Physics Letters A 384 (2020) 126595

Contents lists available at ScienceDirect

Physics Letters A

www.elsevier.com/locate/pla

# A survey on HHL algorithm: From theory to application in quantum machine learning

Bojia Duan [a,1], Jiabin Yuan [a], Chao-Hua Yu [b], Jianbang Huang [a], Chang-Yu Hsieh [c,*]

[a] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[b] School of Information Technology, Jiangxi University of Finance and Economics, Nanchang 330032, China
[c] Tencent Quantum Laboratory, Shenzhen 518000, China

ARTICLE INFO

*Article history:*
Available online 28 May 2020
Communicated by M.G.A. Paris

*Keywords:*
Quantum computation
HHL algorithm
Quantum machine learning
Quantum circuit

ABSTRACT

The Harrow-Hassidim-Lloyd (HHL) algorithm is a method to solve the quantum linear system of equations that may be found at the core of various scientific applications and quantum machine learning models including the linear regression, support vector machines and recommender systems etc. After reviewing the necessary background on elementary quantum algorithms, we provide detailed account of how HHL is exploited in different quantum machine learning (QML) models, and how it provides the desired quantum speedup in all these models. At the end, we briefly discuss some of the remaining challenges ahead for HHL-based QML models and related methods.



## 1. Introduction

Quantum computation has come a long way since Feynman first advocated it as an efficient means to simulate complex quantum systems [1]. Over the course of the past twenty years or so, other exciting and useful applications have been discovered. Shor's [2] and Grover's [3] algorithms are prototypical examples often used to illustrate the capability of a quantum computer. At the core of these two famous algorithms are the quantum phase estimation subroutine [4] and the concept of amplitude amplification [5], respectively, that have been widely adapted by many subsequent quantum algorithms in order to achieve "quantum speedup". Not surprisingly, these two techniques also constitute the bedrock of the algorithms that we review in this article.

In 2009, Harrow, Hassidim and Lloyd proposed an elegant quantum algorithm [6] to solve the quantum linear system problem (QLSP), which is closely related but not exactly the same as solving the standard linear system of equations, $Ax = b$. This algorithm, now commonly referred to as the HHL algorithm, paves the way to address a plethora of mathematical tasks, such as solving the linear differential equations [7], least-square curve fitting [8], matrix inversions [9] etc. on a quantum computer with a provable quantum speedup under certain caveats (to be explicated below). The impact of HHL has been paramount as these mathematical tasks arise ubiquitously across all scientific and engineering disciplines, one may expect highly non-trivial and tremendously useful applications to be built upon the HHL algorithm. For instances, Clader et al. has proposed to use the HHL to accelerate the computation of electromagnetic scattering cross-sections of 3D objects [10], and Wang et al. has proposed to use HHL to estimate effective resistance of an electrical network [11]. While accelerating these engineering applications are clearly desirable, the most inspiring contribution of HHL is possibly its facilitation on the development of quantum machine learning (QML) such as the quantum version of the support vector machine [12] and the recommendation systems [13].

Indeed, built upon HHL, a rapidly growing list of QML algorithms has significantly expanded our understanding of what a quantum computer can accomplish beyond quantum simulations and classic examples of Shor's and Grover's algorithms etc. As discussed in later sections, at the core of these QML algorithms, HHL is used to solve some QLSPs, which are often the computational bottleneck of the training process. This fact implies that HHL-based QML algorithms inherit many inherent challenges of HHL. Failure to satisfy any caveats for HHL easily leads to the loss of quantum advantages in these algorithms. For instance, how to efficiently load classical data onto a quantum computer and the sparsity of a matrix are just two examples that could potentially limit the scope of applications of HHL. Hence, a clear understanding of HHL and its subsequently improved version is essential for further de-

* Corresponding author.
*E-mail addresses:* deja@nuaa.edu.cn (B. Duan), jbyuan@nuaa.edu.cn (J. Yuan), quantum.ych@gmail.com (C.-H. Yu), jianbang06@gmail.com (J. Huang), kimhsieh@tencent.com (C.-Y. Hsieh).

[1] Most of this work was done when the author was an intern in Tencent Quantum Laboratory.

https://doi.org/10.1016/j.physleta.2020.126595
0375-9601/

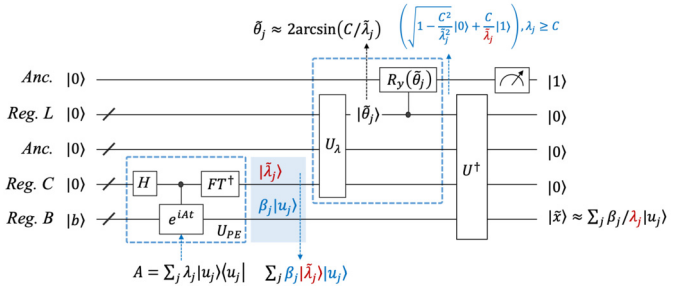


**Fig. 1.** Overview of HHL. The two inputs of HHL are $|b\rangle$ and $A$. The input quantum state $|b\rangle = \sum_{j=1}^{n} b_j |j\rangle$ refers to $b = (b_1, b_2, ..., b_n)^T$. The other input, a Hermitian matrix $A$, is used to construct the unitary matrix $e^{iAt}$ required for phase estimation ($U_{PE}$). If we cast $|b\rangle$ in the eigenspace of $A$ such that $|b\rangle = \sum_{j=1}^{n} \beta_j |u_j\rangle$. Then we find that $U_{PE}$ outputs the approximate eigenvalues $\left|\tilde{\lambda}_j\right\rangle$ of $A$. It is worth noting that the register storing the eigenvalues $\left|\tilde{\lambda}_j\right\rangle$ is entangled with the input register (light blue box), that is, the entangled state of the bottom two registers is $\sum_{j=1}^{n} \beta_j \left|\tilde{\lambda}_j\right\rangle |u_j\rangle$. And the values $\tilde{\lambda}_j$ and $u_j$ are hidden in $\sum_{j=1}^{n} \beta_j \left|\tilde{\lambda}_j\right\rangle |u_j\rangle$ which could not be extracted exactly after $U_{PE}$. HHL manipulates the eigenvalues of $A$ in parallel and these eigenvalues are not revealed to us. In the controlled rotation stage, $U_\lambda$ is a preprocessing which computes the state $|\tilde{\theta}_j\rangle$ with $\tilde{\theta}_j \approx 2 \arcsin\left(C/\tilde{\lambda}_j\right)$. It can be implemented by methods such as Taylor series truncation [14]. Then after the controlled $R_y$ rotations, the top ancilla qubit is approximate to $\sqrt{1 - C^2/\tilde{\lambda}_j^2}|0\rangle + C/\tilde{\lambda}_j|1\rangle$ up to $\epsilon$. $U^\dagger$ represents the inverse of all the operations before controlled $R_y$ rotations. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

velopment of QML algorithms such as the SVM and recommender systems etc.

We now provide a quick overview of HHL before delving into details in the following sections. For a given matrix $A \in \mathbb{R}^{n \times n}$ and a vector $b \in \mathbb{R}^n$, the HHL algorithm solves a system of linear equation, $Ax = b$, in a distinct way. For our purpose, it is sufficient to assume everything is real-valued. Operated on a quantum computer, the algorithm takes a normalized quantum state $|b\rangle$ in $\lceil \log n \rceil$ qubits as input, and returns a quantum state $|\tilde{x}\rangle$ with coefficients derived from $x = A^{-1}b$. Fig. 1 gives a high-level overview of the HHL in terms of a quantum-circuit implementation. The algorithm goes through three phases as follows:

1. A phase estimation ($U_{PE}$) module computes the eigenvalues $\lambda_j$ of $A$ and stores the approximate eigenvalues $\tilde{\lambda}_j$ in the basis states in the middle register as implied by the state $|\tilde{\lambda}_j\rangle$ in Fig. 1.
2. The inverse of the eigenvalues $\tilde{\lambda}_j^{-1}$ of $A$ are computed by a preprocessing $U_\lambda$ and a sequence of controlled $R_y$ rotations. These inverted eigenvalues are then encoded in the amplitudes of the top ancilla qubit.
3. Uncompute $U_\lambda$ and $U_{PE}$ to dis-entangle the middle register and measure the top ancilla qubit. If the measurement result is 1, it means that all the approximate eigenvalues $\tilde{\lambda}_j^{-1}$ have been applied to the amplitudes of $|b\rangle$ in the bottom register, and we obtain a quantum state approximate to $A^{-1}|b\rangle$.

A successful execution of the HHL algorithm requires a post-selection on the top ancilla qubit in Fig. 1. Hence, the amplitude amplification is often invoked at this stage to ensure the quantum circuit outputs the desired state $\left|\tilde{x}\right\rangle \approx A^{-1}|b\rangle$ with success probability close to 1.

The rest of this review article is organized as follows. In Sec. 2, we briefly introduce essential quantum algorithms relevant to HHL. In Sec. 3, we discuss HHL in detail and provide a detailed account of running a HHL experiment on IBM Q. In Sec. 4, we summarize how HHL is used in different machine-learning tasks such as classification and linear regression etc. Finally, we provide a perspective on challenges and outlook for further improvements of HHL and other related algorithms.

## 2. Quantum computing primer

In this section, fundamental quantum algorithms and quantum information processings relevant to the HHL algorithm are reviewed.

### 2.1. Encoding paradigms: from classical data to quantum states

The first challenge we encounter in the HHL or HHL-based algorithms is to encode classical data into quantum states, i.e. to create a superposition of all classical data. Currently, there are three different encoding paradigms as succinctly summarized below.

1. Basis encoding (digital encoding). Classical data, in the form of binary representation, is directly mapped onto the computational basis of qubits. For instance, numerical value 3 can be represented as 11 in binary form and be associated with the quantum state $|11\rangle$ under the digital encoding.
   The advantage of digital-encoding scheme is the ease of loading classical data [15] onto a quantum computer. One simply flips certain qubits from $|0\rangle$ to $|1\rangle$ by using a NOT gate. However, prohibitive amount of qubits (directly proportional to amount of data) will be consumed to simply represent classical data, such as floating point numbers commonly encountered in machine learning.
2. Amplitude encoding (analog encoding). The classical data is encoded in the amplitudes of the computational basis of qubits [16]. For example, a 2-dimensional data $x = (x_0, x_1)^T$ is encoded as a quantum state $|x\rangle = \frac{1}{\sqrt{x_0^2 + x_1^2}} (x_0 |0\rangle + x_1 |1\rangle)$.
   The advantage is that it only needs polynomial logarithm (with respect to data size) of qubits. The disadvantage is that it may be challenging to prepare the state, and one may need to perform tomography to estimate the probability amplitude for reading out the stored data.
3. Hamiltonian encoding. The most prominent example in this category is the adiabatic quantum computation model. The data may be encoded as the ground state of some physical Hamiltonian, such as the Ising model [17],
   $$H = -\sum_{<ij>} J_{ij} \sigma_i \sigma_j - \mu \sum_j h_j \sigma_j, \tag{1}$$
   where the numerical values of couplings $J_{ij}$ and local fields $h_j$ jointly specify the classical data. To load the classical data onto a quantum device, one simply solves for the ground state of $H$.
   It is straightforward to implement this Hamiltonian encoding by simply tuning physical parameters of some qubits. However, not every type of problems can be easily mapped onto $J_{ij}$ and $h_j$. As mentioned earlier, this method is most popularly used to solve optimization or sampling problems [18].
   Another Hamiltonian encoding paradigm, relevant to the HHL context, corresponds to simulating a classical Hermitian matrix $A$, which will be discussed in Section. 2.3.

In short, all three encoding schemes are used in different stages of the HHL or HHL-based QML algorithms. The outputs of the phase estimation subroutine are eigenvalues of a unitary operator stored in the digitally encoded form. The final output of the HHL algorithm is a quantum state $\left|\tilde{x}\right\rangle \approx A^{-1}|b\rangle$ in the analog-encoded form. Finally, to the end of preparing the input unitary $U = e^{iAt}$ for the phase estimation, the Hamiltonian encoding is invoked.

**Table 1**
Complexities of quantum algorithms for Hamiltonian simulation.

| Algorithm | Query complexity | Gate complexity$(t,\epsilon)$ | Gate complexity$(n)$ |
|---|---|---|---|
| Product formula (PF), 1st order [29] | $O\left(d^4t^2/\epsilon\right)$ | $O\left(d^4t^2/\epsilon\right)$ | $O\left(n^5\right)$ |
| Product formula (PF), (2k)th order [30] | $O\left(5^{2k}d^3t\left(dt/\epsilon\right)^{1/2k}\right)$ | $O\left(5^{2k}d^3t\left(dt/\epsilon\right)^{1/2k}\right)$ | $O\left(5^{2k}n^{3+1/k}\right)$ |
| Quantum walk [31] | $O\left(dt/\sqrt{\epsilon}\right)$ | $O\left(dt/\sqrt{\epsilon}\right)$ | $O\left(n^4\log n\right)$ |
| Fractional-query simulation [32] | $O\left(d^2t\frac{\log(dt/\epsilon)}{\log\log(dt/\epsilon)}\right)$ | $O\left(d^2t\frac{\log^2(dt/\epsilon)}{\log\log(dt/\epsilon)}\right)$ | $O\left(n^4\frac{\log^2 n}{\log\log n}\right)$ |
| Taylor series (TS) [33] | $O\left(d^2t\frac{\log(dt/\epsilon)}{\log\log(dt/\epsilon)}\right)$ | $O\left(d^2t\frac{\log^2(dt/\epsilon)}{\log\log(dt/\epsilon)}\right)$ | $O\left(n^3\frac{\log^2 n}{\log\log n}\right)$ |
| Linear combination of q.walk steps [34] | $O\left(dt\frac{\log(dt/\epsilon)}{\log\log(dt/\epsilon)}\right)$ | $O\left(dt\frac{\log^{3.5}(dt/\epsilon)}{\log\log(dt/\epsilon)}\right)$ | $O\left(n^4\frac{\log^2 n}{\log\log n}\right)$ |
| Quantum signal processing (QSP) [35] | $O\left(dt+\frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}\right)$ | $O\left(dt+\frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}\right)$ | $O\left(n^3\right)$ |

### 2.2. Quantum random access memory

In a variety of HHL-based algorithms, an analog-encoded quantum state $|x\rangle=\frac{1}{||x||}\sum_j x_j|j\rangle$ for $x\in\mathbb{R}^N$ is needed. In the early 2000s, different approaches, such as [19] and [20], were proposed for generating $|x\rangle$ directly in a quantum circuit. These state-preparation methods, however, are not sufficiently efficient. In 2008, a bucket-brigade architecture for Quantum Random Memory Access (QRAM) was introduced to better handle classical data [21]. In general, QRAM allows for querying multiple memory addresses $|j\rangle$ in superposition, and returns the corresponding data $\left|x_j\right\rangle$ stored in the $j$-th memory cell:

$$\sum_j \alpha_j\left|j\right\rangle\left|0\right\rangle \overset{QRAM}{\rightarrow} \sum_j \alpha_j\left|j\right\rangle\left|x_j\right\rangle. \tag{2}$$

In many QML-related applications, QRAM is used to store the classical information, i.e. QRAM is designed to store the $|x_j\rangle$ in the digital format. In these cases, one may probabilistically retrieve the desired analog-encoded state $|x\rangle$ as depicted in Appendix A.

Since the bucket-brigade QRAM was proposed, a broad range of proposals have been attempted to realize a QRAM either directly in a quantum circuit or via physical systems such as photonic and acoustic modes [22–28]. As it turns out that the challenge to build a QRAM hardware is paramount. We have yet to see a potentially scalable proposal for QRAM hardware.

The bucket-brigade architecture ensures that only $O\left(\text{polylog}N\right)$ of $O\left(N\right)$ logical gates would be activated during a single memory call [21,22]. The bucket-brigade architecture sounds very appealing because of a fundamental assumption that only errors incurred on active switches may compromise the query. Although this view has been challenged, for instance see [24] which advocates that all gates must be "actively" error corrected to maintain quantum coherence.

In addition to the method just discussed, there are other approaches to load classical data. For instance, in [13], an approach guided by a classical binary-tree data structure has been proposed to generate an analog-encoded quantum state in a quantum circuit, similar to the method given in [19]. It costs $O\left(N\right)$ time to store a classical data $x\in\mathbb{R}^N$ in the data structure. Once stored, it will take $O\left(\text{polylog}N\right)$ time for the quantum circuit to generate the desired state $|x\rangle=\frac{1}{||x||}\sum_j x_j|j\rangle$. And another approach to encode classical data into the analog-encoded states via QRAM is introduced in [16], which is briefly touched upon in Appendix A.

### 2.3. Hamiltonian simulation

Hamiltonian simulation is a task that seeks efficient algorithms to implement the time evolution of a quantum state under a given Hamiltonian $H$. It is leveraged as a subroutine in the HHL algorithm, and more generally applied in many QML algorithms. Particularly, it is often used for executing a sequence of controlled unitaries $U^{2^{j-1}}$ in the phase estimation subroutine (see detail in Section. 2.4).

**Definition 1** *(Hamiltonian simulation).* Given a Hamiltonian $H$, an evolution time $t$ and a simulation error $\epsilon$, the goal of Hamiltonian simulation is to find a quantum algorithm that performs a unitary $U$ satisfying $\left\|U-e^{-iHt}\right\|\leq\epsilon$, where $e^{-iHt}$ is the ideal evolution.

There are several quantum algorithms proposed for simulating Hamiltonian $H$, such as divide and conquer approach, the quantum walk and quantum signal processing etc. [29–36]. The query and gate complexity of these major approaches for efficient Hamiltonian simulations are summarized in Table 1. In this table, $d$ represents the sparsity of the Hamiltonian, and $n$ is the system size, i.e. the number of qubits. Notably, quantum signal processing [35], featured in the last row of Table 1, possesses the best complexity across all three categories. To facilitate discussions, we briefly introduce four well-established Hamiltonian simulations methods that one may adopt for the phase estimation in HHL.

#### 2.3.1. Product formula

The first proposed approach is the product formula (PF) algorithm with the 1st order approximation [29]. Suppose the goal is to simulate $H=\sum_{l=1}^{L}H_l$, we can combine individual simulations with the Lie product formula, e.g. for $L=2$, we then have $H=H_1+H_2$. If these two terms commute, the decomposition below is exact

$$e^{-i(H_1+H_2)t}=e^{-iH_1t}e^{-iH_2t}, \tag{3}$$

else,

$$e^{-i(H_1+H_2)t}=\lim_{r\to\infty}\left(e^{-iH_1t/r}e^{-iH_2t/r}\right)^r, \tag{4}$$

with

$$\left(e^{-iH_1t/r}e^{-iH_2t/r}\right)^r=e^{-i(H_1+H_2)t}+O\left(t^2/r\right). \tag{5}$$

To ensure error at most $\epsilon$, take $r=O\left(\left(||H||t\right)^2/\epsilon\right)$. Moreover, [30] considers a higher-order product formula to get a better approximation,

$$\left(e^{-iH_1t/2r}e^{-iH_2t/r}e^{-iH_1t/2r}\right)^r=e^{-i(H_1+H_2)t}+O\left(t^3/r^2\right). \tag{6}$$

And a Hamiltonian that is a sum of $L$ terms is:

$$e^{-i(H_1+\cdots+H_L)t}=\lim_{r\to\infty}\left(e^{-iH_1t/r}\cdots e^{-iH_Lt/r}\right)^r. \tag{7}$$

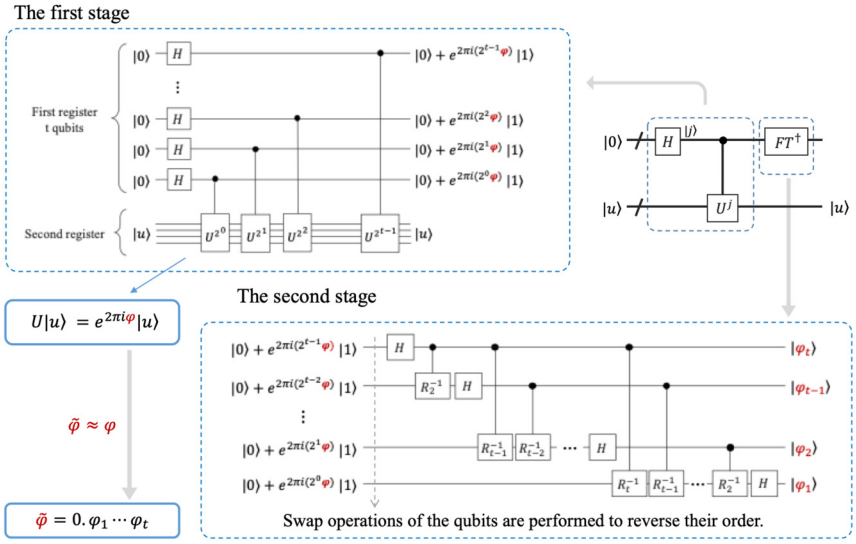


**Fig. 2.** Overview of phase estimation. (top right): This circuit depicts the whole algorithm. (top left): This circuit extracts the eigenvalue of $U$ as being 'kicked back' to the control register (see [37] for detail). (bottom right): The circuit for the inverse of quantum Fourier transform (QFT).

#### 2.3.2. Taylor series

Another approach is the divide and conquer method based on the Taylor series (TS), implementing the series $e^{-iHt} = \sum_{k=0}^{\infty}(-iHt)^k/k! \approx \sum_{k=0}^{K}(-iHt)^k/k!$ [33]. Write $H = \sum_l \alpha_l H_l$ with $H_l$ unitary, then we have a linear combination of unitaries:

$$\sum_{k=0}^{K}\sum_{l_1,\cdots,l_k}\frac{(-it)^k}{k!}\alpha_{l_1}\cdots\alpha_{l_k}H_{l_1}\cdots H_{l_k} \tag{8}$$

Using LCU lemma (see Lemma 9 in Appendix B), one can apply $U$ onto $|\psi\rangle$ with the success probability of $|\sin\theta|^2$,

$$|0\rangle|\psi\rangle \mapsto \sin\theta\,|0\rangle U|\psi\rangle + \cos\theta\,|\Phi\rangle \tag{9}$$

To boost the success probability close to 1, one may invoke amplitude amplification algorithm (to be explicated below). As shown in Table 1, this Taylor-Series approach features favorable complexity.

#### 2.3.3. Quantum walk

Another class of algorithms are based on quantum walk [31]. The basic idea is to define a quantum walk $U$ which is analogous to a given Hamiltonian $H$. Specifically, define a quantum state $|\psi_j\rangle$ from $H$:

$$|\psi_j\rangle := \frac{1}{\sqrt{||\mathrm{abs}(H)||}}\sum_{k=1}^{N}\sqrt{H^*_{jk}\frac{d_k}{d_j}}\,|j,k\rangle\,, \tag{10}$$

where $\mathrm{abs}(H) := \sum_{j,k=1}^{N}|H_{jk}||j\rangle\langle k|$, and $|d\rangle := \sum_{j=1}^{N} d_j|j\rangle$ is a principal eigenvector of $\mathrm{abs}(H)$ with eigenvalue $||\mathrm{abs}(H)||$. And define a quantum walk operator $U$ from $|\psi_j\rangle$:

$$U := iS\left(2TT^\dagger - I\right), \tag{11}$$

where $T = \sum_{j=1}^{N}|\psi_j\rangle\langle j|$ and $S := \sum_{j,k=1}^{N}|j,k\rangle\langle k,j|$. Suppose $\frac{H}{||\mathrm{abs}(H)||}|\lambda\rangle = \lambda|\lambda\rangle$. Then using the spectral theorem (see Theorem 10 in Appendix B) and phase estimation (PE), one may simulate the Hamiltonian dynamics:

$$\begin{aligned}|\lambda\rangle &\mapsto |\lambda\rangle\left|\widetilde{\arcsin\lambda}\right\rangle \qquad \text{(PE)}\\ &\mapsto e^{-i\lambda t}|\lambda\rangle\left|\widetilde{\arcsin\lambda}\right\rangle\\ &\mapsto e^{-i\lambda t}|\lambda\rangle \qquad \text{(inverse PE)}\end{aligned} \tag{12}$$

### 2.4. Phase estimation

Phase estimation [4] is used to estimate the phase $\varphi_u$ of an eigenvalue $e^{2\pi i\varphi_u}$ of a unitary operator $U$ (when applied to a corresponding eigenstate $|u\rangle$) in the form of a digital encoding. In general, for an input $\sum_u c_u|u\rangle$ composed of a linear combination of eigenstate $|u\rangle$ of $U$, the output of phase estimation is a state close to $\sum_u c_u|\tilde{\varphi}_u\rangle|u\rangle$, where $\tilde{\varphi}_u$ is a good approximation to the phase $\varphi_u$ of the eigenvalue $e^{2\pi i\varphi_u}$. Since the spectral decomposition is a powerful and commonly used technique, the phase estimation has been widely used in various quantum algorithms. It is often the source of the exponential speedup claimed by these algorithms. In the context of the HHL algorithm, it is used to effectively rotate and solve the QLSP in the eigenbasis of the input matrix $A$ (see Fig. 1 for detail). HHL requires the spectral decomposition of $A$; therefore, a Hamiltonian simulation $e^{iAt}$ is invoked to get $e^{iAt}|u_j\rangle = e^{i\lambda_j t}|u_j\rangle$ for each eigenstate $|u_j\rangle$ of $A$.

#### 2.4.1. Implementations in quantum circuits

The algorithm is succinctly summarized in Fig. 2. Two inputs of the phase estimation are $|u\rangle$ and $U$, where $|u\rangle$ is an eigenvector of $U$. Phase estimation uses two registers of qubits. The top register is initialized to $|0\rangle$ and stores the phase at the end, and the bottom register holds the input $|u\rangle$. The output of the algorithm is a $t$-qubit estimated $\tilde{\varphi}$ of phase $\varphi$, where $U|u\rangle = e^{2\pi i\varphi}|u\rangle$.

As shown in the circuit schematics, two sets of quantum registers with size $t$ and $\log m$, respectively, are required. The $t$-qubit register (top register in Fig. 2) is initialized in $|0\rangle^{\otimes t}$ state, while the $\log m$ qubit register holds an eigenstate $|u\rangle$. At the first stage of the algorithm (top left panel of Fig. 2), the top register is first put into a uniform superposition of all $2^t$ possible configurations due to the applications of Hadamard gates. Next, a set of controlled rotations $U^{2^{j-1}}$ (conditioned on the $j$-th qubit in the top register) is applied to the bottom register holding $|u\rangle$. At the end of this stage, the overall wave function reads,

$$|\psi\rangle = \frac{1}{\sqrt{2^t}}\otimes_{n=1}^{t}\left(|0\rangle + e^{2\pi i(2^{n-1})\varphi}|1\rangle\right)|u\rangle, \tag{13}$$

where $\varphi$ is the corresponding eigenvalue for $U|u\rangle = e^{2\pi i\varphi}|u\rangle$. Note that the phase factor encoded in Eq. (13) is the $t$-bit estimated phase $\tilde{\varphi}$, i.e. $\tilde{\varphi} = 0.\varphi_1\ldots\varphi_t \approx \varphi$, where $\varphi_1, \ldots, \varphi_t \in \{0, 1\}$.

At the second stage, the inverse Quantum Fourier Transform is applied to the top register. The detail of the circuit appears at the bottom right of Fig. 2. As explicitly indicated, the top register stores a $t$-qubit estimated $\tilde{\varphi}$ of phase $\varphi$ at the end.

Suppose we wish to approximate $\varphi$ to an accuracy $2^{-n}$, as shown in [4], with success probability at least $1-\epsilon$, we choose:

$$t = n + \lceil log\left(2 + \frac{1}{2\epsilon}\right)\rceil. \tag{14}$$

It implies that $n$ qubits in the first register are used to guarantee the accuracy of the output, and the remaining $t-n$ ancilla qubits are used to improve the success probability of phase estimation. With larger $t-n$, the algorithm's output is more likely to approach the optimal estimate.

Fig. 2 depicts the special case that the input state $|u\rangle$ is an eigenvector of the input matrix $U$. More generally, the input state $|b\rangle$ is supposed to be a superposition of the eigenvectors of an input matrix $A$, namely $|b\rangle = \sum_{j=1}^{n} \beta_j |u_j\rangle$ for HHL type of algorithms.

#### 2.4.2. Generalization in QML

A range of QML algorithms based on phase estimation have more general input structures, such as whether the two inputs $|b\rangle$ and $A$ are derived from the same data. Below, we briefly introduce these two classes of algorithms.

1. $|b\rangle$ and $A$ are not derived from the same data.
   Typical algorithms include HHL, quantum support vector machine (QSVM) [12], quantum linear regression [8,38–41], quantum singular value decomposition (QSVD) [42] and so on. For all these cases, the dimension of $A$ matrix equals to that of $|b\rangle$. Similar to the HHL case, the input Hermitian matrix $A$ is used to generate the unitary matrix $e^{iAt}$ and applied to the input $|b\rangle$.
2. $|b\rangle$ and $A$ are derived from the same data.
   Typical algorithms include quantum principle component analysis (QPCA) [43], quantum singular value thresholding (QSVT) [44,45], quantum recommendation system (QRS) [13] and so on.
   One remarkable case is that the dimension of $A$ can be smaller than $|b\rangle$. As an example of this case, QSVT is described in Section. 4.4.

### 2.5. Amplitude amplification

Amplitude amplification was independently discovered and presented in [46] and [47], respectively. It is a generalization of Grover's search algorithm [3]. If a quantum algorithm $\mathcal{A}$ can only be implemented in a probabilistic fashion, then amplitude amplification is a technique for boosting the success probability. In particular, it is used to improve the success probability of getting the solution state $|x\rangle$ for $Ax = b$ in HHL.

To better illustrate how AA is used in a general context, we give a simple example. Consider a Boolean function $\chi : X \to \{0,1\}$ that endows a set $X$ with a bipartite partition. More precisely, we call an element $x$ of $X$ good if $\chi(x) = 1$ or bad otherwise. If a quantum algorithm $\mathcal{A}$ is used to prepare a state, $\mathcal{A}|0\rangle = \sum_{x\in X} \alpha_x |x\rangle$, it is a probabilistic event to extract a good element $x$ upon measurement of $\mathcal{A}|0\rangle$. By following the AA protocol, as displayed in Fig. 3, the success probability to observing a good $x$ may be tremendously boosted to nearly certainty. Using this example as a guide, we introduce the formal definition of AA in the following theorem.

**Theorem 1** *(Quadratic speedup with known initial success probability [5]). Let $\mathcal{A}$ be a quantum algorithm without using measurement and with a known initial success probability $a > 0$. Let $\chi : \mathbb{Z} \to \{0,1\}$ be any Boolean function. There exists a quantum algorithm that finds a good solution with certainty by repeatedly applying $\mathcal{A}$ and $\mathcal{A}^{-1}$ in an alternating sequence which is in $\Theta\left(\frac{1}{\sqrt{a}}\right)$ in the worse case.*

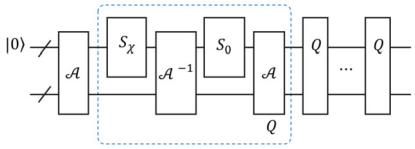


**Fig. 3.** The sketch of quantum circuit of amplitude amplification. The top register stores the input of the algorithm $\mathcal{A}$, and the bottom register refers to the ancilla used in $\mathcal{A}$.

Theorem 1 demands the initial success probability $a > 0$ of $\mathcal{A}$ to be known. This is indeed the case for HHL and most HHL-based QML algorithms, because the controlled rotation angles (stage 2 in Fig. 1) determine the value of $a > 0$. More general usages of amplitude amplification are discussed in [5].

Fig. 3 depicts a quantum circuit for the amplitude amplification algorithm with the guaranteed quadratic speedup mentioned in Theorem 1. The operator $Q$ in Fig. 3 is defined as follows,

$$Q = Q(\mathcal{A}, \chi, \phi, \varphi) = -\mathcal{A} S_0(\phi) \mathcal{A}^{-1} S_\chi(\varphi) \tag{15}$$

where

$$S_\chi(\varphi)|x\rangle = \begin{cases} e^{i\varphi}|x\rangle & if\ \chi(x) = 1 \\ |x\rangle & if\ \chi(x) = 0 \end{cases} \tag{16}$$

and

$$S_0(\phi)|x\rangle = \begin{cases} e^{i\phi}|x\rangle & if\ x = 0 \\ |x\rangle & if\ x \neq 0 \end{cases} \tag{17}$$

By appropriately choosing $\phi$ and $\varphi$ in each $Q$ block of Fig. 3, amplitude amplification is guaranteed to achieve high success probability. One choice is to apply $Q(\mathcal{A}, \chi, \pi, \pi)$ for a number of $\lfloor m \rfloor$ times to the initial state $|\psi\rangle = \mathcal{A}|0\rangle$ (where $m = \pi/4\theta_a - 1/2$, $\sin^2(\theta_a) = a$ and $0 < \theta_a \leq \pi/2$), following by $Q(\mathcal{A}, \chi, \phi, \varphi)$ at the very last iteration. This choice induces a good solution with certainty [48].

## 3. HHL algorithm

In Section 1 we motivate the need for having the HHL algorithm, and briefly describe how it works. Now we take a deeper look at details.

HHL is an algorithm to solve the QLSP $|x\rangle = A^{-1}|b\rangle / |A^{-1}|b\rangle|$. The inverse matrix $A^{-1}$ is obtained by using the phase estimation along with a unitary operator $e^{iAt}$ and a controlled rotation to effectively generate the inverse transformation $\lambda_i^{-1}$ on the matrix $A$'s eigenspectrum.

To ensure the unitarity of $e^{iAt}$, $A$ has to be Hermitian. For non-Hermitian matrix $A$, one may define an isometry superoperator $\mathbf{I}$ such that

$$\mathbf{I} : A \mapsto \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}. \tag{18}$$

Obviously, $\mathbf{I}(A)$ is Hermitian, therefore one can solve $\mathbf{I}(A)\vec{y} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}$ to obtain $\vec{y} = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix}$. In the following, we simply assume that $A$ is Hermitian.

### 3.1. Theoretical foundations

HHL is often claimed to provide exponential quantum speedup for the QSLP. However, this claim does come with four caveats which have been touched upon in the original HHL article [6] and thoroughly summarized in Aaronson's remarks [49]. The first caveat is the need for an efficient approach to prepare the input state $|b\rangle$. For QML application, $|b\rangle$ is essentially a superposition of classical data. If it takes $\mathcal{O}(n)$ time to load classical data onto the quantum register then all exponential speedup is suppressed. Currently, QRAM is required in most QML algorithms to efficiently prepare $|b\rangle$.

For the other two caveats on the matrix $A$. We refer to the following theorem,

**Theorem 2** *(Time complexity of HHL [6]). Let $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$ be the inputs of the matrix inversion estimation problem. Assume that $A$ is $s$-sparse and efficiently row-computable, and the condition number of $A$ is $\kappa$. Then there exists a quantum algorithm that outputs a quantum state proportional to $A^{-1}b$ within accuracy $\varepsilon$ in time $\tilde{O}\left(\log(n)s^4\kappa^2/\varepsilon\right)$, where the notation $\tilde{O}$ suppresses more slowly-growing terms [6,8].*

Theorem 2 requires that each row of $A$ has at most $s$ non-zero entries and given a row index these entries can be computed in time $O(s)$, and all the non-zero singular values of $A$ are in the range $[1/\kappa, 1]$. When all the conditions are satisfied, an efficient Hamiltonian simulation of $e^{iAt}$ may be achieved according to the time complexity specified in the theorem. Hence, sparsity $s$ and condition number $\kappa$ of $A$ critically affect the exponential speedup in the original HHL algorithm.

Finally, the last caveat is that we only obtain the quantum state $|x\rangle$ at the end of HHL. Clearly, it takes $\mathcal{O}(N)$ time to fully characterize every component of $|x\rangle$. Hence, the exponential speedup is again suppressed. Therefore, HHL is only useful if the quantity of interests could be cast as measurements of a Hermitian matrix $M$, i.e. $\langle x|M|x\rangle$.

While these caveats are important, they are not all unsurmountable obstacles. For instance, recent progress allows us to apply HHL to non-sparse matrices with provable quantum speedup. This triumph has also found important applications, such as the quantum principle component analysis introduced in the Appendix D. In this section, we provide an essential theorem relevant to this aforementioned progress.

**Theorem 3** *(Density matrix exponentiation [43]). Given $n = O(t^2/\varepsilon)$ copies of a quantum state with density matrix $\rho$, the unitary operation $e^{-i\rho t}$ for time $t$ can be implemented within error $\varepsilon$.*

To prove Theorem 3, let $\sigma$ be the quantum state on which $e^{-i\rho t}$ operates. For each copy of $\rho$, perform the unitary operation $e^{-iSt/n}$ of the swap operator $S$ on $\rho$ and $\sigma$, which is equivalent to performing $e^{-i\rho t/n}$ on $\sigma$ due to the fact that

$$\mathrm{tr}_1\left\{e^{-iSt/n}(\rho\otimes\sigma)e^{iSt/n}\right\} = e^{-i\rho t/n}\sigma e^{i\rho t/n} + O(t^2/n^2),$$

according to the results of Ref. [43]. Note that $S$ is 1-sparse and thus $e^{-iSt/n}$ can be implemented efficiently [30,50,51]. So repeatedly implementing $e^{-iSt/n}$ $n$ times for these $n$ copies of $\rho$ makes $e^{-i\rho t}$ performed on $\sigma$ within error $O(t^2/n)$, and thus it is easy to see Theorem 3 holds directly by simple calculation.

Theorem 3 allows us to reveal the eigenvalues and eigenvectors of $\rho$ in quantum parallel by performing phase estimation of $e^{-i\rho t}$ on $\rho$ itself. If $\rho$ is approximately dominated by $R$ largest eigenvalues and eigenvectors, i.e., $\rho$ admits a $R$-rank approximation, it generally requires $t = O(1/R)$ to resolve the eigenvalues via phase estimation. This means that density matrix exponentiation would be quite efficient if the density matrices exponentiated are (approximately) low-rank. Moreover, it extends HHL's algorithm to efficiently handle low-rank density matrices.

### 3.2. Algorithm

We now discuss HHL (see Fig. 1) in more details. To facilitate the discussion, we shall first re-cast phase estimation in a more concise form [44],

$$\begin{aligned} U_{\mathrm{PE}} &= U_{\mathrm{PE}}(A) \\ &= \left(F_T^\dagger \otimes I^B\right)\left(\sum_{\tau=0}^{T-1}|\tau\rangle\langle\tau|^C \otimes e^{iA\tau t_0/T}\right)\left(H^{\otimes t}\otimes I^B\right), \end{aligned} \tag{19}$$

where register $C$ with $t$ qubits is used to store the estimated eigenvalues of a Hermitian matrix $A$, register $B$ with $s$ qubits stores the input state $|b\rangle$ (and will also hold the desired output state $|x\rangle$ for HHL), $F_T^\dagger$ is the inverse quantum Fourier transform, and $\sum_{\tau=0}^{T-1}|\tau\rangle\langle\tau|^C \otimes e^{iA\tau t_0/T}$ is the conditional Hamiltonian simulation, where $T$ is a large constant and $t_0 = O(\kappa/\epsilon)$, $\kappa$ is the condition number of $A$ and $\epsilon$ is the additive error achieved in the output state $|x\rangle$ [6]. Similar to the application of a sequence of controlled-$U^{2^j}$ in the first stage of Fig. 2, it requires to apply a sequence of controlled operation $(|0\rangle\langle 0|)_j \otimes I + (|1\rangle\langle 1|)_j \otimes e^{iA2^{j-1}t_0/T}$ to implement the Hamiltonian simulation in $\sum_{\tau=0}^{T-1}|\tau\rangle\langle\tau|^C \otimes e^{iA\tau t_0/T}$.

In addition to the two registers (Reg. $C$ and Reg. $B$), HHL algorithm requires another ancilla qubit to be used for the controlled-rotation stage in Fig. 1.

Next, a step-by-step procedure of HHL is given in Algorithm 1. For a given input $|b\rangle$ with expanding form $|b\rangle = \sum_{j=1}^n \langle u_j|b\rangle|u_j\rangle$, where $|u_j\rangle$ is the eigenstate of $A$. The output of HHL is a quantum state $|\tilde{x}\rangle \approx A^{-1}|b\rangle = \sum_{j=1}^n \langle u_j|b\rangle/\lambda_j|u_j\rangle$.

**Algorithm 1** HHL algorithm.

**Input:** Quantum state $|b\rangle$, Unitary $e^{iAt}$
**Output:** Quantum state $|\tilde{x}\rangle \approx A^{-1}|b\rangle$
Algorithm Start:
1. Prepare the initial quantum state: $|\psi_0\rangle = |0\rangle^a|0\cdots 0\rangle^C|b\rangle^B = |0\rangle^a|0\cdots 0\rangle^C\sum_{j=1}^n\langle u_j|b\rangle|u_j\rangle^B$
2. Perform the unitary operation $U_{\mathrm{PE}}(A)$ on the state: $|\psi_0\rangle \to |\psi_1\rangle = |0\rangle^a\sum_{j=1}^n\langle u_j|b\rangle|\tilde{\lambda}_j\rangle^C|u_j\rangle^B$
3. Apply a controlled rotation $R$ to the ancilla qubit, controlled by Reg. C: $|\psi_1\rangle \to |\psi_2\rangle = \sum_{j=1}^n\left(\sqrt{1-\frac{\gamma^2}{\tilde{\lambda}_j^2}}|0\rangle + \frac{\gamma}{\tilde{\lambda}_j}|1\rangle\right)^a\langle u_j|b\rangle|\tilde{\lambda}_j\rangle^C|u_j\rangle^B$
4. Uncompute the Reg. C and Reg. B: $|\psi_2\rangle \to |\psi_3\rangle = \sum_{j=1}^n\left(\sqrt{1-\frac{\gamma^2}{\tilde{\lambda}_j^2}}|0\rangle + \frac{\gamma}{\tilde{\lambda}_j}|1\rangle\right)^a|0\cdots 0\rangle^C\langle u_j|b\rangle|u_j\rangle^B$
▷ $\gamma$ is a constant that become irrelevant after measurement, owing to the normalization of the projected wave function.
5. *output* ← *MeasureAncilla*()
**if** output is $|1\rangle$ **then**
**return** $|\tilde{x}\rangle \approx \sum_{j=1}^n\langle u_j|b\rangle/\lambda_j|u_j\rangle$
▷ $|\tilde{x}\rangle$ is stored in the Reg. $B$
**else**
*goto* Algorithm Start
**end if**
Algorithm End

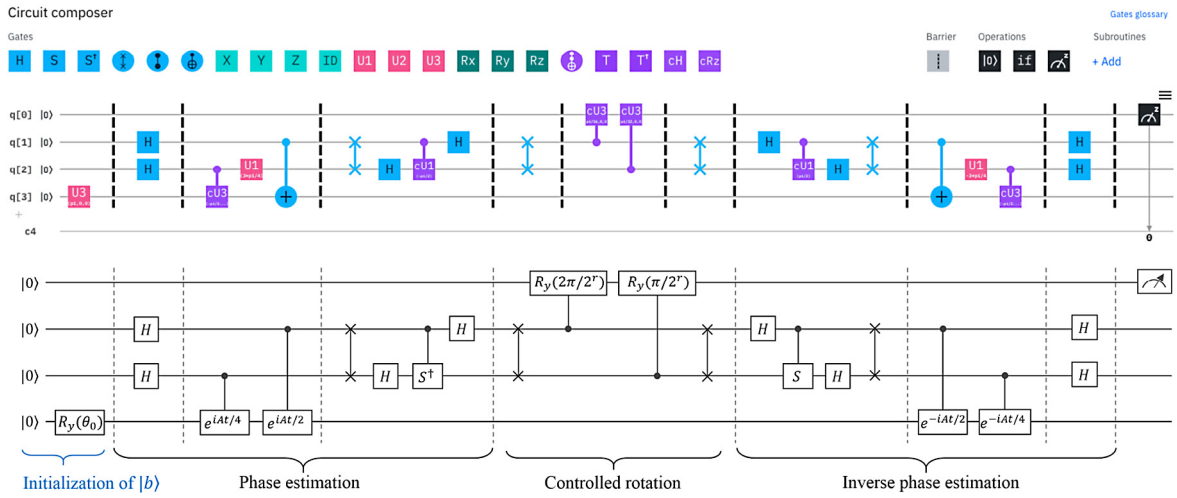


**Fig. 4.** The 4-qubit quantum circuit of HHL algorithm. (Top): The quantum circuit of HHL implemented on IBM Q with IBM logic gates. (Bottom): The theoretical quantum circuit of HHL.

### 3.3. Experiments of HHL

Several experimental demonstrations of some small-scale instances of HHL have already been realized in both photonics [52,53] and nuclear clear magnetic resonance (NMR) [54] experiments. These successful experiments certainly have lent many supports to the growing optimisms for further developments of quantum computations. Moreover, with a rapidly growing list of quantum cloud platforms, such as QDK, Origin Quantum, Quantum Inspire, Amazon Braket, and IBM Q [55–59], it is becoming easier to get hands on and attempt programming quantum algorithms such as HHL on a hardware than ever before. Below, we illustrate how to program HHL to solve a small-scale QLSP on the IBM Q's 4-qubit device, which should contribute to a better understanding of HHL [59] through hands-on experiments.

#### 3.3.1. 4-bit circuit on IBM Q

IBM Q provides two tools for programming quantum algorithms: web-based circuit composer with drag-and-drop graphical user interface and Qiskit library for python programming environment. Both tools have access to the same hardware or classical simulator. For this demonstration, we simply provide the HHL algorithm in quantum assembly language (QASM) file that works on the circuit composer.

We will solve $Ax = b$ having 2 unknown variables. The matrix $A = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$ has two eigenvalues: $\lambda_1 = 1$ and $\lambda_2 = 2$. This simple example ensures that both eigenvalues can be exactly stored using two qubits without truncation error. We will solve $x = A^{-1}b$ with various $b$ vectors in this demonstration. The detailed circuit implemented on IBM Q and the compared theoretical circuit are shown as in Fig. 4. And the code of this IBM circuit is made available on Github [60].

IBM circuit in Fig. 4 (top) has four qubits, where $q[0]$ is the ancilla, $q[3]$ stores the input $|b\rangle$ and $q[1], q[2]$ are used for storing the eigenvalues of $A$. The circuit is most composed of elementary quantum gates, such as Hadamard $H$, swap operator and CNOT. Additionally, two single-qubit gates $U_1$ and $U_3$ are defined as follows,

$$U_1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix},$$
$$U_3(\theta, \phi, \lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2) \end{pmatrix}. \quad (20)$$

A special case for $U_3$ is $U_3(\theta, 0, 0) = R_y(\theta)$, where $R_y$ is the rotation operator with respect to the $y$ axis. In the following, the controlled-$U$ is written as c-$U$.

Our experiment involves five steps.

**Algorithm 2** 4-qubit HHL circuit on IBM Q.

**Input:** $q[0]q[1]q[2]q[3] \leftarrow |0000\rangle$
**Output:** $|x\rangle$
**Initialization:**
$q[3] \leftarrow |b\rangle \leftarrow U_3(\theta, 0, 0)\, q[3]$
▷ Different $|b\rangle$ can be prepared by adjusting $\theta$.
▷ See examples of $|b\rangle$ in Table 2.
**Phase estimation:**
$q[1]q[2] \leftarrow (H \otimes H)\,(q[1]q[2])$
$q[2]q[3] \leftarrow c-U_3\left(-\frac{\pi}{2}, -\frac{\pi}{2}, \frac{\pi}{2}\right)(q[2]q[3])$
$q[2] \leftarrow U_1\left(\frac{3\pi}{4}\right) q[2]$
$q[1]q[3] \leftarrow CNOT\,(q[1]q[3])$
▷ see the equivalence in C.1.
$q[1]q[2] \leftarrow Inverse\_QFT\,(q[1]q[2])$
**Controlled rotation:**
$q[1]q[2] \leftarrow Swap\,(q[1]q[2])$
$q[1]q[0] \leftarrow c-U_3\left(\frac{\pi}{16}, 0, 0\right)(q[1]q[0])$
$q[2]q[0] \leftarrow c-U_3\left(\frac{\pi}{32}, 0, 0\right)(q[2]q[0])$
**Inverse:**
$q[1]q[2] \leftarrow Swap\,(q[1]q[2])$
$q[1]q[2]q[3] \leftarrow Inverse\_PE\,(q[1]q[2]q[3])$
**Measurement:**
Applied the $Z$ measurement on qubit $q[0]$.
**if** $q[0] = 1$ **then**
**return** $|x\rangle \leftarrow q[3]$.
**end if**

#### 3.3.2. Experimental results on IBM Q

The experimental results are shown in Table 2. Three types of results are provided. The first one, $x^1_{exp}$, is the theoretical outcome of the quantum circuit, i.e. applying the unitary operators directly on the 4-qubit wave function. The second result $x^2_{exp}$ is given by the classical simulator using 1024 shots for measurement. The third result $x^3_{exp}$ is an actual run on the ibmqx2 (a quantum processor) with 8192 shots. As the result of the classical simulator is based on a quarter of the shots used for the experiments on ibmqx2; therefore, some of $f_2$s (fidelities of classical simulations) are lesser than $f_3$s (fidelities of ibmqx2). If we increase the classical simulations up to 8192 shots, all $f_2$s are essentially 1. The high fidelities are due to small angles of controlled rotations, but an adversarial effect is small success probability upon measurements.

**Table 2**
Experimental results of quantum circuit for solving HHL. $x_{theory} = A^{-1}b$ is the theoretical solution. $x^1_{exp}$ is the theoretical outcome shown in visualizations in terms of statevector on IBM, $x^2_{exp}$ is the experimental outcome of IBM Q system using *ibmq_qasm_simulator* with 1024 shots, and $x^3_{exp}$ is the experimental outcome of IBM Q system using *ibmqx2* with 8192 shots. The fidelities $f_1$ ($f_2$ or $f_3$) are computed by the inner product of normalizations of $x_{theory}$ and $x^1_{exp}$ ($x^2_{exp}$ or $x^3_{exp}$), respectively.

| No. | $\theta(rad)$ | $b$ | $x_{theory}$ | $x^1_{exp}$ | $x^2_{exp}$ | $x^3_{exp}$ | $f_1$ | $f_2$ | $f_3$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | $\pi$ | $\begin{pmatrix}0\\1\end{pmatrix}$ | $\begin{pmatrix}-0.250\\0.750\end{pmatrix}$ | $\begin{pmatrix}-0.024\\0.074\end{pmatrix}$ | $\begin{pmatrix}-\sqrt{0.098\%}\\\sqrt{0.586\%}\end{pmatrix}$ | $\begin{pmatrix}-\sqrt{0.977\%}\\\sqrt{5.53\%}\end{pmatrix}$ | 1.0000 | 0.9978 | 0.9971 |
| 2 | $\pi/2$ | $\begin{pmatrix}0.707\\0.707\end{pmatrix}$ | $\begin{pmatrix}0.354\\0.354\end{pmatrix}$ | $\begin{pmatrix}0.035\\0.035\end{pmatrix}$ | $\begin{pmatrix}\sqrt{0.293\%}\\\sqrt{0.098\%}\end{pmatrix}$ | $\begin{pmatrix}\sqrt{1.172\%}\\\sqrt{1.233\%}\end{pmatrix}$ | 1.0000 | 0.9661 | 0.9999 |
| 3 | -2.23 | $\begin{pmatrix}0.440\\-0.898\end{pmatrix}$ | $\begin{pmatrix}0.555\\-0.784\end{pmatrix}$ | $\begin{pmatrix}0.054\\-0.077\end{pmatrix}$ | $\begin{pmatrix}\sqrt{0.098\%}\\-\sqrt{0.684\%}\end{pmatrix}$ | $\begin{pmatrix}\sqrt{0.94\%}\\-\sqrt{1.184\%}\end{pmatrix}$ | 1.0000 | 0.9679 | 0.9938 |
| 4 | -0.57 | $\begin{pmatrix}0.960\\-0.281\end{pmatrix}$ | $\begin{pmatrix}0.790\\-0.451\end{pmatrix}$ | $\begin{pmatrix}0.077\\-0.044\end{pmatrix}$ | $\begin{pmatrix}\sqrt{0.781\%}\\-\sqrt{0.098\%}\end{pmatrix}$ | $\begin{pmatrix}\sqrt{1.77\%}\\-\sqrt{1.648\%}\end{pmatrix}$ | 1.0000 | 0.9842 | 0.9692 |

## 4. HHL-based algorithms in QML

In this section, we review several QML algorithms that utilize HHL as an important subroutine or inspired by HHL. A summary of these HHL-based QML algorithms is given in Table 3. In particular, we classify these algorithms into five categories based on their applications. In the remaining part of this section, we describe how HHL is used in these different contexts. For simplicity, we skip the discussion of qHop [61] as the way HHL is used is similar to the QSVM algorithm that we thoroughly present in the section 4.1.

### 4.1. Quantum support vector machines

Support vector machine (SVM) is a well-established supervised machine learning algorithm, which is used to classify a new data into one of two classes. Given a training data set $D = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)\}$ with $\vec{x}_j \in \mathbb{R}^d$, and labels $y_j \in \{+1, -1\}_{j=1\ldots m}$. The task of SVM is to find an optimal separating hyperplane $\vec{w}\cdot\vec{x}+b$ between the two classes of data such that the margins from the data to the hyperplane are maximized. The schematic of SVM is shown in Fig. 5.

Training a SVM is to solve a dual optimization problem [62]:

$$\begin{aligned} &\underset{\alpha}{\text{argmax}} \quad L(\vec{a}) = \sum_{j=1}^{m} y_j\alpha_j - \frac{1}{2}\sum_{j,k=1}^{m}\alpha_j K_{jk}\alpha_k, \\ &s.t. \quad \sum_{j=1}^{m}\alpha_j = 0, \quad y_j\alpha_j \geq 0, \end{aligned} \tag{21}$$

where $K_{jk} = k(\vec{x}_j, \vec{x}_k) = \vec{x}_j \cdot \vec{x}_k$ is the linear kernel function $k(x, x')$.

And classify a new data $\vec{x}$ is to compute:

$$y(\vec{x}) = \text{sgn}\left(\sum_{j=1}^{m}\alpha_j k(\vec{x}_j, \vec{x}) + b\right). \tag{22}$$

A least-squares formulation of the SVM re-express the original quadratic programming (21) in terms of a linear programming (23). Since the linear programming can be solved as a matrix inversion problem, one may use the HHL algorithm to benefit from a quantum speedup,

$$F\begin{pmatrix}b\\\vec{\alpha}\end{pmatrix} \equiv \begin{pmatrix}0 & \vec{1}^T\\\vec{1} & K+\gamma^{-1}I\end{pmatrix}\begin{pmatrix}b\\\vec{\alpha}\end{pmatrix} = \begin{pmatrix}0\\\vec{y}\end{pmatrix}. \tag{23}$$

As the matrix $F$ is not sparse, Theorem 3 is needed in this case.

To train a QSVM is actually to solve the matrix inversion problem $\left(b, \vec{\alpha}^T\right)^T = F^{-1}\left(0, \vec{y}^T\right)^T$ by implementing the same algorithmic procedure of HHL in 3.2:

**Table 3**
Summary of HHL-based QML algorithms. Here $N$ and $M$ respectively represents the number and the dimension of data points involved in each algorithm, $\epsilon$ is the error of output, $\kappa_\alpha = \frac{\max\{1,\sqrt{\alpha}\}}{\min\{1/\kappa,\sqrt{\alpha}\}}$ according to the Ref. [38], $c$ denotes the number of classes addressed in QLDA [14], $k$ denotes the rank of a good approximation of the preference data matrix in QRS [13], and $\mu$ is a user-defined number as the trade-off between the run time and the error in calculating the pseudoinverse in qHop algorithm [61].

| ML task | QML algorithm | Complexity |
|---|---|---|
| Data classification | QSVM [12] | $\tilde{O}(\log(NM)\kappa^2/\epsilon^3)$ |
| | QLDA [14] | $O(c\log(NM)\kappa^3/\epsilon^3)$ |
| Linear regression | QOLR [8] | $\tilde{O}(\log(N)s^3\kappa^6/\epsilon)$ |
| | QOLR [38] | $O(\log(N+M)s^2\kappa^3/\epsilon^2)$ |
| | QOLRP [39] | $O(\log(N)\kappa^2/\epsilon^3)$ |
| | QRR [38] | $O(\log(N)s^2\kappa_\alpha^3/\epsilon^2)$ |
| | QRR [40] | $O(\text{polylog}(N+M)\kappa^5/\epsilon^4)$ |
| | QRRCDM [41] | $O(\text{polylog}(N)\kappa^2/\epsilon)$ |
| Recommendation system | QRS [13] | $O(\text{poly}(k)\,\text{polylog}(NM))$ |
| Singular value thresholding | QSVT [45] | $O(\log(NM)/\epsilon)$ |
| Hopfield neural network | qHop [61] | $O(\text{poly}(N, \log(M), 1/\epsilon, 1/\mu))$ |

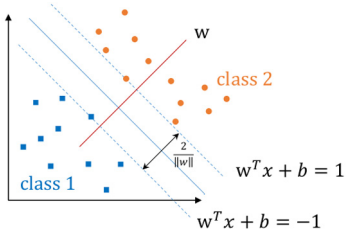


**Fig. 5.** A simple illustrative example of SVM.

$$\begin{aligned} |y\rangle|0\rangle|0\rangle &= \sum_{j=1}^{m+1}\langle u_j|y\rangle|u_j\rangle|0\rangle|0\rangle \\ &\xmapsto{\text{PE}} \sum_{j=1}^{m+1}\langle u_j|y\rangle|u_j\rangle\left|\tilde{\lambda}_j\right\rangle|0\rangle \\ &\xmapsto{\text{CR}} \sum_{j=1}^{m+1}\langle u_j|y\rangle|u_j\rangle\left|\tilde{\lambda}_j\right\rangle|0\rangle\left(\sqrt{1-\frac{\gamma^2}{\tilde{\lambda}_j^2}}|0\rangle + \frac{\gamma}{\tilde{\lambda}_j}|1\rangle\right) \\ &\xmapsto{\text{Inverse}} \sum_{j=1}^{m+1}\langle u_j|y\rangle|u_j\rangle\left(\sqrt{1-\frac{\gamma^2}{\tilde{\lambda}_j^2}}|0\rangle + \frac{\gamma}{\tilde{\lambda}_j}|1\rangle\right) \\ &\xrightarrow[result=1]{Measure} \sum_{j=1}^{m+1}\frac{\langle u_j|y\rangle}{\tilde{\lambda}_j}|u_j\rangle \end{aligned} \tag{24}$$

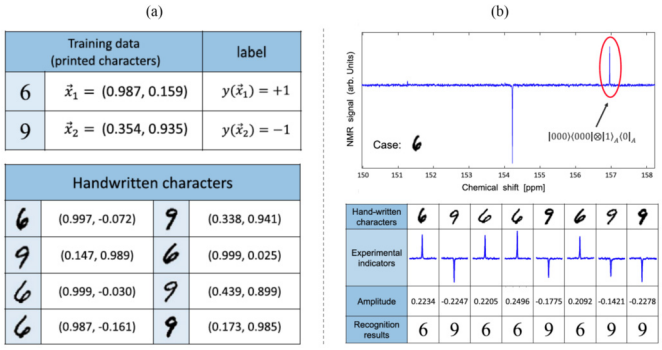


**Fig. 6.** The experiment data of QSVM for recognizing handwritten characters 6 and 9.

In the basis of training set labels, the desired SVM parameters are encoded in the amplitudes of the final state:

$$|b, \vec{\alpha}\rangle = \frac{1}{\sqrt{C}} \left( b|0\rangle + \sum_{k=1}^{m} \alpha_k |k\rangle \right), \tag{25}$$

where $C = b^2 + \sum_{k=1}^{m} \alpha_k^2$.

**Theorem 4** *(Time complexity of QSVM [12]). Given $m$ training data of the form $\{(\vec{x}_j, y_j) : \vec{x}_j \in \mathbb{R}^d, y_j = \pm 1\}$ and a kernel $K$. Let the condition number of $F$ in (23) is $\kappa$. Then there exists a quantum algorithm that solves the training problem of the least-square support vector machine within accuracy $\varepsilon$ in time $\tilde{O}\left(\log(md)\kappa^2/\varepsilon^3\right)$.*

Furthermore, the parameters in (25) could be used for classifying new data. Construct $|u\rangle$ by calling the training-data oracle [12] and construct the query state $|x\rangle$:

$$\begin{aligned} |u\rangle &= \frac{1}{\sqrt{N_u}} \left( b|0\rangle|0\rangle + \sum_{k=1}^{m} \alpha_k |\vec{x}_k| |k\rangle |\vec{x}_k\rangle \right) \\ |x\rangle &= \frac{1}{\sqrt{N_x}} \left( |0\rangle|0\rangle + \sum_{k=1}^{m} |\vec{x}| |k\rangle |\vec{x}\rangle \right) \end{aligned} \tag{26}$$

where $N_u = b^2 + \sum_{k=1}^{m} \alpha_k^2 |\vec{x}_k|^2$ and $N_x = M|\vec{x}|^2 + 1$.

Then perform a quantum swap-test algorithm [63]. Add an ancilla to construct the state

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left( |0\rangle|u\rangle + |1\rangle|x\rangle \right), \tag{27}$$

and measure the ancilla in $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ with success probability $P = |\langle\psi|-\rangle|^2 = \frac{1}{2}(1 - \langle u|x\rangle)$, where the inner product $\langle u|x\rangle = \frac{1}{\sqrt{N_x N_u}} \left( b + \sum_{k=1}^{m} \alpha_k |\vec{x}_k| |\vec{x}| \langle x_k|x\rangle \right)$ corresponds to the classifying result in (22).

If $P < \frac{1}{2}$, the new data $|x\rangle$ is classified to class $y = +1$; otherwise, $y = -1$.

Experiment of QSVM for handwriting recognition on a four-qubit NMR test have been implemented [64]. The QSVM algorithm was used to solve a minimal optical character recognition problem. Particularly, QSVM was trained with the standard fonts of the characters 6 and 9, and then utilized to classify a new-coming handwritten character. The encoded training and test data are shown in Fig. 6(a), and the classification results on the NMR platform are shown in Fig. 6(b). This work has demonstrated the feasibility of executing QSVM in a near-term quantum computer. However, further works to improve quantum hardware and error corrections are needed to execute a large-scale experiment.

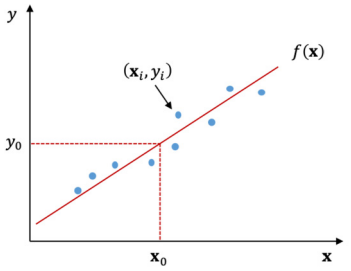


**Fig. 7.** A simple illustrative example of LR.

An improved implementation of QSVM for NISQ (noisy intermediate scale quantum) computers are proposed, leading to better classification results for both OCR and Iris datasets on IBMQX2 [65]. This NISQ-adapted strategy overcomes the drawbacks of [64]. Specifically, the proposed QSVM system extends the specific OCR classification to any two-dimensional datasets that are linearly separably; and the quantum circuit depth is reduced by introducing a new training-data oracle and a new HHL quantum circuit.

### 4.2. Quantum linear regression

Linear regression (LR) is a well-known machine learning task with wide applications in the fields of biology, behavioristic, sociology, finance, and so on [66]. Given $N$ data points $(\mathbf{x}_i, y_i)_{i=1}^{N}$, where $\mathbf{x}_i = (x_{i1}, \cdots, x_{iM})^{\mathrm{T}} \in \mathbb{R}^M$ are $M$-dimensional column vectors as input and $y_i \in \mathbb{R}$ are scalars as output, LR assumes that $\mathbf{x}_i$ and $y_i$ are linearly correlated and aims to construct a linear function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ characterized by the parameters $\mathbf{w} = (w_1, \cdots, w_M)^T$ that can best fit such relationship, i.e., making every $f(\mathbf{x}_i)$ as close as possible to $y_i$. It should be emphasized that the inputs $\mathbf{x}_i$ can be generated by a nonlinear map on the original data set, such as polynomial function, enabling nonlinear regression. After obtaining $f(\mathbf{x})$ by training the given set of $N$ data points, one can predict the output $y_0 = f(\mathbf{x}_0)$ of a new input data $\mathbf{x}_0$. A simple illustrative example of LR is given in Fig. 7.

In 2012, built upon HHL algorithm, Wiebe et al. developed a quantum ordinary linear regression (QOLR) algorithm [8] for the simplest LR model—ordinary linear regression (OLR), whose optimal parameters $\mathbf{w}$ is attained via least squares [66], that is,

$$\begin{aligned} \mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} |f(\mathbf{x}_i) - y_i|^2 \\ &= (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \end{aligned} \tag{28}$$

where $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)^{\mathrm{T}}$ is called *data matrix* and $\mathbf{y} = (y_1, \cdots, y_N)^{\mathrm{T}}$. To finally obtain a quantum state encoding the desired optimal parameter vector $\mathbf{w}$ in its amplitudes, the algorithm implements a two-phase procedure. First, perform $\mathbf{I}(\mathbf{X}^{\mathrm{T}})$ on the prepared quantum state $|1, \mathbf{y}\rangle = \sum_{i=M+1}^{N+M} y_i|i\rangle / |\mathbf{y}|$ following the same algorithmic procedure of HHL algorithm, except that implementing the eigenvalue itself instead of its inverse is done in the controlled rotation part. In the second phase, perform $\mathbf{I}(\mathbf{X})^{-2}$ on the resulting state of the last phase, denoted by $|1, \mathbf{X}^{\mathrm{T}}\mathbf{y}\rangle$, following HHL algorithm too, except that the controlled rotation part implements squaring the eigenvalue rather than inverting it. Then we obtain the final $(N + M)$-dimensional quantum state $|0, \mathbf{w}\rangle = \sum_{i=1}^{M} w_i|i\rangle / |\mathbf{w}|$ which encodes the desired optimal parameter vector $\mathbf{w}$ in its first $M$ amplitudes. The overall process can be summarized by the flow chart shown below:

$$|1, \mathbf{y}\rangle \overset{\mathbf{I}(\mathbf{X}^{\mathrm{T}})}{\longmapsto} |1, \mathbf{X}^{\mathrm{T}}\mathbf{y}\rangle \overset{\mathbf{I}(\mathbf{X})^{-2}}{\longmapsto} |0, \mathbf{w}\rangle.$$

The QOLR algorithm takes time $\tilde{O}(\log(N)s^3\kappa^6/\epsilon)$, where $s$ and $\kappa$ is the sparsity and the condition number of $\mathbf{X}$, respectively, and $\epsilon$ is the error of generating $|0,\mathbf{w}\rangle$. Later, this algorithm was simplified by just performing $\mathbf{I}(X)^{-1}$ on the prepared $(N+M)$-dimensional state $|0,\mathbf{y}\rangle = \sum_{i=1}^{N} y_i|i\rangle/|\mathbf{y}|$ via HHL algorithm, resulting in the desired quantum state $|1,\mathbf{w}\rangle = \sum_{i=1}^{M} w_i|N+i\rangle$ encoding the optimal $\mathbf{w}$. The simplicity leads to an improved time complexity $O(\log(N+M)s^2\kappa^3/\epsilon^2)$, achieving better dependence on $s$ and $\kappa$, but slightly worse dependence on $\epsilon$. It should be noted that after generating the state $|0,\mathbf{w}\rangle$ (or $|1,\mathbf{w}\rangle$), one can use it to predict the output $f(\mathbf{x}_0) = \mathbf{w}^T\mathbf{x}_0$ of a new input data $\mathbf{x}_0$ encoded in a $(N+M)$-dimensional quantum state $|0,\mathbf{x}\rangle = \sum_{i=1}^{M} \mathbf{x}_{0i}|i\rangle/|\mathbf{x}_0|$ (or $|1,\mathbf{x}\rangle = \sum_{i=1}^{M} \mathbf{x}_{0i}|i+N\rangle/|\mathbf{x}_0|$) by estimating the product of these two states via swap test [63,12,39].

To make both quantum algorithms above highly efficient, the data matrices are required to be sparse, i.e., $s = \text{polylog}(N,M)$. In 2016, utilizing the HHL trick [6] together with the density matrix exponentiation technique [43], Schuld et al. presented a quantum OLR-based algorithm [39] for prediction (QOLRP). Different from the previous two quantum LR algorithms, the algorithm can efficiently handle nonsparse data matrix as long as it is approximately low-rank.

Ridge regression is another type of LR, which generalizes OLR by introducing a fraction of regularization of $\mathbf{w}$ to OLR to circumvent multicollinearity of data, leading to the optimal $\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \alpha\mathbb{I})^{-1}\mathbf{X}^T\mathbf{y}$, where $\alpha$ denotes the fraction of regularization. The first quantum ridge regression (QRR) algorithm [38] was developed by Liu and Zhang, which aims to output the quantum state encoding the optimal $\mathbf{w}$ by directly applying the HHL trick. The algorithm is efficient only when the data matrices are sparse. Recently, Yu et al. gave another QRR algorithm [40] that can efficiently handle nonsparse data matrices. More importantly, also using the HHL trick, the algorithm developed the technique of quantum $K$-fold cross validation that is the quantum version of $K$-fold cross validation and can efficiently choose a suitable $\alpha$ for QRR. In the same year, Yu et al. also presented a HHL-based QRR algorithm [41] for circulant data matrices (QRRCDM), and showed how to apply it to implement the tasks related to visual tracking, a fundamental problem in the field of computer vision.

### 4.3. Quantum recommendation systems

A recommendation system is primarily used in commercial applications seeking to predict items that users may have an interest in. One example is the playlist generators for video services like Netflix: given a bunch of users and each one expresses their preferences for a subset of movies, the system then generates a list of recommended movies based on their past preferences.

The questions discussed above can be modeled by a preference matrix $P \in \mathbb{R}^{m\times n}$ with $m$ users and $n$ products, where $P_{ij}$ denotes how much the user $i$ values the product $j$. And most entries of $P$ are unknown. By given $P$, recommendation systems aims to suggest product $j'$ to the individual user $i'$ given the predicted values of $P_{ij}$. One powerful method to address this challenge is the matrix reconstruction. By setting the unknown entires of $P$ to be 0, and by doing SVD we can reconstruct the preference matrix using the low-rank approximation.

Quantum recommendation system (QRS) aims to solve the same challenge. The basic idea of QRS is using quantum algorithms to perform matrix sampling, i.e. a sampling of items is done for a user with the anticipation that he or she will like the item with high probability. Compared to the approached mentioned above, matrix sampling does not require to reconstruct the whole matrix, and the time complexity of QRS is shown in Theorem 5.

**Theorem 5** *(Time complexity of QRS [13]). There exists a quantum recommendation algorithm with running time* $O\left(\text{poly}(k)\text{poly}\log(mn)\right)$.

The main tool is quantum singular value estimation (QSVE). Let the SVD of the matrix $A$ is $A = \sum_i \sigma_i u_i v_i^T$, with singular value $\sigma_i$ and singular vectors $u_j$ and $v_i$. And the rank-$k$ approximation $A_k = \sum_{i\in[k]} \sigma_i u_i v_i^T$ minimizes $||A - A_k||_F$. The cost of QSVE is depicted in Theorem 6. It is worth noting that QSVE can be adopted to solve the linear system of equations with general dense matrices [67].

**Theorem 6** *(Time complexity of QSVE [13]). There exists an algorithm with running time* $O\left(\text{polylog}(mn)/\varepsilon\right)$ *that transforms* $\sum_i \alpha_i|v_i\rangle \to \sum_i \alpha_i|v_i\rangle|\tilde{\sigma}_i\rangle$ *where* $||\tilde{\sigma}_i - \sigma_i|| \le \varepsilon||A||_F$, *with probability at least* $1 - 1/\text{poly}(n)$.

While the quantum algorithm for solving the recommendation system is inspired by HHL, the actual approach to attacking the underlying linear-algebra problems are substantially distinct. More details may be found in [13].

Recently, a quantum-inspired classical algorithm for recommendation systems is proposed with time complexity $O\left(\text{poly}(k)\log(mn)\right)$ for a given $m \times n$ matrix with a rank-$k$ approximation, which is only polynomially slower than the quantum algorithm [68], which constitutes a breakthrough in shortening the boundary of time complexity between classical and quantum algorithms.

Successively, [69] showed that the proven time complexity bounds of this quantum-inspired classical algorithm do not actually reflect its practical runtimes. The results of [69] indicate that this method can perform well in practice only under the restrictive conditions of large-dimensional input matrices with very low rank and low condition number. By contrast, quantum algorithm can efficiently solve problems with sparse but full-rank matrices having low condition number. Dealing with matrices with large condition number is still a challenge for both quantum and quantum-inspired classical algorithm.

### 4.4. Quantum singular value thresholding

Singular value thresholding (SVT) is a core module in many mathematical models in computer vision and machine learning. SVT is typically used to solve nuclear norm minimizing (NNM) problems, so it can be applied in image extraction, image denoising, etc.

**Definition 2** *(Singular value thresholding [45,70]).* Given a low-rank matrix $Y \in \mathbb{R}^{n_1\times n_2}$ with truncated singular value decomposition $Y = U\Sigma V^T$, $\Sigma = diag(\sigma_1,\cdots,\sigma_r)$ with rank $r \ll \min\{n_1,n_2\}$, $U \in \mathbb{R}^{n_1\times r}$, $V \in \mathbb{R}^{n_2\times r}$. Let $\tau \ge 0$, then $\mathcal{D}_\tau(Y) = U\mathcal{D}_\tau(\Sigma)V^T$ is the singular value thresholding (SVT) operator, where $\mathcal{D}_\tau(\Sigma) = diag\left((\sigma_1-\tau)_+,\cdots,(\sigma_r-\tau)_+\right)$, and $(\sigma_i-\tau)_+ = \sigma_i - \tau$ if $\sigma_i > \tau$, and 0 else.

QSVT is an analogous quantum algorithm to SVT, which solves the problem $|x\rangle = \mathcal{D}_\tau|b\rangle$ with the normalization factors omitted for simplification. Specifically, the original data is a matrix $A_0 \in \mathbb{R}^{p\times q}$ with singular value decomposition (SVD) $A_0 = \sum_j \sigma_j u_j v_j^T$, where $\sigma_j$ are the singular values, and $u_j$ and $v_j$ are the singular vectors. Both inputs $|b\rangle$ and $A$ come from $A_0 \in \mathbb{R}^{p\times q}$, where $|b\rangle = \text{vec}(A_0) = \sum_j \sigma_j|u_j\rangle|v_j\rangle \in \mathbb{R}^{pq}$, and $A$ comes from either $A_0A_0^\dagger$ or $A_0^\dagger A_0$. Therefore, the output of QSVT is a state $|\tilde{x}\rangle \approx \sum_j (\sigma_j-\tau)_+|u_j\rangle|v_j\rangle$.

As mentioned in section 2.4.2, the dimension of $A$ is smaller than $|b\rangle$ in this case. For $p < q$, $A$ is chosen as $A = A_0A_0^\dagger =$

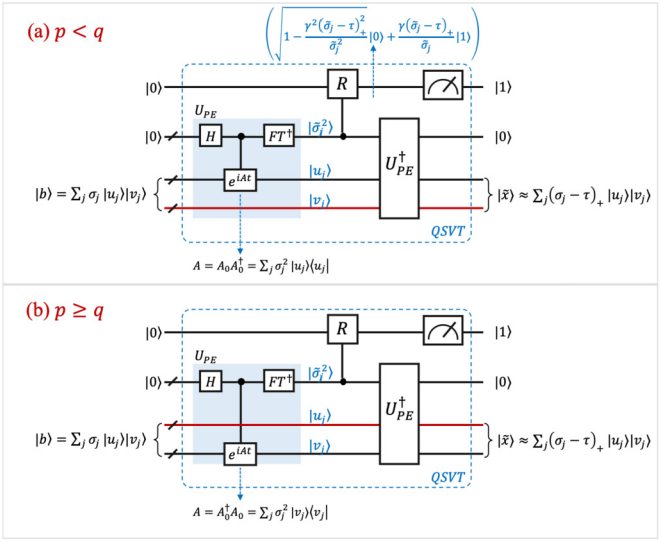


**Fig. 8.** The quantum circuit of QSVT.

$\sum_j \sigma_j^2 |u_j\rangle\langle u_j| \in \mathbb{R}^{p\times p}$, and for $p \geq q$, $A = A_0^\dagger A_0 = \sum_j \sigma_j^2 |v_j\rangle\langle v_j| \in \mathbb{R}^{q\times q}$. It is interesting that the controlled unitary $e^{iAt}$ can be just applied to part of $|b\rangle$ (first $p$ dimension in Fig. 8(a) or last $q$ dimension in Fig. 8(b)), namely no operation is applied on the red line in either case. This algorithm indicates that alternative methods to construct $A$ can be used to ensure $e^{iAt}$ may be applied to a subsystem with lower dimension.

As depicted in Fig. 8, the procedure of the QSVT algorithm is similar to HHL. See more detailed of this algorithm in [45]. The following Theorem 7 gives the time complexity of QSVT.

**Theorem 7** *(Time complexity of QSVT [43]). Let $A_0 \in \mathbb{R}^{p\times q}$ be the input of the QSVT problem. Without loss of generality, let $p < q$. Assume that $A = A_0 A_0^\dagger$ is low-rank and well-conditioned. Given an thresholding $\tau$, then there exists a quantum algorithm that outputs a quantum state proportional to $\mathcal{D}_\tau(\mathrm{vec}(A_0))$ within accuracy $\varepsilon$ in time $O\left(t^2\varepsilon^{-1}\log(pq)\right)$, where $\mathcal{D}_\tau$ is SVT operator with thresholding $\tau$.*

Note that, QSVT has a special property that most HHL-based algorithms do not have. That is, by choosing a proper angle of the controlled rotation (which can be computed in terms of the largest singular value of $A_0$), a single iteration of the QSVT could output a desired result with both high fidelity and high success probability. This trick helps avoid extra circuits involved in [14] and also avoid amplitude amplification to boost the success probability as in most HHL-based algorithms.

## 5. Challenges and outlook

We have thoroughly reviewed the standard HHL algorithm and its usages in the context of quantum machine learning. At the end, we echo the point of view raised by Aaronson [49] that HHL should be viewed as a template for other quantum algorithms. For any specific problem, one should analyze whether the state $|b\rangle$ can be efficiently prepared, whether the Hamiltonian simulation of $e^{-iAt}$ can be efficiently executed, and how the information contained in the state $|x\rangle$ should be extracted. There are at least one caveat [49] associated with each of the three tasks mentioned above. Continuous algorithmic innovations, such as novel Hamiltonian simulation techniques [36] including approaches [71,72] that may completely avoid the phase estimation as well as variable-time amplitude amplifications [73], directly improve the performance of the original HHL algorithm. Future algorithmic innovations should further optimize the circuit size and depth for problems of practical interests. For QML problems, there is an additional challenge to efficiently port the classical data into the quantum domain. The QRAM, while being a promising proposal, is still far from being a matured technology to routinely prepare big data for QML.

While HHL is the first algorithm that starts the trend of developing quantum algorithms for linear-algebraic applications, there exists other methods which worth further attentions. For instance, the block-encoding method [74] is an alternative approach that can replace HHL for many core computational tasks in QML. Similarly, novel method proposed in the work of quantum recommendation system [13] is another interesting option that clearly shows that many more competitive options other than HHL exist for QML. It will be crucial to further explore alternative schemes beyond HHL for solving linear algebra and developing related QML algorithms.

In this Noisy Intermediate-Scale Quantum (NISQ) era [75], we work with quantum devices without quantum error corrections. The hybrid quantum-classical (HQC) algorithms [76] have emerged as a popular paradigm to solve challenging computational problems. Not surprisingly, novel HQC algorithms [77,78] have been proposed to address the QLSP (that HHL was designed for); hence these algorithms bear relevance to boosting performance of machine learning models in the near term. Although, we caution these HQC approaches are typically not related to the original HHL algorithm other than the fact that they target the same QLSP. Besides the efforts to apply QML to process classical data in the near term, it is also extremely beneficial to investigate novel applications that utilize quantum machine learning to interact with quantum data [79]. Some interesting examples include a QML model to characterize and control other quantum hardwares.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This work was supported by the Funding of National Natural Science Foundation of China (Grants No. 61901218 and No. 61701229), Natural Science Foundation of Jiangsu Province, China (Grants No. BK20170802 and No. BK20190407), China Postdoctoral Science Foundation funded Project (Grants No. 2018M630557 and No. 2018T110499), Jiangsu Planned Projects for Postdoctoral Research Funds (Grant No. 1701139B). The authors also acknowledge Michele Mosca, Hailing Liu for inspiring discussions.

## Appendix A. Encoding classical data into analog-encoded states

One approach of encoding classical data into analog-encoded states in [16] is shown in Fig. A.9. The aim is to generate a quantum state $|x\rangle = \frac{1}{||x||}\sum_j x_j|j\rangle$, where $x \in \mathbb{R}^N$ denotes a classical data. Let $\{x_j\}_{j=1}^N$ be a set of real numbers in [0,1), each of which may be represented by $x_j = \sum_{k=1}^m x_j^{(k)} 2^{-k}$ with binary variables $x_j^{(k)} \in \{0,1\}$. In this appendix, $|x_j\rangle \equiv |x_j^{(1)}\cdots x_j^{(m)}\rangle$ implies the $m$-bit binary representation of the real number $x_j$. An $m$-bit QDAC operation transforms digital-encoded state $\frac{1}{\sqrt{N}}\sum_{j=1}^N |j\rangle|x_j\rangle$ to $C\sum_j x_j|j\rangle|0\rangle^{\otimes m}$, where $C$ is the normalization factor.

The basic idea of QDAC is to compute an angle $\varphi_j = \frac{2}{\pi} cos^{-1} x_j$ by quantum arithmetics, and implement a sequence of controlled rotation $R_y$ with $\varphi_j$, then measure the ancilla and uncompute $\varphi_j$ and $U_D^\dagger$ to get the analog-encoded state:

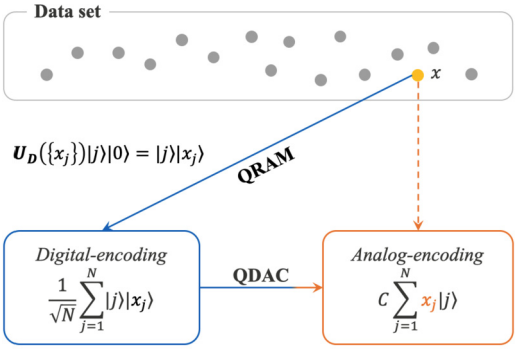


**Fig. A.9.** Schematic of loading classical data to quantum states. $U_D$ is often deemed as QRAM. QDAC refers to quantum digital-to-analog conversion.

$$\begin{aligned}
&\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle|x_j\rangle|0\rangle^{\otimes m}|0\rangle_a \\
&\mapsto \frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle|x_j\rangle|\varphi_j\rangle|0\rangle_a \\
&\mapsto \frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle|x_j\rangle|\varphi_j\rangle\left(x_j|0\rangle_a+\sqrt{1-x_j^2}|1\rangle_a\right) \\
&\mapsto C\sum_{j=1}^{N}x_j|j\rangle
\end{aligned} \tag{A.1}$$

Theorem 8 in Appendix A indicts that the complexity of encoding classical data into the amplitude of states is determined by the complexity of QRAM.

**Theorem 8** *(QDAC [16]). There exists a quantum algorithm that performs QDAC using* $m = O\left(\log\epsilon^{-1}\right)$ *qubits and* $O\left(poly\left(\log\epsilon^{-1}\right)\right)$ *single- and two-qubit gates and one* $U_D^{\dagger}$*, with probability* $\sum_{j=1}^{2}x_j{}^2/N$.

And a variant of this method is to compute out all the controlled angles $\varphi_j = \frac{2}{\pi}cos^{-1}x_j$ classically, which may omit the quantum arithmetics procedure in (A.1).

## Appendix B. Lemma and theorem for Hamiltonian simulation

**Lemma 9** *(LCU Lemma [33]). Given the ability to perform unitaries* $V_j$ *with unit complexity, one can perform the operation* $U = \sum_j \beta_j V_j$ *with complexity* $O\left(\sum_j|\beta_j|\right)$*. Furthermore, if U is (nearly) unitary then this implementation can be made (nearly) deterministic.*

**Theorem 10** *(Spectral theorem [31]). Each eigenvalue* $\lambda$ *of H corresponds to two eigenvalues* $\pm e^{\pm i\arcsin\lambda}$ *of the walk operator (with eigenvectors closely related to those of H).*

## Appendix C. IBM Q experiment

As mentioned in Algorithm 2, phase estimation includes two $H$ gates and controlled unitarities and inverse QFT. Here, we set $t = 2\pi$, apply c-$U_3\left(-\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2}\right)$ and $U_1\left(\frac{3\pi}{4}\right)$ to achieve c-$e^{iAt/4}$, and applied a CNOT to achieve c-$e^{iAt/2}$, where $A = \begin{bmatrix}1.5 & 0.5\\ 0.5 & 1.5\end{bmatrix}$, the equivalence are shown as follows.

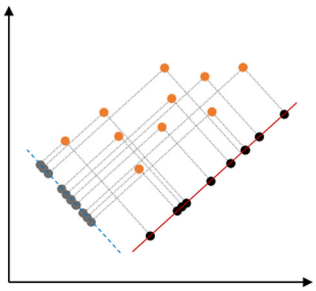

**Fig. D.10.** A simple illustrative example of PCA.

### C.1. The equivalence of controlled $e^{iAt}$

Obviously, in the case $t = 2\pi$, $e^{iAt/2}$ equals to $X$ gate. And $e^{iAt/4}$ can be decomposed as follows:

$$e^{iA\frac{t}{4}} = e^{-i\left(-\frac{3\pi}{2}\right)\frac{I}{2}}\cdot e^{-i\left(-\frac{\pi}{2}\right)\frac{\sigma}{2}} = R_I\left(-\frac{3\pi}{2}\right)\cdot R_X\left(-\frac{\pi}{2}\right) \tag{C.1}$$

According to eq. (20), we have $R_X\left(-\frac{\pi}{2}\right) = U_3\left(-\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2}\right)$, and c-$R_I\left(-\frac{3\pi}{2}\right) = U_1\left(\frac{3\pi}{4}\right)$.

## Appendix D. Quantum principal component analysis

From the perspective of algorithm design, QPCA is not strictly a generalization based on HHL because it only implements a phase estimation and does not include a controlled rotation stage. However, one may still argue this algorithm is largely inspired by HHL. We briefly introduce this algorithm in this appendix. QPCA provides a way to exponentiate a non-sparse but low-rank matrix (i.e. making a related unitary operation), which proves to be useful under many circumstances. For instance, QPCA plays a role in QML algorithms for clustering and pattern recognitions [80].

Principal component analysis (PCA) is a tool in data analysis for revealing the internal structure of the data by projecting the data onto the directions of maximal variance. It is often used for dimensionality reduction in terms of focusing on the overall data variance. The schematic of PCA is shown in Fig. D.10.

Given a data set $X = (x_1,\cdots,x_n), x_i\in\mathbb{R}^d, (i = 1,\cdots,n)$ and $\sum_{i=0}^{n}x_i = 0$. PCA aims to find a projector $P$ mapping $X$ to $Y$: $Y = P^TX$, where $Y = (y_1,\cdots,y_n), y_i = p_i^Tx_i\in\mathbb{R}^r, (i = 1,\cdots,n, r < d)$. To solve the optimization problem of PCA

$$\begin{aligned}
&\max_P\ \mathrm{tr}\left(P^TXX^TP\right),\\
&s.t.\ \ P^TP = I,
\end{aligned} \tag{D.1}$$

is actually to solve the eigenvalue decomposition of the data covariance matrix $XX^T$. Using the first few principal components of $XX^T$ to construct the projector $P$, the representation of the data can be compressed and further used for predicting future behavior. Classically, this procedure scale as $O\left(d^2\right)$ in terms of computational complexity and query complexity.

QPCA is a quantum analog of PCA with a slightly different goal. QPCA is more often used for revealing the properties of an unknown quantum density matrix.

**Theorem 11** *(Time complexity of QPCA [43]). Given an unknown non-sparse but low-rank density matrix* $\rho$*. Then there exists a quantum algorithm that uses multiple copies of* $\rho$ *to construct the principle components in time* $O(\log d)$.

Specifically, by firstly applying a density matrix $\rho$ as a Hamiltonian on another density matrix $\sigma$, and using Theorem 3, $e^{-i\rho t}$ can be implemented in time $O(t^2/\varepsilon\log d)$ to accuracy $\varepsilon$.

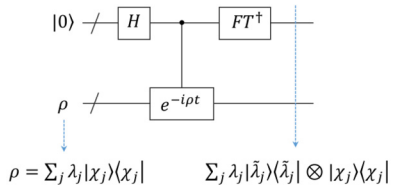


**Fig. D.11.** The quantum circuit of QPCA.

If $\rho$ is dominated by a few large eigenvalues, meaning it is well represented by its principal components, then one can apply $e^{-i\rho t}$ to perform phase estimation to $\rho$ itself, yielding the state:

$$\sum_i \lambda_i \left|\chi_i\right\rangle \left\langle\chi_i\right| \otimes \left|\tilde{\lambda_i}\right\rangle\left\langle\tilde{\lambda_i}\right|, \tag{D.2}$$

where $|\chi_i\rangle$ are the eigenvectors of $\rho$ and $\tilde{\lambda_i}$ are the estimated eigenvalues. A simplified view of QPCA circuit is provided in Fig. D.11.

## References


[1] R.P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. 21 (6) (1982) 467–488.
[2] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35th Annual Symposium on Foundations of Computer Science, IEEE, 1994, pp. 124–134.
[3] L.K. Grover, Quantum mechanics helps in searching for a needle in a haystack, Phys. Rev. Lett. 79 (2) (1997) 325.
[4] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information (10th Anniv. Version), 2010.
[5] G. Brassard, P. Hoyer, M. Mosca, A. Tapp, Quantum amplitude amplification and estimation, Contemp. Math. 305 (2002) 53–74.
[6] A.W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. 103 (15) (2009) 150502.
[7] D.W. Berry, High-order quantum algorithm for solving linear differential equations, J. Phys. A, Math. Theor. 47 (10) (2014) 105301.
[8] N. Wiebe, D. Braun, S. Lloyd, Quantum algorithm for data fitting, Phys. Rev. Lett. 109 (5) (2012) 050505.
[9] A. Ta-Shma, Inverting well conditioned matrices in quantum logspace, in: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, ACM, 2013, pp. 881–890.
[10] B.D. Clader, B.C. Jacobs, C.R. Sprouse, Preconditioned quantum linear system algorithm, Phys. Rev. Lett. 110 (25) (2013) 250504.
[11] G. Wang, Efficient quantum algorithms for analyzing large sparse electrical networks, arXiv preprint, arXiv:1311.1851, 2013.
[12] P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification, Phys. Rev. Lett. 113 (13) (2014) 130503.
[13] I. Kerenidis, A. Prakash, Quantum recommendation systems, arXiv preprint, arXiv:1603.08675, 2016.
[14] I. Cong, L. Duan, Quantum discriminant analysis for dimensionality reduction and classification, New J. Phys. 18 (7) (2016) 073011.
[15] J.A. Cortese, T.M. Braje, Loading classical data into a quantum computer, arXiv preprint, arXiv:1803.01958, 2018.
[16] K. Mitarai, M. Kitagawa, K. Fujii, Quantum analog-digital conversion, Phys. Rev. A 99 (1) (2019) 012301.
[17] A. Lucas, Ising formulations of many NP problems, Front. Phys. 2 (2014) 5.
[18] Z. Bian, F. Chudak, R. Israel, B. Lackey, W.G. Macready, A. Roy, Discrete optimization using quantum annealing on sparse Ising models, Front. Phys. 2 (2014) 56.
[19] L. Grover, T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions, arXiv preprint, arXiv:quant-ph/0208112, 2002.
[20] P. Kaye, M. Mosca, Quantum networks for generating arbitrary quantum states, arXiv preprint, arXiv:quant-ph/0407102, 2004.
[21] V. Giovannetti, S. Lloyd, L. Maccone, Quantum random access memory, Phys. Rev. Lett. 100 (16) (2008) 160501.
[22] V. Giovannetti, S. Lloyd, L. Maccone, Architectures for a quantum random access memory, Phys. Rev. A 78 (5) (2008) 052310.
[23] F.-Y. Hong, Y. Xiang, Z.-Y. Zhu, L.-z. Jiang, L.-n. Wu, Robust quantum random access memory, Phys. Rev. A 86 (1) (2012) 010306.
[24] S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca, P.V. Srinivasan, On the robustness of bucket brigade quantum ram, New J. Phys. 17 (12) (2015) 123010.
[25] O. Di Matteo, V. Gheorghiu, M. Mosca, Fault tolerant resource estimation of quantum random-access memories, arXiv preprint, arXiv:1902.01329, 2019.
[26] D.K. Park, F. Petruccione, J.-K.K. Rhee, Circuit-based quantum random access memory for classical data, Sci. Rep. 9 (1) (2019) 3949.
[27] J. Bang, A. Dutta, S.-W. Lee, J. Kim, Optimal usage of quantum random access memory in quantum machine learning, Phys. Rev. A 99 (1) (2019) 012326.
[28] C.T. Hann, C.-L. Zou, Y. Zhang, Y. Chu, R.J. Schoelkopf, S.M. Girvin, L. Jiang, Hardware-efficient quantum random access memory with hybrid quantum acoustic systems, Phys. Rev. Lett. 123 (25) (2019) 250501.
[29] S. Lloyd, Universal quantum simulators, Science (1996) 1073–1078.
[30] D.W. Berry, G. Ahokas, R. Cleve, B.C. Sanders, Efficient quantum algorithms for simulating sparse Hamiltonians, Commun. Math. Phys. 270 (2) (2007) 359–371.
[31] D.W. Berry, A.M. Childs, Black-box Hamiltonian simulation and unitary implementation, Quantum Inf. Comput. 12 (1&2) (2012) 0029.
[32] D.W. Berry, A.M. Childs, R. Cleve, R. Kothari, R.D. Somma, Exponential improvement in precision for simulating sparse Hamiltonians, in: Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, ACM, 2014, pp. 283–292.
[33] D.W. Berry, A.M. Childs, R. Cleve, R. Kothari, R.D. Somma, Simulating Hamiltonian dynamics with a truncated Taylor series, Phys. Rev. Lett. 114 (9) (2015) 090502.
[34] D.W. Berry, A.M. Childs, R. Kothari, Hamiltonian simulation with nearly optimal dependence on all parameters, in: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, IEEE, 2015, pp. 792–809.
[35] G. Hao Low, I.L. Chuang, Hamiltonian simulation by qubitization, arXiv preprint, arXiv:1610.06546, 2016.
[36] A.M. Childs, D. Maslov, Y. Nam, N.J. Ross, Y. Su, Toward the first quantum simulation with quantum speedup, Proc. Natl. Acad. Sci. 115 (38) (2018) 9456–9461.
[37] P. Kaye, R. Laflamme, M. Mosca, et al., An Introduction to Quantum Computing, Oxford University Press, 2007.
[38] Y. Liu, S. Zhang, Fast quantum algorithms for least squares regression and statistic leverage scores, Theor. Comput. Sci. 657 (2017) 38–47.
[39] M. Schuld, I. Sinayskiy, F. Petruccione, Prediction by linear regression on a quantum computer, Phys. Rev. A 94 (2) (2016) 022342.
[40] C.-H. Yu, F. Gao, Q. Wen, An improved quantum algorithm for ridge regression, IEEE Trans. Knowl. Data Eng. (2019).
[41] C.-H. Yu, F. Gao, C. Liu, D. Huynh, M. Reynolds, J. Wang, Quantum algorithm for visual tracking, Phys. Rev. A 99 (2) (2019) 022301.
[42] P. Rebentrost, A. Steffens, S. Lloyd, Quantum singular value decomposition of non-sparse low-rank matrices, arXiv preprint, arXiv:1607.05404, 2016.
[43] S. Lloyd, M. Mohseni, P. Rebentrost, Quantum principal component analysis, Nat. Phys. 10 (9) (2014) 631.
[44] B. Duan, J. Yuan, Y. Liu, D. Li, Quantum algorithm for support matrix machines, Phys. Rev. A 96 (3) (2017) 032301.
[45] B. Duan, J. Yuan, Y. Liu, D. Li, Efficient quantum circuit for singular-value thresholding, Phys. Rev. A 98 (1) (2018) 012308.
[46] G. Brassard, P. Hoyer, An exact quantum polynomial-time algorithm for Simon's problem, in: Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems, IEEE, 1997, pp. 12–23.
[47] L.K. Grover, Quantum computers can search rapidly by using almost any transformation, Phys. Rev. Lett. 80 (19) (1998) 4329.
[48] D.P. Chi, J. Kim, Quantum database search by a single query, in: NASA International Conference on Quantum Computing and Quantum Communications, Springer, 1998, pp. 148–151.
[49] S. Aaronson, Read the fine print, Nat. Phys. 11 (4) (2015) 291.
[50] A.M. Childs, On the relationship between continuous- and discrete-time quantum walk, arXiv preprint, arXiv:0810.0312, 2008.
[51] D. Aharonov, A. Ta-Shma, Adiabatic quantum state generation and statistical zero knowledge, in: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, ACM, 2003, pp. 20–29.
[52] X.-D. Cai, C. Weedbrook, Z.-E. Su, M.-C. Chen, M. Gu, M.-J. Zhu, L. Li, N.-L. Liu, C.-Y. Lu, J.-W. Pan, Experimental quantum computing to solve systems of linear equations, Phys. Rev. Lett. 110 (23) (2013) 230501.
[53] S. Barz, I. Kassal, M. Ringbauer, Y.O. Lipp, B. Dakic, A. Aspuru-Guzik, P. Walther, Solving systems of linear equations on a quantum computer, arXiv preprint, arXiv:1302.1210, 2013.
[54] J. Pan, Y. Cao, X. Yao, Z. Li, C. Ju, H. Chen, X. Peng, S. Kais, J. Du, Experimental realization of quantum algorithm for solving linear systems of equations, Phys. Rev. A 89 (2) (2014) 022313.
[55] Microsoft cooperation, https://azure.microsoft.com/, 2019. (Accessed 11 May 2019).
[56] Origin quantum cooperation, http://qubitonline.cn/, 2017. (Accessed 1 November 2018).
[57] Qutech cooperation, https://www.quantum-inspire.com/, 2018. (Accessed 9 April 2018).
[58] Amazon cooperation, https://aws.amazon.com/braket/, 2019. (Accessed 12 January 2019).
[59] IBM cooperation, https://quantum-computing.ibm.com/, 2016. (Accessed 12 January 2017).
[60] IBMQ HHL code, https://github.com/dreamingangela/ibmq_hhl/blob/master/hhl.qasm/, 2019. (Accessed 22 December 2019).
[61] P. Rebentrost, T.R. Bromley, C. Weedbrook, S. Lloyd, Quantum Hopfield neural network, Phys. Rev. A 98 (4) (2018) 042308.

[62] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[63] H. Buhrman, R. Cleve, J. Watrous, R. De Wolf, Quantum fingerprinting, Phys. Rev. Lett. 87 (16) (2001) 167902.
[64] Z. Li, X. Liu, N. Xu, J. Du, Experimental realization of a quantum support vector machine, Phys. Rev. Lett. 114 (14) (2015) 140504.
[65] J. Yang, A.J. Awan, G. Vall-Llosera, Support vector machines on noisy intermediate scale quantum computers, arXiv preprint, arXiv:1909.11988, 2019.
[66] K.P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.
[67] L. Wossnig, Z. Zhao, A. Prakash, Quantum linear system algorithm for dense matrices, Phys. Rev. Lett. 120 (5) (2018) 050502.
[68] E. Tang, A quantum-inspired classical algorithm for recommendation systems, in: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, ACM, 2019, pp. 217–228.
[69] J.M. Arrazola, A. Delgado, B.R. Bardhan, S. Lloyd, Quantum-inspired algorithms in practice, arXiv preprint, arXiv:1905.10415, 2019.
[70] J.-F. Cai, E.J. Candès, Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. Optim. 20 (4) (2010) 1956–1982.
[71] A.M. Childs, R. Kothari, R.D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision, SIAM J. Comput. 46 (6) (2017) 1920–1950.
[72] S. Subramanian, S. Brierley, R. Jozsa, Implementing smooth functions of a Hermitian matrix on a quantum computer, Comput. Phys. Commun. (2019).
[73] A. Ambainis, Variable time amplitude amplification and quantum algorithms for linear algebra problems, in: 29th Symposium on Theoretical Aspects of Computer Science, STACS'12, in: LIPIcs, vol. 14, 2012, pp. 636–647.
[74] S. Chakraborty, A. Gilyén, S. Jeffery, The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation, arXiv preprint, arXiv:1804.01973, 2018.
[75] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum 2 (2018) 79.
[76] J.R. McClean, J. Romero, R. Babbush, A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. 18 (2) (2016) 023023.
[77] X. Xu, J. Sun, S. Endo, Y. Li, S.C. Benjamin, X. Yuan, Variational algorithms for linear algebra, arXiv preprint, arXiv:1909.03898, 2019.
[78] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P.J. Coles, Variational quantum linear solver: a hybrid algorithm for linear systems, arXiv preprint, arXiv:1909.05820, 2019.
[79] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nature 549 (7671) (2017) 195.
[80] S. Lloyd, M. Mohseni, P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, arXiv preprint, arXiv:1307.0411, 2013.