum circuit of step 2 in the QMEDR framework, where / denotes a bundle of wires,  $H$  is the Hadamard gate,  $X$  is the NOT gate,  $U = e^{i(e^{S_1/2} e^{-S_2} e^{S_1/2})t}$ ,  $FT$  denotes the quantum Fourier transformation, and  $U_{\min}$  denotes the quantum minimum-finding algorithm.

The entire quantum circuit of step 2 is shown in Fig. 2.

(3) *Extract the first  $m$  principal generalized eigenvectors.* With  $\{\lambda_j\}_{j=0}^{m-1}$  we obtained in the preceding step, here we first use the quantum amplitude amplification [53] to get the first  $m$  principal eigenvectors of  $e^{S_1/2} e^{-S_2} e^{S_1/2}$  and then use the matrix computation technique (similar to the Harrow-Hassidim-Lloyd algorithm [3]) to get the corresponding generalized eigenvectors.

(3.1) *Prepare the state  $\frac{1}{\sqrt{m}} \sum_{i=0}^{m-1} |\lambda_i\rangle |\mathbf{u}_i\rangle |\mathbf{u}_i\rangle$ .* We prepare the state  $|\Psi\rangle$  as the initial state. Then an oracle  $\mathcal{O}_\lambda$  is defined to mark the items  $\{\lambda_j\}_{j=0}^{m-1}$  (similar to  $\mathbf{O}_\chi$ ), i.e.,  $\mathcal{O}_\lambda |\lambda_j\rangle |\mathbf{u}_j\rangle |\mathbf{u}_j\rangle = -|\lambda_j\rangle |\mathbf{u}_j\rangle |\mathbf{u}_j\rangle$  for  $j = 0, 1, \dots, m-1$ . With  $\mathcal{O}_\lambda$ , we use quantum amplitude amplification [53] to get

$$\frac{1}{\sqrt{m}} \sum_{i=0}^{m-1} |\lambda_i\rangle |\mathbf{u}_i\rangle |\mathbf{u}_i\rangle, \quad (15)$$

where the Grover operator  $G = (2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_\lambda$ .

(3.2) *Obtain the first  $m$  principal generalized eigenvectors.* Suppose that the Hermitian matrix  $e^{-S_1}$  has a spectral decomposition form  $\sum_{j=0}^{M-1} \sigma_j |\mathbf{w}_j\rangle\langle\mathbf{w}_j|$ . Obviously,  $\sigma_j \in [e^{-1}, e^{-1/\kappa_1}] \subset [e^{-1}, 1)$ . We decompose  $|\mathbf{u}_i\rangle$  of the second register in the eigenbasis of  $e^{-S_1}$  and then Eq. (15) can be rewritten as

$$\frac{1}{\sqrt{m}} \sum_{i=0}^{m-1} |\lambda_i\rangle \left( \sum_{j=0}^{M-1} \gamma_{ij} |\mathbf{w}_j\rangle \right) |\mathbf{u}_i\rangle, \quad (16)$$

where  $\gamma_{ij} = \langle \mathbf{w}_j | \mathbf{u}_i \rangle$ .

To obtain the corresponding generalized eigenvector  $|\mathbf{v}_i\rangle$ , we introduce two auxiliary registers to produce

$$\frac{1}{\sqrt{m}} \sum_{i=0}^{m-1} |\lambda_i\rangle \left( \sum_{j=0}^{M-1} \gamma_{ij} |\mathbf{w}_j\rangle \right) |\mathbf{u}_i\rangle |0\rangle^{\otimes q_1} |0\rangle. \quad (17)$$

By Lemma 1 we can implement an  $(e^2, a + O(\log_2[\kappa_1 \log_2(\frac{1}{\epsilon_1})]), e^2 \epsilon_1)$  block encoding of  $e^{-S_1}$ . With it and Theorem 1 we perform quantum phase estimation on the second and fourth registers to get

$$\frac{1}{\sqrt{m}} \sum_{i=0}^{m-1} |\lambda_i\rangle \left( \sum_{j=0}^{M-1} \gamma_{ij} |\mathbf{w}_j\rangle \right) |\mathbf{u}_i\rangle |\sigma_j\rangle |0\rangle. \quad (18)$$

Then we rotate the last qubit, conditioned on  $|\sigma_j\rangle$ , to yield

$$\frac{1}{\sqrt{m}} \sum_{i=0}^{m-1} |\lambda_i\rangle \left( \sum_{j=0}^{M-1} \gamma_{ij} |\mathbf{w}_j\rangle \right) |\mathbf{u}_i\rangle |\sigma_j\rangle [\sqrt{1 - \sigma_j} |0\rangle + (\sigma_j)^{1/2} |1\rangle]. \quad (19)$$

Next we amplify the amplitude of  $|1\rangle$  and discard the last two registers to get

$$\frac{\sum_{i=0}^{m-1} \sum_{j=0}^{M-1} (\sigma_j)^{1/2} \gamma_{ij} |\lambda_i\rangle |\mathbf{w}_j\rangle |\mathbf{u}_i\rangle}{\sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{M-1} \sigma_j \gamma_{ij}^2}}, \quad (20)$$

which can be rewritten as

$$\begin{aligned} & \sum_{i=0}^{m-1} \frac{\sqrt{\sum_{j=0}^{M-1} \sigma_j \gamma_{ij}^2}}{\sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{M-1} \sigma_j \gamma_{ij}^2}} |\lambda_i\rangle \left( \frac{\sum_{j=0}^{M-1} (\sigma_j)^{1/2} \gamma_{ij} |\mathbf{w}_j\rangle}{\sqrt{\sum_{j=0}^{M-1} \sigma_j \gamma_{ij}^2}} \right) |\mathbf{u}_i\rangle \\ &= \sum_{i=0}^{m-1} \frac{\sqrt{\sum_{j=0}^{M-1} \sigma_j \gamma_{ij}^2}}{\sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{M-1} \sigma_j \gamma_{ij}^2}} |\lambda_i\rangle |\mathbf{v}_i\rangle |\mathbf{u}_i\rangle, \end{aligned}$$

where  $\lambda_i$  and  $|\mathbf{v}_i\rangle$  are the generalized eigenvalue and the corresponding generalized eigenvector of Eq. (4). After that, we do the same for  $|\mathbf{u}_i\rangle$  of the third register. Finally, we get the state

$$\sum_{i=0}^{m-1} \zeta_i |\lambda_i\rangle |\mathbf{v}_i\rangle |\mathbf{v}_i\rangle := |\varphi\rangle, \quad (21)$$

where  $\zeta_i = \frac{\sum_{j=0}^{M-1} \sigma_j \gamma_{ij}^2}{\sqrt{\sum_{i=0}^{m-1} (\sum_{j=0}^{M-1} \sigma_j \gamma_{ij}^2)^2}}$  is the normalization coefficient and  $\frac{1}{e\sqrt{m}} < \zeta_i < \frac{e}{\sqrt{m}}$  (see Appendix B for the proof).

The entire quantum circuit of step 3 is shown in Fig. 3.

(4) *Construct the compressed digital-encoded state.* Since  $y_{ij} = \|\mathbf{x}_i\| \langle \mathbf{x}_i | \mathbf{v}_j \rangle$ , we can obtain the value of  $y_{ij}$  by computing the inner product  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle$ ,  $i = 0, 1, \dots, N-1$ ,  $j = 0, 1, \dots, m-1$ . Without loss of generality, we assume  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle \geq 0$  because both  $|\mathbf{v}_j\rangle$  and  $-|\mathbf{v}_j\rangle$  are generalized eigenvectors of Eq. (4) corresponding to the generalized eigenvalue  $\lambda_j$ . Then we can get the value of  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle$  by computing  $(\langle \mathbf{x}_i | \mathbf{v}_j \rangle)^2$ . An intuitive idea is to use the Hadamard test [57]. (See Appendix C for the reason why we do not compute  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle$  directly.) However, it would be exhausting to do so over a large data set. Instead, here we use the inner product estimation to accomplish this task in parallel. The following lemmas are required for step 4.

*Lemma 2 (distance or inner product estimation [54]).* Assume that the unitaries  $|i\rangle|0\rangle \mapsto |i\rangle|\mathbf{x}_i\rangle$  and  $|j\rangle|0\rangle \mapsto |j\rangle|\mathbf{x}_j\rangle$  can be performed in time  $T$  and the norms of the vectors are known. For any  $\Delta > 0$  and  $\epsilon > 0$  there exists a quantum algorithm that can compute

$$|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle \|\mathbf{x}_i - \mathbf{x}_j\|^2 \quad (22)$$

or

$$|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle \|\mathbf{x}_i - \mathbf{x}_j\|^2 \quad (23)$$

with a probability of at least  $1 - 2\Delta$  for any  $\epsilon$  with complexity  $\tilde{O}(\frac{\|\mathbf{x}_i\| \|\mathbf{x}_j\| T \log_2(1/\Delta)}{\epsilon})$ , where  $\epsilon$  is the error of  $\mathbf{x}_i^T \mathbf{x}_j$  or  $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ .

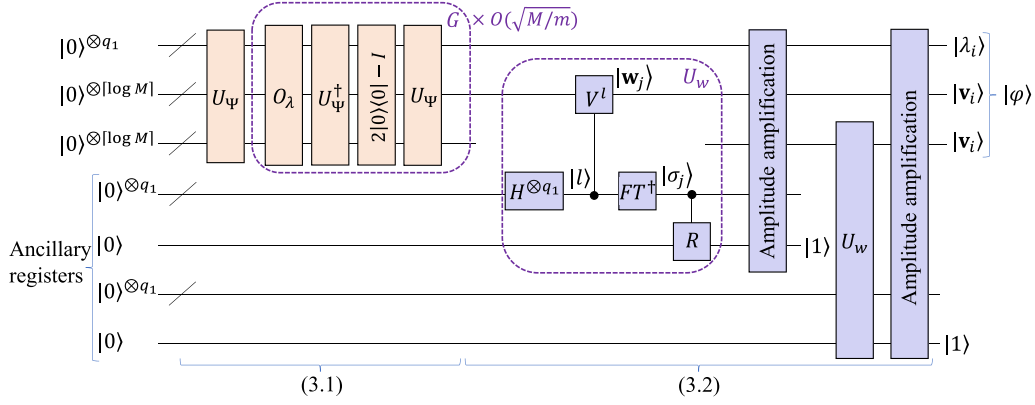


FIG. 3. Quantum circuit of step 3 in the QMEDR framework, where  $U_{\Psi}$  is the unitary operation for preparing state  $|\Psi\rangle$  (see Fig. 2),  $V = e^{i(c^{-S_1})^t}$ , and  $R$  is the controlled unitary operation corresponding to controlled rotation.

**Lemma 3 (quantum multiplier [14,58,59]).** Let integers  $a$  and  $b$  be an  $n$ -bit string. Then there is a quantum algorithm with  $O(\text{poly}(n))$  single- and two-qubit gates that can realize

$$|a\rangle|b\rangle \mapsto |a\rangle|ab\rangle. \quad (24)$$

Note that for accuracy defined as  $\varepsilon = 2^{-n}$ , the complexity of the quantum multiplier (QM) is given by  $O(\text{polylog}(\frac{1}{\varepsilon}))$ .

We now elaborate on step 4 as follows.

(4.1) With these two unitaries  $|i\rangle \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |j\rangle|0\rangle \mapsto |i\rangle \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |j\rangle|\phi_{ij}\rangle$  ( $|\phi_{ij}\rangle = |\lambda_j\rangle|\mathbf{x}_i\rangle|\mathbf{x}_i\rangle$ ) and  $|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle|\varphi\rangle$ , we use Hadamard gates and Lemma 2 to produce the state

$$\frac{1}{\sqrt{Nm}} \sum_{i=0}^{N-1} \sum_{j=0}^{m-1} |i\rangle|j\rangle \left| \frac{\zeta_j(\langle \mathbf{x}_i | \mathbf{v}_j \rangle)^2}{\sqrt{m}} \right\rangle |0\rangle^{\otimes q_2} |0\rangle^{\otimes q_3} |0\rangle^{\otimes q_3}, \quad (25)$$

where  $q_2$  is the largest number of qubits necessary to store  $\frac{1}{\zeta_i}$  and  $q_3$  is the largest number of qubits necessary to store  $y_{ij}$  and  $\|\mathbf{x}_i\|$ .

For  $|\phi_{ij}\rangle$ , the mapping

$$|i\rangle \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |j\rangle|0\rangle \mapsto |i\rangle \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |j\rangle|\phi_{ij}\rangle \quad (26)$$

is performed by  $\mathbf{O}_2$  and a sequence of controlled unitary operations  $C(j) : |j\rangle|0\rangle \mapsto |j\rangle|\lambda_j\rangle$  [24],  $j = 0, 1, \dots, m-1$ . Each  $C(j)$  can be performed efficiently because we have obtained  $\{\lambda_j\}_{j=0}^{m-1}$  in step 2. The quantum circuit for preparing the state  $|\phi_{ij}\rangle$  is shown in Fig. 4.

(4.2) In this stage, we need the values of  $\{\frac{1}{\zeta_i}\}_{i=0}^{m-1}$  which can be obtained by measuring the first registers of  $|\varphi\rangle$  and counting the probability distribution of the generalized eigenvalues. Then we store them in a structured QRAM [50,51], which allows us to perform the mapping

$$\mathbf{O}_4 : |i\rangle|0\rangle \mapsto |i\rangle \left| \frac{1}{\zeta_i} \right\rangle \quad (27)$$

in time  $O(\log_2 m)$ .

Based on this, we perform  $\mathbf{O}_4$  on the second and fourth registers of Eq. (25) to get

$$\frac{1}{\sqrt{Nm}} \sum_{i=0}^{N-1} \sum_{j=0}^{m-1} |i\rangle|j\rangle \left| \frac{\zeta_j(\langle \mathbf{x}_i | \mathbf{v}_j \rangle)^2}{\sqrt{m}} \right\rangle \left| \frac{1}{\zeta_j} \right\rangle |0\rangle|0\rangle \quad (28)$$

and then perform QM on the third and fourth registers to get

$$\frac{1}{\sqrt{Nm}} \sum_{i=0}^{N-1} \sum_{j=0}^{m-1} |i\rangle|j\rangle \left| \frac{\langle \mathbf{x}_i | \mathbf{v}_j \rangle^2}{\sqrt{m}} \right\rangle \left| \frac{1}{\zeta_j} \right\rangle |0\rangle|0\rangle. \quad (29)$$

(4.3) We perform  $\mathbf{O}_1$  on the first and fifth registers to obtain

$$\frac{1}{\sqrt{Nm}} \sum_{i=0}^{N-1} \sum_{j=0}^{m-1} |i\rangle|j\rangle \left| \frac{\langle \mathbf{x}_i | \mathbf{v}_j \rangle^2}{\sqrt{m}} \right\rangle \left| \frac{1}{\zeta_j} \right\rangle \|\mathbf{x}_i\| |0\rangle \quad (30)$$

and then perform QM twice on the third and fifth registers to get

$$\frac{1}{\sqrt{Nm}} \sum_{i=0}^{N-1} \sum_{j=0}^{m-1} |i\rangle|j\rangle \left| \frac{\langle \mathbf{x}_i | \mathbf{v}_j \rangle^2 \|\mathbf{x}_i\|^2}{\sqrt{m}} \right\rangle \left| \frac{1}{\zeta_j} \right\rangle \|\mathbf{x}_i\| |0\rangle, \quad (31)$$

that is,

$$\frac{1}{\sqrt{Nm}} \sum_{i=0}^{N-1} \sum_{j=0}^{m-1} |i\rangle|j\rangle \left| \frac{y_{ij}^2}{\sqrt{m}} \right\rangle \left| \frac{1}{\zeta_j} \right\rangle \|\mathbf{x}_i\| |0\rangle. \quad (32)$$

The compressed digital-encoded state  $|\psi\rangle$  can be obtained by uncomputing the third, fourth, and fifth registers after

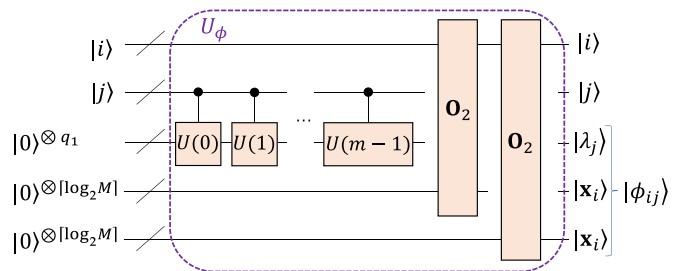


FIG. 4. Quantum circuit for preparing state  $|\phi_{ij}\rangle$  where controlled  $U(j)$  corresponds to  $C(j)$  for  $j = 0, 1, \dots, m-1$ .

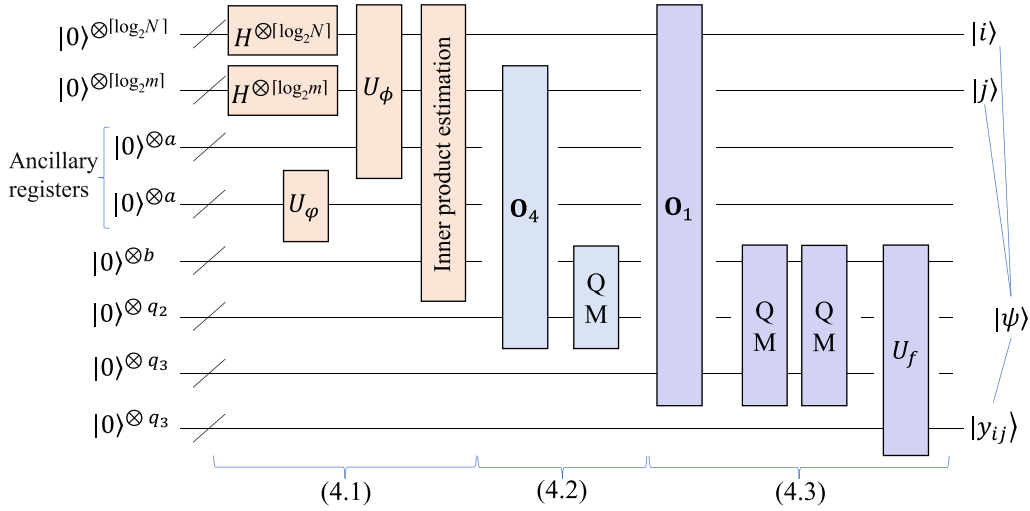


FIG. 5. Quantum circuit of step 4 in the QMEDR framework. For simplicity,  $a = q_1 + 2\lceil \log_2 M \rceil$ ,  $b$  denotes the number of qubits required for the inner product estimation, and  $U_\varphi$  and  $U_\phi$  are the unitary operations for preparing states  $|\varphi\rangle$  and  $|\phi_{ij}\rangle$ , respectively (here we omit the ancillary registers; see Figs. 3 and 4 for more details). Note that the ancillary registers are required for inner product estimation and we omit them from Eq. (25) for convenience.

applying  $U_f : |x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$  on the third and sixth registers, where  $f(x) = \sqrt{\sqrt{m}x}$  and  $x = \frac{y_{ij}^2}{\sqrt{m}}$ .

The entire quantum circuit of step 4 is shown in Fig. 5.

**B. Complexity analysis**

Now we analyze the complexity of each step and discuss the overall complexity.

In step 1,  $U_1$  is an  $(\alpha, a, \varepsilon)$  block encoding of  $S_1$  and  $U_2$  is a  $(\beta, b, \delta)$  block encoding of  $S_2$ , which can be implemented in times  $T_1$  and  $T_2$ , respectively. Then, by Lemma 1, the time complexity of implementing the block encodings of  $e^{S_1}$  and  $e^{-S_2}$  are  $\tilde{O}(\alpha\kappa_1(a + T_1))$  and  $\tilde{O}(\beta\kappa_2(b + T_2))$ , respectively. Then we can implement a block encoding of  $e^{S_1/2}e^{-S_2}e^{S_1/2}$  in time  $\tilde{O}(\alpha\kappa_1(a + T_1) + \beta\kappa_2(b + T_2)) := T$  by Lemma B.4 in [56].

In step 2, for stage (2.1),  $O(\lceil \log_2 M \rceil)$  Hadamard gates and CNOT gates are needed to prepare the state in Eq. (10). Next, with the block encoding of  $e^{S_1/2}e^{-S_2}e^{S_1/2}$ , the unitary  $e^{i(e^{S_1/2}e^{-S_2}e^{S_1/2})t}$  can be implemented in time  $\tilde{O}(Tt)$  by Theorem 1. Hence, the quantum phase estimation has a query complexity of  $O(\frac{1}{\varepsilon_1}(2 + \frac{1}{2\eta}))$  and each query has a time complexity  $\tilde{O}(T)$ , where  $\varepsilon_1$  is the error of quantum phase estimation and  $1 - \eta$  is the probability to succeed. Suppose we wish to approximate  $\lambda_j$  to an accuracy  $2^{-n}$  with a probability of success of at least  $1 - \eta$ ; we should choose  $q_1 = n + \lceil \log_2(2 + \frac{1}{\eta}) \rceil$  [39]. For stage (2.2) we should invoke the quantum minimum-finding algorithm  $m$  times to get the first  $m$  smallest eigenvalues, and each takes  $O(\sqrt{M})$  query complexity.

In step 3, for stage (3.1), with  $|\Psi\rangle$ ,  $O(\sqrt{\frac{M}{m}})$  Grover operator iterations is enough to obtain the state in Eq. (15). For stage (3.2), which is similar to (2.1), with the block encoding of  $e^{-S_1}$ , the unitary  $e^{i(e^{-S_1})t}$  can be implemented in time  $\tilde{O}(\alpha\kappa_1(a + T_1)t)$  by Theorem 1. Hence, the time complexity of quantum phase estimation is  $O(\frac{\alpha\kappa_1(a+T_1)}{\varepsilon_1})$ . The time

complexity of controlled rotations and quantum amplitude amplifications can be neglected compared with other subroutines.

In step 4, for stage (4.1), when preparing the state  $|\phi_{ij}\rangle$ ,  $C(j)$  ( $j = 0, 1, \dots, m - 1$ ) and two  $\mathbf{O}_2$  are needed to perform the mapping in Eq. (26). Each  $C(j)$  takes  $O(\log_2(\frac{1}{\varepsilon_1})\log_2 m)$  elementary gates [24,39], so  $O(m\log_2(\frac{1}{\varepsilon_1})\log_2 m)$  time is needed to perform all  $C(j)$ . Each  $\mathbf{O}_2$  can be done with  $\varepsilon_x$  precision in time  $O(\text{polylog}(\frac{NM}{\varepsilon_x}))$ . Hence, the time complexity for preparing the state  $|\phi_{ij}\rangle$  is  $\tilde{O}(m)$ . Moreover, the time complexity for preparing the state  $|\varphi\rangle$  is  $\tilde{O}(\frac{T}{\varepsilon_1}\sqrt{\frac{M}{m}})$ . Based on the above, the time complexity of producing the state in Eq. (25) is  $\tilde{O}(\frac{T}{\varepsilon_1\varepsilon_2}\sqrt{\frac{M}{m}})$ , where  $\varepsilon_2$  is the error of the inner product estimation. For stage (4.2),  $O(\frac{1}{\min_i \zeta_i} \log_2(\frac{1}{\min_i \zeta_i}) \log_2(\frac{1}{\varepsilon_\zeta}))$  measurements is enough to get all the values of  $\{\zeta_i\}_{i=0}^{m-1}$  with accuracy  $\varepsilon_\zeta$ , where  $\min_i \zeta_i = \frac{1}{e\sqrt{m}}$ . Then the time and space complexity to store  $\{\frac{1}{\zeta_i}\}_{i=0}^{m-1}$  in a structured QRAM are  $O(m\log_2^2 m)$ . Note that here the error of  $\frac{1}{\zeta_i}$  will become  $O(m\varepsilon_\zeta)$ . To ensure the error of  $\frac{(\langle \mathbf{x}_i | \mathbf{v}_j \rangle)^2}{\sqrt{m}}$  is  $\varepsilon_2$ , we should control  $\varepsilon_\zeta = O(\frac{\varepsilon_2}{m})$ . After that, we use  $\mathbf{O}_4$  to get the state in Eq. (28) in time  $O(\log_2 m)$ . The time complexity of QM is  $O(\text{polylog}(\frac{1}{\varepsilon_2}))$  with accuracy  $\varepsilon_2$ , which can be neglected compared with other subroutines. For stage (4.3), we use  $\mathbf{O}_1$  to get the state in Eq. (30) in time  $O(\text{polylog}(NM))$ . Moreover, we omit the complexity of QM and  $U_f$ .

The complexity of each step of the QMEDR framework is summarized as Table I. Furthermore, if every  $\lambda_j = O(\frac{1}{m})$ ,  $\varepsilon_1$  should take  $O(\frac{1}{m})$  and thus the time complexities of steps 2 and 3 are  $\tilde{O}(Tm^2\sqrt{M})$  and  $\tilde{O}(T\sqrt{mM})$ , respectively. In step 4, the error of  $\frac{(\langle \mathbf{x}_i | \mathbf{v}_j \rangle)^2}{\sqrt{m}}$  is  $\varepsilon_2$ , which will make the error of  $y_{ij}^2 = (\langle \mathbf{x}_i | \mathbf{v}_j \rangle)^2 \|\mathbf{x}_i\|^2$  equal to  $\sqrt{m}\|\mathbf{x}_i\|^2\varepsilon_2$ . Suppose  $y_{ij} = O(1)$ . To ensure the final error of  $y_{ij}$  is within  $\epsilon$ , we should take

TABLE I. Time complexity of each step of the QMEDR framework. Here  $\varepsilon_1$  is the error of quantum phase estimation and  $\varepsilon_2$  is the error of inner product estimation.

Step	Time complexity
1	$T = \tilde{O}(\alpha\kappa_1(a + T_1) + \beta\kappa_2(b + T_2))$
2	$\tilde{O}\left(\frac{Tm\sqrt{M}}{\varepsilon_1}\right)$
3	$\tilde{O}\left(\frac{T}{\varepsilon_1}\sqrt{\frac{M}{m}}\right)$
4	$\tilde{O}\left(\frac{T}{\varepsilon_1\varepsilon_2}\sqrt{\frac{M}{m}} + \frac{T\sqrt{mM}}{\varepsilon_1}\right)$

$\varepsilon_2 = O\left(\frac{\varepsilon}{\sqrt{m \max_i \|\mathbf{x}_i\|^2}}\right)$ . Therefore, the total time complexity of step 4 is  $\tilde{O}\left(\frac{T \max_i \|\mathbf{x}_i\|^2 m \sqrt{M}}{\varepsilon}\right)$ .

In conclusion, the QMEDR framework can output the compressed digital-encoded state  $|\psi\rangle$  in time  $\tilde{O}\left(\frac{T \max_i \|\mathbf{x}_i\|^2 m \sqrt{M}}{\varepsilon}\right)$ , where  $T = \tilde{O}(\alpha\kappa_1(a + T_1) + \beta\kappa_2(b + T_2))$  and  $\varepsilon$  is the error of  $y_{ij}$ .

#### IV. APPLICATIONS

In this section we use the QMEDR framework to accelerate the classical ELPP [42], EUDP [19,23], ENPE [43], and EDA [44] algorithms, which all belong to the MEDR framework. The core is to implement the block encodings of  $S_1$  and  $S_2$  of these algorithms. Once the two block encodings are implemented efficiently, we can design the corresponding quantum algorithms by Theorem 2.

##### A. Quantum ELPP algorithm

In ELPP,  $S_1 = X^T L X$  and  $S_2 = X^T D X$ , where  $L = D - S$ ,  $D$  is a diagonal matrix,  $D_{ii} = \sum_{j \neq i} S_{ij}$ , and the similarity matrix  $S$  is defined as

$$S_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} & \text{for } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_k(\mathbf{x}_i) \\ 0 & \text{otherwise,} \end{cases} \quad (33)$$

where  $\sigma$  is a parameter that is determined empirically and  $N_k(\mathbf{x}_j)$  denotes the set of the  $k$  nearest neighbors of  $\mathbf{x}_j$ . To construct the block encodings of  $S_1$  and  $S_2$ , we should be able to compute the matrices  $S$ ,  $D$ , and  $L$ . We first use the quantum  $k$  nearest-neighbor algorithm, a straightforward generalization of quantum nearest-neighbor classification in [60], to obtain the  $k$  nearest neighbors of each sample, which takes  $\tilde{O}(kN\sqrt{N})$  time. Then the matrices  $S$ ,  $D$ , and  $L$  can be computed in classical in time  $O(kN)$ .

For  $S_1$ , we can implement the block encodings of  $X$  and  $L$  and then implement its block encoding by the product of block-encoded matrices [35]. Since  $X$  is stored in a structured QRAM [50,51], an  $(\|X\|_F, \lceil \log_2(N+M) \rceil, \varepsilon_x)$  block encoding of  $X$  can be implemented in time  $O(\text{polylog}(\frac{NM}{\varepsilon_x}))$  by Lemma 6 in [36]. Since  $L$  is a  $(k+1)$ -sparse matrix, we can implement its  $(k+1, \text{polylog}(\frac{N}{\varepsilon_l}), \varepsilon_l)$  block encoding with  $O(1)$  queries for the sparse-access oracles and  $O(\text{polylog}(\frac{N}{\varepsilon_l}))$  elementary gates by Lemma 7 in [36]. Then we implement a  $((k+1)\|X\|_F^2, 2\lceil \log_2(N+M) \rceil + \text{polylog}(\frac{N}{\varepsilon_l}), \|X\|_F^2 \varepsilon_l + 2(k+1)\|X\|_F \varepsilon_x)$  block encoding of  $S_1$  with complexity  $O(\text{polylog}(\frac{NM}{\varepsilon}))$ , where  $\varepsilon = \min\{\varepsilon_l, \varepsilon_x\}$  [35].

For the semipositive-definite matrix  $S_2 = \sum_i D_{ii} \mathbf{x}_i \mathbf{x}_i^T$ , we can construct its block encoding by preparing the purified density operator  $\frac{S_2}{\text{tr}(S_2)} := \rho$  [35]. We first store the vector  $\mathbf{d} = [d_0, d_1, \dots, d_{N-1}]^T$ ,  $d_i = \sqrt{D_{ii}}$ , in a structured QRAM [50,51]. Note that the time and space complexity of storing  $\mathbf{d}$  are  $O(N \log_2^2 N)$  and  $O(kN \log_2^2(kN))$ , respectively. Then there exists a quantum algorithm that can perform the mapping  $U_d : |i\rangle|0\rangle \mapsto |i\rangle|d_i\rangle$  in time  $O(\log_2 N)$ . With  $U_d$ ,  $\mathbf{O}_2$ , and  $\mathbf{O}_3$ , we prepare the state  $\frac{\sum_i d_i \|\mathbf{x}_i\|}{\sqrt{\sum_i D_{ii} \|\mathbf{x}_i\|^2}} |i\rangle |\mathbf{x}_i\rangle$  whose partial trace over the first register is  $\rho$ . Then a  $(1, O(\log_2 N), \varepsilon_2)$  block encoding of  $\rho$  can be constructed by Lemma 25 in [35], where  $\varepsilon_2$  comes from the unitary operation for preparing the above state. Then we implement a  $(\text{tr}(S_2), O(\log_2 N), \varepsilon_2)$  block encoding of  $S_2$  in time  $O(\text{polylog}(\frac{NM}{\varepsilon_x}))$  [55], where the value of  $\text{tr}(S_2)$  can be computed in time  $O(N)$  and  $\text{tr}(S_2) = O(\|X\|_F^2)$ . The quantum ELLP algorithm can then be achieved by Theorem 2.

##### B. Quantum EUDP algorithm

In EUDP,  $S_1 = X^T L X$  and  $S_2 = X^T L' X$ , where  $L' = D' - S'$ , with  $D'$  a diagonal matrix,  $D'_{ii} = \sum_j S'_{ij}$ , and  $S'_{ij} = 1 - S_{ij}$ . The block encoding of  $S_1$  is the same as ELPP's. For  $S_2$ , we first compute the matrix  $L'$  in time  $O(N^2)$  and store it in a structured QRAM [50,51]. The space and time complexities to construct the data structure are  $O(N^2 \log_2^2 N)$ . Then we implement an  $(\|L'\|_F, \lceil \log_2(2N) \rceil, \varepsilon_{L'})$  block encoding of  $L'$  and further implement an  $(\|X\|_F^2 \|L'\|_F, 2\lceil \log_2(N+M) \rceil + \lceil \log_2(2N) \rceil, \|X\|_F^2 \varepsilon_{L'} + 2\|X\|_F \|L'\|_F \varepsilon_x)$  block encoding of  $S_2$  in time  $O(\text{polylog}(\frac{NM}{\varepsilon_x}))$  by the product of block-encoded matrices [35]. Then we invoke Theorem 2 to obtain the quantum ELLP algorithm.

##### C. Quantum ENPE algorithm

In ENPE,  $S_1 = X^T W X$  and  $S_2 = X^T X$ , where  $W = \arg \min \sum_i \|\mathbf{x}_i - \sum_{\mathbf{x}_j \in Q(\mathbf{x}_i)} W_{ij} \mathbf{x}_j\|$  and  $\sum_{\mathbf{x}_j \in Q(\mathbf{x}_i)} W_{ij} = 1$ . Here we use the  $\varepsilon$ -neighborhood criterion to get the nearest-neighbor set  $Q(\mathbf{x}_i)$  of  $\mathbf{x}_i$ , in which each set has  $\Theta(k)$  samples. For  $S_1$ , we first use the method of the quantum NPE algorithm [11] to get the classical information of the matrix  $W$ , which can be done in time  $\tilde{O}(N)$ . Since  $W$  is a matrix of  $\Theta(k)$  nonzero elements in each row and column, we assume that it is a  $k$ -sparse matrix. We can implement a  $(k, \text{polylog}(\frac{N}{\varepsilon_w}), \varepsilon_w)$  block encoding of  $W$  by Lemma 7 in [36] and further implement an  $(\|X\|_F^2 k, 2\lceil \log_2(N+M) \rceil + \text{polylog}(\frac{N}{\varepsilon_w}), \|X\|_F^2 \varepsilon_w + 2\|X\|_F k \varepsilon_x)$  block encoding of  $S_1$  with complexity  $O(\text{polylog}(\frac{NM}{\varepsilon}))$ , where  $\varepsilon = \min\{\varepsilon_w, \varepsilon_x\}$  [35]. For  $S_2$ , we can implement an  $(\|X\|_F^2, 2\lceil \log_2(N+M) \rceil, 2\|X\|_F \varepsilon_x)$  block encoding of it in time  $O(\text{polylog}(\frac{NM}{\varepsilon_x}))$  [35]. The quantum ENPE algorithm is obtained by Theorem 2.

##### D. Quantum EDA algorithm

In EDA,  $S_1$  and  $S_2$  are the between-class scatter matrix and the within-class scatter matrix, respectively [44]. According to the quantum LDA algorithm [9], we can first construct the density operators corresponding to  $S_1$  and  $S_2$  in time  $O(\log_2(NM))$ . Then a  $(1, O(\log_2 M), \varepsilon_1)$  block encoding of



TABLE II. Complexity comparisons between the classical ELPP, EUDP, ENPE, and EDA algorithms and their quantum versions.

Algorithm	$S_1$	$S_2$	Classical complexity	Quantum complexity <sup>a</sup>
ELPP <sup>b</sup>	$X^T L X$	$X^T D X$	$O(MN^2 + M^3)$	$\tilde{O}(N^{3/2} + T\eta M^{1/2}), T = \tilde{O}(\ X\ _F^2 \kappa_1 + \ X\ _F^2 \kappa_2)$
EUDP <sup>c</sup>	$X^T L X$	$X^T L' X$	$O(MN^2 + M^3)$	$\tilde{O}(N^2 + T\eta M^{1/2}), T = \tilde{O}(\ X\ _F^2 \kappa_1 + \ X\ _F^2 \ L'\ _F \kappa_2)$
ENPE <sup>d</sup>	$X^T W X$	$X^T X$	$O(k^3 N M + M^3), k \ll N^e$	$\tilde{O}(kN + T\eta M^{1/2}), T = \tilde{O}(\ X\ _F^2 \kappa_1 + \ X\ _F^2 \kappa_2)$
EDA <sup>f</sup>	$S_b$	$S_w$	$O(MN^2 + N^3)$	$\tilde{O}(\kappa_1 + \kappa_2 \eta M^{1/2})$

<sup>a</sup>For convenience, we let  $1/\epsilon, m = O(\text{polylog}(NM))$  in Theorem 2 and delete the factor  $k$  in ELPP and EUDP. In addition,  $\eta = \max_i \|\mathbf{x}_i\|^2$  and  $\kappa_1$  and  $\kappa_2$  correspond to  $S_1$  and  $S_2$ , respectively, in different algorithms.

<sup>b</sup>From Ref. [42].

<sup>c</sup>From Refs. [19,23].

<sup>d</sup>From Ref. [43].

<sup>e</sup>From Ref. [11].

<sup>f</sup>From Ref. [44]. In EDA,  $S_b$  and  $S_w$  are the between-class scatter matrix and the within-class scatter matrix, respectively.

$S_1$  and a  $(1, O(\log_2 M), \epsilon_2)$  block encoding of  $S_2$  are implemented in time  $O(\log_2(NM))$ , where  $\epsilon_1$  and  $\epsilon_2$  come from the unitary operations for preparing these two density operators corresponding to  $S_1$  and  $S_2$ . The quantum EDA algorithm can then be achieved by Theorem 2. Note that in quantum EDA, the first  $m$  principal eigenvectors are the eigenvectors corresponding to the first  $m$  largest generalized eigenvalues. We can replace the quantum minimum-finding algorithm in step 2 with the quantum maximum-finding algorithm [61].

The complexity comparison between the classical ELPP, EUDP, ENPE, and EDA algorithms and their quantum versions is summarized in Table II. When  $\kappa_1, \kappa_2, \|\mathbf{x}_i\| = O(\text{polylog}(NM))$ , the results show that the quantum ELPP and quantum NPE algorithms achieve polynomial speedups in both  $N$  and  $M$ , the quantum EUDP algorithm achieves a polynomial speedup in  $M$ , and the quantum EDA algorithm provides an exponential speedup on  $N$  and a polynomial speedup on  $M$  over their classical counterparts.

## V. DISCUSSION

One core of the QMEDR framework is to construct the block encoding of the matrix exponential. It provides a method for simulating the matrix exponential or the product of matrix exponentials. With it, we can easily use quantum phase estimation to reveal the eigenvectors and eigenvalues of  $e^{S_1/2} e^{-S_2} e^{S_1/2}$ . The block-encoding framework we used here is a useful tool, which can be applied to algorithms for various problems, such as Hamiltonian simulation and density matrix preparation. By using it, one can significantly improve the existing quantum DR algorithms, such as quantum LPP [30] and quantum LDA [9], and further reduce the dependence of their complexity on error. Moreover, it is useful for constructing the density matrix corresponding to the matrix chain product in the form  $(A_l \cdots A_2 A_1)(A_l \cdots A_2 A_1)^\dagger$ , which can be seen as a special simplified version of the Hermitian chain product in [9], but here the matrix  $A_i, i = 1, 2, \dots, l$ , is not limited to a Hermitian matrix. For the general Hermitian chain product in the form  $[f_1(A_1) \cdots f_2(A_2) f_1(A_1)][f_1(A_1) \cdots f_2(A_2) f_1(A_1)]^\dagger$ , the role of block encoding remains to be explored.

The other core of the QMEDR framework is to construct the compressed digital-encoded state which can be utilized as input for QML tasks to overcome the curse of dimensionality. For example, the quantum  $k$ -medoids algorithm [41] has a

time complexity  $\tilde{O}(N^{1/2} M^2)$  for one iteration and is therefore not suitable for dealing with high-dimensional data. One can select a suitable quantum DR algorithm as a preprocessing subroutine to produce the compressed digital-encoded state and input it to the quantum  $k$ -medoids algorithm. This leads to a better dependence on  $M$  in the algorithm complexity. While the digital-encoded state can be used as input for QML, the analog encoding is sometimes required, such as a quantum support vector machine [62] and quantum  $k$ -means clustering [54]. One can also construct the compressed analog-encoded state  $\frac{1}{\|Y\|_F} \sum_{i=0}^{N-1} \sum_{j=0}^{m-1} y_{ij} |i\rangle |j\rangle$  as in [24,32]; however, in this case, extra work may be needed, which is worth further exploration.

Now we divide the quantum linear DR algorithms into three types: (I) those that output the quantum states corresponding to the column vectors of the transformation matrix, (II) those that output the compressed analog-encoded state, and (III) those that output the digital-encoded state. The type I algorithms have not provided the desired quantum data compression, namely, obtaining its corresponding low-dimensional data set. The algorithms of types II and III are well adapted directly to other QML algorithms. The comparison between the QMEDR framework and the existing quantum linear DR algorithms in an end-to-end setting is summarized in Table III.

Our QMEDR framework is related to solving the generalized eigenproblem  $A\mathbf{v} = \lambda B\mathbf{v}$ , where the matrices  $A = e^{S_1}$  and  $B = e^{S_2}$  are positive-definite Hermitian. This is a typical class of symmetric generalized eigenvalue problems. The standard algorithm for solving this class of eigenvalue problems is to reduce them to the Hermitian eigenvalue problem  $B^{-1/2} A B^{-1/2} \mathbf{u} = \lambda \mathbf{u}$ , where  $\mathbf{u} = B^{1/2} \mathbf{v}$ . Parker and Joseph [64] studied the symmetric generalized eigenproblem through this idea for some types of  $A$  and  $B$  arising from applications in physics such that  $B^{-1/2} A B^{-1/2}$  is sparse and can be simulated efficiently. Later, Shao and Liu [65] proposed a new quantum algorithm for symmetric generalized eigenvalue problems by solving ordinary differential equations instead of Hamiltonian simulation. In this paper, because of the special properties of matrices  $e^{S_1}$  and  $e^{S_2}$ , we solve Eq. (4) by transforming it into  $e^{S_1/2} e^{-S_2} e^{S_1/2} \mathbf{u} = \lambda \mathbf{u}$ , where  $\mathbf{u} = e^{S_1/2} \mathbf{v}$ . It is a Hermitian eigenvalue problem where the matrices  $e^{S_1/2} e^{-S_2} e^{S_1/2}$  and  $e^{-S_1}$  can be simulated efficiently by optimal block Hamiltonian simulation.

TABLE III. Comparison between the QMEDR framework and the existing quantum linear DR algorithms in an end-to-end setting.

Algorithm	Input	Output <sup>a</sup>		
		Type I	Type II	Type III
quantum PCA <sup>b</sup>	multiple copies of the density operators, $m$	✓		
quantum PCA <sup>c</sup>	$X$ stored in a structured QRAM, $m$		✓	
quantum LDA <sup>d</sup>	$X$ stored in a structured QRAM, $m$	✓		
quantum LDA <sup>e</sup>	$X$ stored in a structured QRAM, $m$		✓	
quantum LPP <sup>f</sup>	$X$ stored in a structured QRAM, $m$	✓		
quantum AOP <sup>g</sup>		✓		
quantum NPE <sup>h</sup>	oracles preparing quantum states $\{ \mathbf{x}_i\rangle\}_{i=0}^{N-1}$ , $m$	✓		
quantum NPE <sup>i</sup>	$X$ stored in a structured QRAM, $m$	✓		
quantum DCCA <sup>j</sup>	$X$ stored in a QRAM, <sup>k</sup> $m$	✓		
QMEDR framework	$X$ stored in a structured QRAM, $m$			✓

<sup>a</sup>Here type I denotes that the output is quantum states corresponding to the column vectors of the transformation matrix, type II denotes that the output is a compressed analog-encoded state, and type III denotes that the output is a compressed digital-encoded state.

<sup>b</sup>From Ref. [8].

<sup>c</sup>From Ref. [24].

<sup>d</sup>From Ref. [9].

<sup>e</sup>From Ref. [32].

<sup>f</sup>From Ref. [30].

<sup>g</sup>From Refs. [10,26]. The quantum AOP algorithms output quantum superposition states corresponding to transformation matrices; here we classify them as type I for convenience.

<sup>h</sup>From Ref. [27]. Although two methods of getting compressed data are given in this algorithm, no superposition compressed state is constructed, so it is classified as type I.

<sup>i</sup>From Ref. [11].

<sup>j</sup>From Ref. [31].

<sup>k</sup>From Ref. [63].

## VI. CONCLUSION

In this paper we proposed the QMEDR framework, which is configurable and from which a series of alternative quantum DR algorithms can be derived. The applications on ELPP, EUDP, ENPE, and EDA showed the quantum superiority of this framework. The techniques we presented in this paper can be extended to solve many important computational problems, such as computing the matrix exponential and simulating the matrix exponential. Moreover, the QMEDR framework can also be regarded as a common framework in which to explore the quantization of other linear DR techniques. This work builds a bridge between quantum linear DR algorithms and other QML algorithms, which is helpful to overcome the curse of dimensionality and solve problems of practical importance. We hope it can inspire the study of QML.

## ACKNOWLEDGMENTS

This work was supported by Beijing Natural Science Foundation (Grant No. 4222031) and National Natural Science Foundation of China (Grants No. 61976024, No. 61972048, and No. 62171056).

## APPENDIX A: PROOF OF LEMMA 1

To prove Lemma 1, we will use the following tools.

*Lemma 4 (block encoding of controlled-Hamiltonian simulation [46]).* Let  $\mathcal{M} = 2^J$  for some  $J \in \mathbb{N}$ ,  $\gamma \in \mathbb{R}$ , and  $\epsilon \geq 0$ . Suppose that  $U$  is an  $(\alpha, a, \frac{\epsilon}{2(J+1)^2 \mathcal{M} \gamma})$  block encoding of the Hamiltonian  $H$ . Then we can implement a  $(1, a + 2, \epsilon)$  block

encoding of a controlled  $(\mathcal{M}, \gamma)$  simulation of the Hamiltonian  $H$ , with  $O(|\alpha \mathcal{M} \gamma| + J \frac{\log_2(J/\epsilon)}{\log_2 \log_2(J/\epsilon)})$  uses of controlled  $U$  or its inverse and with  $O(a|\alpha \mathcal{M} \gamma| + aJ \frac{\log_2(J/\epsilon)}{\log_2 \log_2(J/\epsilon)})$  three-qubit gates.

Note that here the controlled  $(\mathcal{M}, \gamma)$  simulation of the Hamiltonian  $H$  is defined as a unitary

$$W := \sum_{m=-\mathcal{M}}^{\mathcal{M}-1} |m\rangle\langle m| \otimes e^{im\gamma H}, \quad (\text{A1})$$

where  $|m\rangle$  denotes a (signed) bit string  $|b_J b_{J-1} \cdots b_0\rangle$  such that  $m = -b_J 2^J + \sum_{j=0}^{J-1} b_j 2^j$ .

*Theorem 3 (implementing a smooth function of a Hamiltonian [36]).* Let  $x_0 \in \mathbb{R}$  and  $r > 0$  be such that  $f(x_0 + x) = \sum_{l=0}^{\infty} a_l x^l$  for all  $x \in [-r, r]$ . Suppose that  $B > 0$  and  $\delta \in (0, r]$  are such that  $\sum_{l=0}^{\infty} (r + \delta)^l |a_l| \leq B$ . If  $\|H - x_0 I\| \leq r$  and  $\epsilon \in (0, 1/2]$ , then we can implement a unitary  $\tilde{U}$  that is a  $(B, b + O(\log_2(\frac{r \log_2(1/\epsilon)}{\delta})), B\epsilon)$  block encoding of  $f(H)$ , with a single use of a circuit  $V$ , which is a  $(1, b, \frac{\epsilon}{2})$  block encoding of the controlled  $(O(\frac{r \log_2(1/\epsilon)}{\delta}), O(\frac{1}{r}))$  simulation of  $H$ , and using  $O(\frac{r}{\delta} \log_2(\frac{r}{\delta \epsilon}) \log_2(\frac{1}{\epsilon}))$  two-qubit gates.

Now we provide the proof of Lemma 1. We first consider the case of  $c = 1$ . Let  $f(x) := e^x$  and observe that

$$\begin{aligned} f(1+x) &= \sum_{n=0}^{\infty} \frac{(1+x)^n}{n!} = \sum_{n=0}^{\infty} \frac{\sum_{l=0}^{\infty} \binom{n}{l} x^l}{n!} \\ &= \sum_{l=0}^{\infty} \frac{\sum_{n=0}^{\infty} \binom{n}{l}}{n!} x^l \end{aligned} \quad (\text{A2})$$

for all  $x \in [-1, 1]$ , where  $\binom{n}{l} = \frac{n(n-1)\dots(n-l+1)}{l!}$ . We choose  $x_0 := 1, r := 1 - \frac{1}{\kappa}$ , and  $\delta := \frac{1}{\kappa}$  and observe that

$$\begin{aligned} \sum_{l=0}^{\infty} (r + \delta)^l \left| \frac{\sum_{n=0}^{\infty} \binom{n}{l}}{n!} \right| &= \sum_{l=0}^{\infty} \frac{\sum_{n=0}^{\infty} \binom{n}{l}}{n!} \\ &= \sum_{l=0}^{\infty} \frac{2^n}{n!} = e^2 := B. \end{aligned} \quad (\text{A3})$$

$$O\left(\left|\alpha \kappa \log_2\left(\frac{1}{\epsilon}\right)\right| + \log_2\left[(\kappa - 1) \log_2\left(\frac{1}{\epsilon}\right)\right] \frac{\log_2\{\log_2[(\kappa - 1) \log_2(\frac{1}{\epsilon})/\epsilon]\}}{\log_2 \log_2\{\log_2[(\kappa - 1) \log_2(\frac{1}{\epsilon})/\epsilon]\}}\right) \quad (\text{A4})$$

controlled  $U$  or its inverse and with

$$O\left(a \left|\alpha \kappa \log_2\left(\frac{1}{\epsilon}\right)\right| + a \log_2\left[(\kappa - 1) \log_2\left(\frac{1}{\epsilon}\right)\right] \frac{\log_2\{\log_2[(\kappa - 1) \log_2(\frac{1}{\epsilon})/\epsilon]\}}{\log_2 \log_2\{\log_2[(\kappa - 1) \log_2(\frac{1}{\epsilon})/\epsilon]\}}\right) \quad (\text{A5})$$

three-qubit gates, where  $U$  is an  $(\alpha, a, \sigma)$  block encoding of  $H$  and  $\sigma = \frac{\epsilon}{4[\log_2[(\kappa-1) \log_2(1/\epsilon)]+1]^2 \kappa \log_2(1/\epsilon)}$ .

For simplicity, we delete some items with a small proportion and let  $T_U$  denote the cost of  $U$ . Then the total cost of  $\tilde{U}$  is

$$O\left(\alpha \kappa \log_2\left(\frac{1}{\epsilon}\right)(a + T_U) + (\kappa - 1) \log_2\left(\frac{\kappa - 1}{\epsilon}\right) \log_2\left(\frac{1}{\epsilon}\right)\right). \quad (\text{A6})$$

The case of  $c = -1$  can be proved similarly. Then Lemma 1 holds.

**APPENDIX B: PROOF OF  $\frac{1}{e\sqrt{m}} < \zeta_i < \frac{e}{\sqrt{m}}$**

Since  $\frac{1}{e} \leq \sigma_j < 1$ , then

$$\frac{\frac{1}{e} \sum_{j=0}^{M-1} \gamma_{ij}^2}{\sqrt{\sum_{i=0}^{m-1} (\sum_{j=0}^{M-1} \gamma_{ij}^2)^2}} < \zeta_i < \frac{e \sum_{j=0}^{M-1} \gamma_{ij}^2}{\sqrt{\sum_{i=0}^{m-1} (\sum_{j=0}^{M-1} \gamma_{ij}^2)^2}}. \quad (\text{B1})$$

Let  $U_1$  and  $U_2$  be two unitaries such that  $U_1|\mathbf{u}_j\rangle = |\mathbf{w}_j\rangle$  and  $U_2|\mathbf{e}_j\rangle = |\mathbf{u}_j\rangle$ , where  $|\mathbf{e}_j\rangle$  is the computational basis state. Then we have

$$\begin{aligned} \sum_{j=0}^{M-1} \gamma_{ij}^2 &= \sum_{j=0}^{M-1} (\langle \mathbf{w}_j | \mathbf{u}_i \rangle)^2 \\ &= \sum_{j=0}^{M-1} (\langle \mathbf{u}_j | U_1^\dagger | \mathbf{u}_i \rangle)^2 \end{aligned}$$

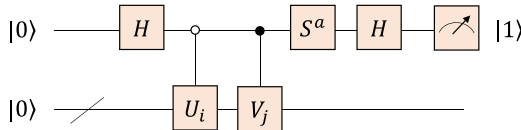


FIG. 6. Hadamard test circuit for measuring the real and imaginary parts of  $\langle \Psi_i | \Phi_j \rangle$  for any arbitrary unitary operations  $U_i : |0\rangle \mapsto |\Psi_i\rangle$  and  $V_j : |0\rangle \mapsto |\Phi_j\rangle$  [57]. Here  $S$  is the phase gate and  $a \in \{0, 1\}$ . The success probability of measuring  $|1\rangle$  is given by  $P_{ij} = \frac{1 - \text{Re}(\zeta \langle \Psi_i | \Phi_j \rangle)}{2}$ , where  $\zeta = 1$  if  $a = 0$  and  $\zeta = i$  if  $a = 1$ . Then  $\zeta = 1$  and  $\zeta = i$  recover the real and imaginary parts of  $\langle \Psi_i | \Phi_j \rangle$ , respectively.

Let  $\kappa \geq 2$  and  $H$  be a Hermitian matrix such that  $\frac{1}{\kappa} \leq H \leq I$ . If  $\epsilon \in (0, 1/2]$ , then we can implement a unitary  $\tilde{U}$  that is an  $(e^2, b + O(\log_2[(\kappa - 1) \log_2(\frac{1}{\epsilon})]), e^2\epsilon)$  block encoding of  $e^H$ , with a single use of a circuit  $V$ , which is a  $(1, b, \frac{\epsilon}{2})$  block encoding of controlled  $(O((\kappa - 1) \log_2(\frac{1}{\epsilon})), O(\frac{\kappa}{\kappa-1}))$  simulation of  $H$ , and using  $O((\kappa - 1) \log_2(\frac{\kappa-1}{\epsilon}) \log_2(\frac{1}{\epsilon}))$  two-qubit gates. Letting  $b := a + 2$  and  $\frac{\epsilon}{2} := \epsilon$ , then the circuit  $V$  uses

$$\begin{aligned} &= \sum_{j=0}^{M-1} (\langle \mathbf{e}_j | U_2^\dagger U_1^\dagger U_2 | \mathbf{e}_i \rangle)^2 \\ &= \sum_{j=0}^{M-1} (U_{ji})^2 \\ &= 1, \end{aligned} \quad (\text{B2})$$

where  $U_{ji}$  is the  $(j, i)$  entry of the unitary  $U = U_2^\dagger U_1^\dagger U_2$  and the last equation holds by  $U^\dagger U = I$ . Therefore,  $\frac{1}{e\sqrt{m}} < \zeta_i < \frac{e}{\sqrt{m}}$ .

**APPENDIX C: HADAMARD TEST FAILS TO COMPUTE  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle$  DIRECTLY**

The Hadamard test [57] is a modified version of the standard SWAP test [66] and their quantum circuits are shown in Figs. 6 and 7, respectively. The biggest difference between them is that Hadamard test can estimate the inner product of two quantum states (note that here we need unitaries to prepare these quantum states) and the SWAP test can only estimate the modular square of the inner product. Moreover,

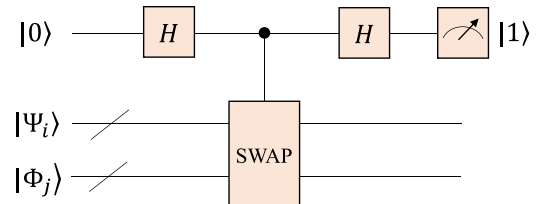


FIG. 7. SWAP test circuit for measuring the value of  $|\langle \Psi_i | \Phi_j \rangle|^2$  for any arbitrary states  $|\Psi_i\rangle$  and  $|\Phi_j\rangle$  [66]. The success probability of measuring  $|1\rangle$  is given by  $P_{ij} = \frac{1 - |\langle \Psi_i | \Phi_j \rangle|^2}{2}$ .

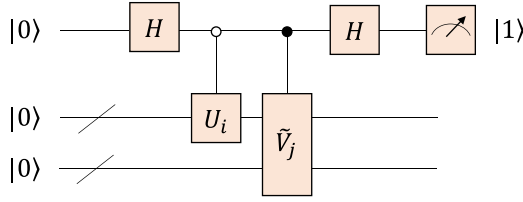


FIG. 8. Simple example.

if the measurement in the Hadamard test is replaced with parallel amplitude estimation [24], then Lemma 2 is derived.

Now we explain why the Hadamard test cannot be directly calculated  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle$  in the QMEDR framework. As shown in Fig. 6, if we want to compute  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle$ , we need two unitaries  $U_i : |0\rangle \mapsto |\mathbf{x}_i\rangle$  and  $V_j : |0\rangle \mapsto |\mathbf{v}_j\rangle$ . The  $U_i$  is naturally achieved with  $\mathbf{O}_2$ . However, for  $V_j$ , as discussed in step 3, we are only provided with the unitary  $|0\rangle|0\rangle \mapsto |\mathbf{v}_j\rangle|\mathbf{v}_j\rangle$  and the additional  $|\mathbf{v}_j\rangle$  cannot be avoided, which will inevitably impact the inner product value. For example, as illustrated in Fig. 8, with  $U_i$  and  $\tilde{V}_j : |0\rangle|0\rangle \mapsto |\mathbf{v}_j\rangle|\mathbf{v}_j\rangle$ , we can compute the value of  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle \langle 0 | \mathbf{v}_j \rangle$ , but we do not know what the value of  $\langle 0 | \mathbf{v}_j \rangle$  is. This is what causes the Hadamard test to fail. Furthermore, the same reason will lead to the failure of Lemma 2 when we utilize it to estimate the inner product  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle$  directly. Therefore, we only get the value of  $\langle \mathbf{x}_i | \mathbf{v}_j \rangle$  by computing its square rather than computing it directly.

#### APPENDIX D: THE QMEDR FRAMEWORK WHEN $e^{-S_2}e^{S_1}$ IS HERMITIAN

For Eq. (4), if  $e^{-S_2}e^{S_1}$  is Hermitian, one could rewrite it as

$$e^{-S_2}e^{S_1}\mathbf{v} = \lambda\mathbf{v} \quad (\text{D1})$$

and construct the state  $|\Psi\rangle := \sum_{i=0}^{M-1} |\lambda_i\rangle |\mathbf{v}_i\rangle |\mathbf{v}_i\rangle$ . Then the compressed digital-encoded state can be constructed in a similar way.

The core of constructing the state  $|\Psi\rangle$  lies in implementing the block encoding of  $e^{-S_2}e^{S_1}$ . Given  $U_1$  and  $U_2$ , which are the block encodings of  $S_1$  and  $S_2$ , respectively, we can implement an  $(e^2, a + O(\log_2[\kappa_1 \log_2(\frac{1}{\epsilon_1})]), e^2\epsilon_1)$  block encoding of  $e^{S_1}$  and an  $(e^2, b + O(\log_2[\kappa_2 \log_2(\frac{1}{\epsilon_2})]), e^2\epsilon_2)$  block encoding of  $e^{-S_2}$  by Lemma 1, where  $\epsilon_1, \epsilon_2 \in (0, \frac{1}{2}]$ . Then we implement an  $(e^4, a + b + O(\log_2[\kappa_1 \log_2(\frac{1}{\epsilon_1})] + \log_2[\kappa_2 \log_2(\frac{1}{\epsilon_2})]), e^4(\epsilon_1 + \epsilon_2))$  block encoding of  $e^{-S_2}e^{S_1}$  in time  $\tilde{O}(\max\{\alpha\kappa_1(a + T_1), \beta\kappa_2(b + T_2)\}) := T$ , where  $\epsilon_1, \epsilon_2 \in (0, \frac{1}{2}]$ . Next we can implement the unitary  $e^{i(e^{-S_2}e^{S_1})t}$  in time  $\tilde{O}(Tt)$  by Theorem 1. Based on this, we could produce the compressed digital-encoded state in time  $\tilde{O}(\frac{T \max_i \|\mathbf{x}_i\|^2 m \sqrt{M}}{\epsilon})$ .

- [1] P. W. Shor, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Sante Fe* (IEEE Computer Society, Washington, DC, 1994), pp. 124–134.
- [2] L. K. Grover, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia* (Association for Computing Machinery, New York, 1996), pp. 212–219.
- [3] A. W. Harrow, A. Hassidim, and S. Lloyd, *Phys. Rev. Lett.* **103**, 150502 (2009).
- [4] Z.-Q. Li, B.-B. Cai, H.-W. Sun, H.-L. Liu, L.-C. Wan, S.-J. Qin, Q.-Y. Wen, and F. Gao, *Sci. China Phys. Mech. Astron.* **65**, 290311 (2022).
- [5] S. Lloyd, M. Mohseni, and P. Rebentrost, [arXiv:1307.0411](https://arxiv.org/abs/1307.0411).
- [6] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, C. A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete *et al.*, [arXiv:1712.05771](https://arxiv.org/abs/1712.05771).
- [7] I. Kerenidis and J. Landman, *Phys. Rev. A* **103**, 042415 (2021).
- [8] S. Lloyd, M. Mohseni, and P. Rebentrost, *Nat. Phys.* **10**, 631 (2014).
- [9] I. Cong and L. Duan, *New J. Phys.* **18**, 073011 (2016).
- [10] S.-J. Pan, L.-C. Wan, H.-L. Liu, Q.-L. Wang, S.-J. Qin, Q.-Y. Wen, and F. Gao, *Phys. Rev. A* **102**, 052402 (2020).
- [11] S.-J. Pan, L.-C. Wan, H.-L. Liu, Y.-S. Wu, S.-J. Qin, Q.-Y. Wen, and F. Gao, *Chin. Phys. B* **31**, 060304 (2022).
- [12] L.-C. Wan, C.-H. Yu, S.-J. Pan, F. Gao, Q.-Y. Wen, and S.-J. Qin, *Phys. Rev. A* **97**, 062322 (2018).
- [13] L.-C. Wan, C.-H. Yu, S.-J. Pan, S.-J. Qin, F. Gao, and Q.-Y. Wen, *Phys. Rev. A* **104**, 062414 (2021).
- [14] H.-L. Liu, L.-C. Wan, C.-H. Yu, S.-J. Pan, S. J. Qin, F. Gao, and Q.-Y. Wen, *Adv. Quantum Technol.* **6**, 2300031 (2023).
- [15] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Nature (London)* **549**, 195 (2017).
- [16] P. C. Hammer, *SIAM Rev.* **4**, 163 (1962).
- [17] C. Bishop, *Pattern Recognition and Machine Learning* (Springer, Berlin, 2006).
- [18] X. He and P. Niyogi, in *Proceedings of the 16th International Conference on Neural Information Processing Systems, Whistler*, edited by S. Thrun, L. K. Saul, and B. Schölkopf (MIT Press, Cambridge, 2003), pp. 153–160.
- [19] J. Yang, D. Zhang, Z. Jin, and J.-Y. Yang, in *Proceedings of the 18th International Conference on Pattern Recognition, Hong Kong* (IEEE Computer Society, Washington, DC, 2006), Vol. 1, pp. 904–907.
- [20] X. He, D. Cai, S. Yan, and H.-J. Zhang, in *Proceedings of the Tenth IEEE International Conference on Computer Vision, Beijing* (IEEE Computer Society, Washington, DC, 2005), Vol. 2, pp. 1208–1213.
- [21] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, *IEEE Trans. Pattern Anal. Mach. Intell.* **19**, 711 (1996).
- [22] S. Yan, D. Xu, B. Zhang, H.-j. Zhang, Q. Yang, and S. Lin, *IEEE Trans. Pattern Anal. Mach. Intell.* **29**, 40 (2007).
- [23] S.-J. Wang, S. Yan, J. Yang, C.-G. Zhou, and X. Fu, *IEEE Trans. Image Process.* **23**, 920 (2014).
- [24] C.-H. Yu, F. Gao, S. Lin, and J. Wang, *Quantum Inf. Process.* **18**, 249 (2018).



- [25] X. He, L. Sun, C. Lyu, and X. Wang, *Quantum Inf. Process.* **19**, 309 (2020).
- [26] B.-J. Duan, J.-B. Yuan, J. Xu, and D. Li, *Phys. Rev. A* **99**, 032311 (2019).
- [27] J.-M. Liang, S.-Q. Shen, M. Li, and L. Li, *Phys. Rev. A* **101**, 032323 (2020).
- [28] Y. Li, R.-G. Zhou, R. Xu, W. Hu, and P. Fan, *Quantum Sci. Technol.* **6**, 014001 (2021).
- [29] A. Sornsaeng, N. Dangniam, P. Palittapongarnpim, and T. Chotibut, *Phys. Rev. A* **104**, 052410 (2021).
- [30] X.-Y. He, A.-Q. Zhang, and S.-M. Zhao, *Quantum Inf. Process.* **21**, 86 (2022).
- [31] Y.-M. Li, H.-L. Liu, S.-J. Pan, S.-J. Qin, F. Gao, and Q.-Y. Wen, *Quantum Inf. Process.* **22**, 163 (2023).
- [32] K. Yu, S. Lin, and G.-D. Guo, *Physica A* **614**, 128554 (2023).
- [33] J. van Apeldoorn and A. Gilyén, in *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 132, edited by C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, 2019), pp. 99:1–99:15.
- [34] G. H. Low and I. L. Chuang, *Quantum* **3**, 163 (2019).
- [35] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (ACM, New York, 2019), pp. 193–204.
- [36] S. Chakraborty, A. Gilyén, and S. Jeffery, in *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, Patras, 2019*, edited by C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 132 (Schloss Dagstuhl, Dagstuhl, 2019), pp. 33:1–33:14.
- [37] C. Shao, *J. Phys. A: Math. Theor.* **53**, 045301 (2020).
- [38] N. J. Higham, *Functions of Matrices* (Society for Industrial and Applied Mathematics, Philadelphia, 2008).
- [39] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 2nd ed. (Cambridge University Press, New York, 2011).
- [40] K. Mitarai, M. Kitagawa, and K. Fujii, *Phys. Rev. A* **99**, 012301 (2019).
- [41] Y.-M. Li, H.-L. Liu, S.-J. Pan, S.-J. Qin, F. Gao, D.-X. Sun, and Q.-Y. Wen, *Phys. Rev. A* **107**, 022421 (2023).
- [42] S.-J. Wang, H.-L. Chen, X.-J. Peng, and C.-G. Zhou, *Neurocomputing* **74**, 3654 (2011).
- [43] R. Ran, B. Fang, and X. G. Wu, *IEICE Trans. Inf. Syst.* **E101-D**, 1410 (2018).
- [44] T. Zhang, B. Fang, Y. Y. Tang, Z. Shang, and B. Xu, *IEEE Trans. Syst. Man Cyber. B* **40**, 186 (2010).
- [45] C. Moler and C. Van Loan, *SIAM Rev.* **45**, 3 (2003).
- [46] S. Chakraborty, A. Gilyén, and S. Jeffery, [arXiv:quant-ph/1804.01973v2](https://arxiv.org/abs/1804.01973v2) (We refer here specifically to Appendix A.2, which is not found in the published version (Ref. [36])).
- [47] I. Kerenidis and A. Prakash, *Phys. Rev. A* **101**, 022316 (2020).
- [48] J. van Apeldoorn and A. Gilyén, [arXiv:1804.05058](https://arxiv.org/abs/1804.05058) [quant-ph]. We refer specifically to Theorem 9, which is omitted from the published version (Ref. [33]).
- [49] H.-L. Liu, C.-H. Yu, L.-C. Wan, S.-J. Qin, F. Gao, and Q. Wen, *Physica A* **607**, 128227 (2022).
- [50] I. Kerenidis and A. Prakash, in *Proceedings of the Eighth Innovations in Theoretical Computer Science Conference, Berkeley, 2017*, edited by C. H. Papadimitriou, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 67 (Schloss Dagstuhl, Dagstuhl, 2017), pp. 49:1–49:21.
- [51] L. Wossnig, Z. Zhao, and A. Prakash, *Phys. Rev. Lett.* **120**, 050502 (2018).
- [52] C. Durr and P. Hoyer, [arXiv:quant-ph/9607014](https://arxiv.org/abs/quant-ph/9607014).
- [53] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, *Contemp. Math.* **305**, 53 (2002).
- [54] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, in *Advances in Neural Information Processing Systems, Vancouver, 2019*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett (Curran, Red Hook, 2019), Vol. 32.
- [55] S. Takahira, A. Ohashi, T. Sogabe, and T. S. Usuda, *Quantum Inf. Comput.* **22**, 965 (2021).
- [56] Q. T. Nguyen, B. T. Kiani, and S. Lloyd, *Quantum* **6**, 876 (2022).
- [57] N. Liu and P. Rebentrost, *Phys. Rev. A* **97**, 042315 (2018).
- [58] S. S. Zhou, T. Loke, J. A. Izaac, and J. B. Wang, *Quantum Inf. Process.* **16**, 82 (2017).
- [59] L. Ruiz-Perez and J. C. Garcia-Escartin, *Quantum Inf. Process.* **16**, 152 (2017).
- [60] N. Wiebe, A. Kapoor, and K. M. Svore, *Quantum Inf. Comput.* **15**, 316 (2015).
- [61] A. Ahuja and S. Kapoor, [arXiv:quant-ph/9911082](https://arxiv.org/abs/quant-ph/9911082).
- [62] P. Rebentrost, M. Mohseni, and S. Lloyd, *Phys. Rev. Lett.* **113**, 130503 (2014).
- [63] V. Giovannetti, S. Lloyd, and L. Maccone, *Phys. Rev. Lett.* **100**, 160501 (2008).
- [64] J. B. Parker and I. Joseph, *Phys. Rev. A* **102**, 022422 (2020).
- [65] C. Shao and J.-P. Liu, *Proc. R. Soc. A* **478**, 20210797 (2020).
- [66] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, *Phys. Rev. Lett.* **87**, 167902 (2001).