




Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicators

Kouhei Nakaji ^{1,2} Shumpei Uno ^{3,2} Yohichi Suzuki,² Rudy Raymond,^{4,2} Tamiya Onodera,^{4,2} Tomoki Tanaka,^{5,2,1} Hiroyuki Tezuka ^{6,2,1} Naoki Mitsuda,^{7,2} and Naoki Yamamoto^{2,8,*}

¹Graduate School of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522, Japan

²Quantum Computing Center, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522, Japan

³Mizuho Research & Technologies, Ltd., 2-3 Kanda-Nishikicho, Chiyoda-ku, Tokyo 101-8443, Japan

⁴IBM Quantum, IBM Research-Tokyo, 19-21 Nihonbashi Hakozaiki-cho, Chuo-ku, Tokyo 103-8510, Japan

⁵Mitsubishi UFJ Financial Group, Inc. and MUFG Bank, Ltd., 2-7-1 Marunouchi, Chiyoda-ku, Tokyo 100-8388, Japan

⁶Sony Group Corporation, 1-7-1 Konan, Minato-ku, Tokyo 108-0075, Japan

⁷Sumitomo Mitsui Trust Bank, Ltd., 1-4-1, Marunouchi, Chiyoda-ku, Tokyo 100-8233, Japan

⁸Department of Applied Physics and Physico-Informatics, Keio University, Hiyoshi 3-14-1, Kohoku-ku, Yokohama 223-8522, Japan



(Received 7 April 2021; accepted 4 April 2022; published 20 May 2022)

Efficient methods for loading given classical data into quantum circuits are essential for various quantum algorithms. In this paper, we propose an algorithm called Approximate Amplitude Encoding that can effectively load all the components of a given real-valued data vector into the amplitude of quantum state, while the previous proposal can load only the absolute values of those components. The key of our algorithm is to variationally train a shallow parameterized quantum circuit, using the results of two types of measurement: the standard computational-basis measurement plus the measurement in the Hadamard-transformed basis, introduced in order to handle the sign of the data components. The variational algorithm changes the circuit parameters so as to minimize the sum of two costs corresponding to those two measurement basis, both of which are given by the efficiently computable maximum mean discrepancy. We also consider the problem of constructing the singular value decomposition entropy via the stock market data set to give a financial market indicator; a quantum algorithm (the variational singular value decomposition algorithm) is known to produce a solution faster than classical, which yet requires the sign-dependent amplitude encoding. We demonstrate, with an in-depth numerical analysis, that our algorithm realizes loading of time series of real stock prices on quantum state with small approximation error, and thereby it enables constructing an indicator of the financial market based on the stock prices.

DOI: [10.1103/PhysRevResearch.4.023136](https://doi.org/10.1103/PhysRevResearch.4.023136)

I. INTRODUCTION

Quantum computing is expected to solve problems that cannot be solved efficiently by any classical means. The promising quantum algorithms are Shor's factoring algorithm [1] and the Grover search algorithm [2]. After these landmark findings, a number of quantum algorithms have been developed, including quantum-enhanced linear algebra solvers [3–14]. An important caveat is that those algorithms assume that the classical data (i.e., elements of the linear equation) has been loaded into the (real) amplitude of a quantum state, i.e., *amplitude encoding*. However, to realize the amplitude encoding without ancillary qubits, in general we are required to operate quantum circuit with exponential depth

with respect to the number of qubits [15–26]. Hence there have been developed several techniques to achieve amplitude encoding without using an exponential depth circuit, e.g., the method introducing ancillary qubits [27–30], which, however, may introduce an exponential number of the ancillary qubits in the worst case. References [31–34] achieve this purpose by limiting the data to a unary one. Also, Refs. [35–39] employ the black-box oracle approach. Note that these are perfect or precision-guaranteed data-loading methods, which consequently can require a large quantum circuit involving hard-to-implement gate operations. On the other hand, there are many problems that need only an approximate calculation (e.g., calculation of a financial market indicator mentioned below), in which case it is reasonable to seek an approximate data-loading method that effectively runs even on currently available shallow and limited-structural quantum circuit.

In this paper, we propose an algorithm called the approximate amplitude encoding (AAE) that trains a shallow parameterized quantum circuit (PQC) to approximate the ideal exact data-loading process. Note that, because of unavoidable approximation error, the application of AAE must

*yamamoto@appi.keio.ac.jp

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

be the one that allows imperfection in the focused quantity, such as a global trend of the financial market indicator, which will be described later. That is, the scope of this paper is not to propose a perfect or precision-guaranteed encoding algorithm. Rather, AAE is a data-loading algorithm that works with fewer gates, despite the unavoidable error caused by the limited representation ability of a fixed ansatz and possibly the incomplete optimization.

To describe the problem more precisely, let $|\text{Data}\rangle$ be the target n -qubit state whose amplitude represents the classical data component. Then AAE provides the training policy of a PQC represented by the unitary $U(\theta)$, so that the finally obtained $U(\theta)$ followed by another shallow circuit V approximately generates $|\text{Data}\rangle$, with the help of an auxiliary qubit. Namely, as a result of training, $VU(\theta)|0\rangle^{\otimes n}|0\rangle$ approximates the state $e^{i\alpha}|\text{Data}\rangle|y\rangle$, where $|y\rangle$ is a state of the auxiliary qubit and $e^{i\alpha}$ is the global phase. Hence, though there appears an approximation error, the $O(1) \sim O(\text{poly}(n))$ -depth quantum circuit $VU(\theta)$ achieves the approximate data loading, instead of the ideal exponential-depth circuit [15–26]. It should also be mentioned that the number of classical bits for storing 2^n dimensional vector data is $O(2^n)$, while the number of qubits for this purpose is $O(\text{poly}(n))$ in several data-loading algorithms [15–26,31–39] and the proposed AAE is included in this class.

Our work is motivated by Ref. [40], which proposed a variational algorithm for constructing an approximate quantum data receiver, in the framework of generative adversarial network (GAN); the idea is to train a shallow PQC so that the absolute values of the amplitude of the final state approximate the absolute values of the data vector components. Hence the method is limited to the case where the sign of the data components does not matter or the case where the data are given by a probability vector as in the setting of [40]. In other words, this method cannot be applied to a quantum algorithm based on the amplitude encoding that needs loading the classical data onto the quantum state without dropping their sign. In contrast, our proposed method can encode the sign in addition to the absolute value, although there may be an approximation error between the generated state and the target state.

As another contribution of this paper, we show that the combination of our AAE algorithm and the variational quantum Singular Value Decomposition (qSVD) algorithm [41] offers an alternative quantum algorithm for computing the SVD entropy for stock price dynamics [42], which is used as a good indicator of the financial market. In fact, this algorithm requires that the signs of the stock price data are correctly loaded into the quantum amplitudes; on the other hand, the goal is to capture a global trend of the SVD entropy over time rather than its precise values, meaning that this problem satisfies the basic requirement of AAE described in the second paragraph. We give an in-depth numerical simulation with a set of real stock price data, to demonstrate that this algorithm generates a good approximating solution of the correct SVD entropy.

The rest of the paper is organized as follows. In Sec. II we describe the algorithm of AAE. Section III gives a demonstration of AAE applied to approximately compute the SVD entropy for stock market dynamics. Finally, we conclude the paper with some remarks in Sec. IV.

II. APPROXIMATE AMPLITUDE ENCODING ALGORITHM

A. The goal of the AAE algorithm

In quantum algorithms that process a classical data represented by a real-valued N -dimensional vector \mathbf{d} , first it has to be encoded into the quantum state; a particular encoding that can potentially be linked to quantum advantage is to encode \mathbf{d} to the amplitude of an n -qubits state $|\text{Data}\rangle$. More specifically, given $|j\rangle$ as $|j\rangle = |j_1 j_2 \cdots j_n\rangle$ where j_k is the state of the k th qubit in computational basis and $j = \sum_{k=1}^n 2^{n-k} j_k$, the data quantum state is given by

$$|\text{Data}\rangle = \sum_{j=0}^{N-1} \mathbf{d}_j |j\rangle, \quad (1)$$

where $N = 2^n$ and \mathbf{d}_j denotes the j th element of the vector \mathbf{d} . Also here \mathbf{d} is normalized; $\sum_j \mathbf{d}_j^2 = 1$. Recall that, even when all the elements of \mathbf{d} are fully accessible, in general, a quantum circuit for generating the state (1) requires an exponential number of gates, which might destroy the quantum advantage [15–26].

In contrast, our algorithm uses a ℓ -depth PQC [hence composed of $O(\ell n)$ gates] to try to approximate the ideal state (1). The depth ℓ is set to be $O(1) \sim O(\text{poly}(n))$. Suppose now that, given an N -dimensional vector \mathbf{a} , the state generated by a PQC, represented by the unitary matrix $U(\theta)$ with θ the vector of parameters, is given by $U(\theta)|0\rangle^{\otimes n} = \sum_{j=0}^{N-1} \mathbf{a}_j |j\rangle$. If the probability to have $|j\rangle$ as a result of measurement in the computational basis is \mathbf{d}_j^2 , this means $|\mathbf{a}_j| = |\mathbf{d}_j|$ for all j . Therefore, if only the absolute values of the amplitudes are necessary in a quantum algorithm after the data loading as in the case of [40], the goal is to train $U(\theta)$ so that the following condition is satisfied:

$$|\mathbf{a}_j|^2 = |\langle j|U(\theta)|0\rangle|^2 = \mathbf{d}_j^2, \quad \forall j \in [0, 1, \dots, N-1]. \quad (2)$$

However, some quantum algorithms need a quantum state containing \mathbf{d}_j itself, rather than \mathbf{d}_j^2 . Naively, hence, the goal is to train $U(\theta)$ so that $U(\theta)|0\rangle^{\otimes n} = |\text{Data}\rangle$. But as will be discussed later, in general we need an auxiliary qubit and thereby aim to train $U(\theta)$ so that

$$VU(\theta)|0\rangle^{\otimes n}|0\rangle = e^{i\alpha}|\text{Data}\rangle|y\rangle, \quad (3)$$

where V represents a fixed operator containing postselection and $e^{i\alpha}$ is the global phase. $|0\rangle$ in the left-hand side and $|y\rangle$ in the right-hand side are the auxiliary qubit state, which might not be necessary in a particular case (Case 1 is shown later). This is the goal of the proposed AAE algorithm. When Eq. (3) is satisfied, the first n -qubits of $VU(\theta)|0\rangle^{\otimes n}$ serve as an input of the subsequent quantum algorithm.

In the following, we assume that all matrix components of $U(\theta)$ are real in the computational basis for any θ , to ensure that $U(\theta)|0\rangle^{\otimes n}$ generates only real amplitude quantum states. In particular, we take $U(\theta)$ composed of only the parameterized R_y rotational gate $R_y(\theta_r) = \exp(-i\theta_r \sigma_y/2)$ and controlled-not (CNOT) gates; here θ_r is the r th element of θ and σ_y is the Pauli Y operator. There is still a huge freedom for constructing the PQC as a sequence of R_y and CNOT gates, but in this paper we take the so-called hardware efficient ansatz [43] due to its high expressibility or rich state

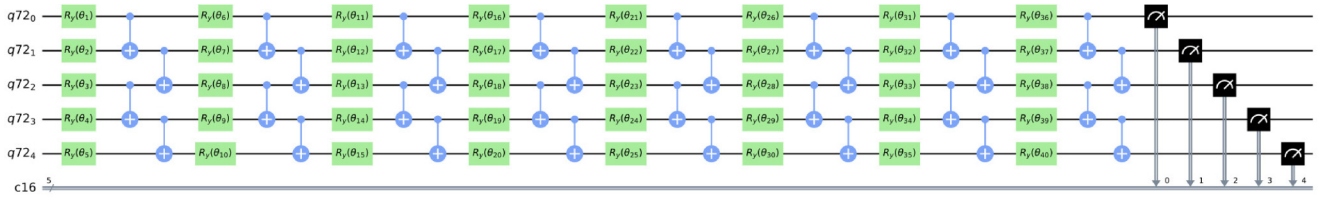


FIG. 1. Example of the structure of the hardware efficient ansatz $U(\theta)$, composed of five qubits with eight layers. We use the ansatz in the numerical demonstration in Sec. III. Each layer is composed of the set of parameterized single-qubit rotational gate $R_y(\theta_r) = \exp(-i\theta_r\sigma_y/2)$ and controlled-not (CNOT) gates that connect adjacent qubits; θ_r is the r th parameter and σ_y is the Pauli Y operator [hence $U(\theta)$ is a real matrix]. We randomly initialize all θ_r at the beginning of each training.

generation capability. We show the example of the structure of the hardware efficient ansatz in Fig. 1. Note that according to the literature [44], the alternating layered ansatz proposed in [45] also has high expressibility comparable to the hardware efficient ansatz; thus, the alternating layered ansatz is another viable ansatz for our problem.

B. The proposed algorithm

This section is twofold; first we identify a condition that guarantees the equality in Eq. (3); then, based on this condition, we specify a valid cost function and describe the design procedure of $U(\theta)$. The algorithm depends on the following two cases related to the elements of target \mathbf{d} :

Case 1: The elements of \mathbf{d} are all nonpositive or all nonnegative.

Case 2: Otherwise.

It should be noted that, even in Case 1, the previously proposed method [40] does not always load the signs correctly. For instance, suppose that we aim to create the ansatz state to approximate the target data state $|\text{Data}\rangle = (|0\rangle + |1\rangle + |2\rangle + |3\rangle)/2$. The method [40] guarantees only that, even ideally (i.e., the case where the cost takes the minimum value zero), the absolute value of the amplitude of $U(\theta^*)|0\rangle^{\otimes n}$ is $(1/2, 1/2, 1/2, 1/2)$; but the output state can be, e.g., $U(\theta^*)|0\rangle^{\otimes n} = (|0\rangle - |1\rangle + |2\rangle - |3\rangle)/2$. On the other hand, our method guarantees that, in the ideal case, the output state is exactly the target state, i.e., $U(\theta^*)|0\rangle^{\otimes n} = (|0\rangle + |1\rangle + |2\rangle + |3\rangle)/2 = |\text{Data}\rangle$.

1. Condition for the perfect encoding

In Case 1, we consider the following two conditions:

$$|\langle j|U(\theta)|0\rangle^{\otimes n}|^2 = \mathbf{d}_j^2 \quad (\forall j), \quad (4)$$

$$\begin{aligned} |\langle j|H^{\otimes n}U(\theta)|0\rangle^{\otimes n}|^2 &= \left(\sum_{k=0}^{N-1} \mathbf{d}_k \langle j|H^{\otimes n}|k\rangle \right)^2 \\ &\equiv (\mathbf{d}_j^H)^2 \quad (\forall j). \end{aligned} \quad (5)$$

Note that \mathbf{d}_j^H is classically computable with complexity $O(N \log N)$, by using the Walsh-Hadamard transform [46]; in particular, if \mathbf{d} is a sparse vector, this complexity can be reduced; that is, if \mathbf{d} has only $K = N^\alpha$ nonzero elements ($0 < \alpha < 1$), there exists a modified Walsh-Hadamard-based algorithm with computational complexity $O(K \log K \log(N/K))$ such that the success probability asymptotically approaches to 1 as N increases [47].

If both two conditions (4) and (5) are satisfied, it is guaranteed that our goal is exactly satisfied, which is stated in the following theorem (the proof is found in Appendix A):

Theorem 1. In Case 1, if the n -qubits PQC $U(\theta)$ satisfies Eqs. (4) and (5), then $U(\theta)|0\rangle^{\otimes n} = \sum_j \mathbf{d}_j |j\rangle$ or $U(\theta)|0\rangle^{\otimes n} = -\sum_j \mathbf{d}_j |j\rangle$ holds.

In Case 2, i.e., the case where some (not all) elements of \mathbf{d} are nonnegative while the others are positive, the target state $|\text{Data}\rangle$ can be decomposed to

$$|\text{Data}\rangle = |\text{Data}^+\rangle + |\text{Data}^-\rangle, \quad (6)$$

where the amplitudes of $|\text{Data}^+\rangle$ are positive and those of $|\text{Data}^-\rangle$ are nonpositive. Then, by introducing an auxiliary single qubit, we can represent the state $|\text{Data}\rangle$ in the form considered in Case 1; that is, the amplitudes of the $(n+1)$ -qubits state

$$|\bar{\psi}\rangle \equiv |\text{Data}^+\rangle|0\rangle - |\text{Data}^-\rangle|1\rangle \quad (7)$$

are nonnegative and $\langle \bar{\psi}|\bar{\psi}\rangle = 1$. We write this state as $|\bar{\psi}\rangle = \sum_{i=0}^{2N-1} \bar{\mathbf{d}}_i |i\rangle$ in terms of the computational basis $\{|j\rangle\}$ and the corresponding $2N$ -dimensional vector $\bar{\mathbf{d}}$. Then Theorem 1 states that, if the conditions

$$|\langle j|U(\theta)|0\rangle^{\otimes n+1}|^2 = \bar{\mathbf{d}}_j^2 \quad (\forall j), \quad (8)$$

$$\begin{aligned} |\langle j|H^{\otimes n+1}U(\theta)|0\rangle^{\otimes n+1}|^2 &= \left(\sum_{k=0}^{2N-1} \bar{\mathbf{d}}_k \langle j|H^{\otimes n+1}|k\rangle \right)^2 \\ &\equiv (\bar{\mathbf{d}}_j^H)^2 \quad (\forall j) \end{aligned} \quad (9)$$

are satisfied, then $U(\theta)|0\rangle^{\otimes n+1} = \pm|\bar{\psi}\rangle$ holds. Further, once we obtain $|\bar{\psi}\rangle$, this gives us the target $|\text{Data}\rangle$, via the following procedure. That is, operating the Hadamard transform to the last auxiliary qubit yields

$$\begin{aligned} I^{\otimes n} \otimes H |\bar{\psi}\rangle &= \frac{|\text{Data}^+\rangle - |\text{Data}^-\rangle}{\sqrt{2}} |0\rangle \\ &\quad + \frac{|\text{Data}^+\rangle + |\text{Data}^-\rangle}{\sqrt{2}} |1\rangle, \end{aligned} \quad (10)$$

and then the postselection of $|1\rangle$ via the measurement on the last qubit in Eq. (10) gives us $|\text{Data}\rangle$ in the first n -qubits. The above result is summarized as follows.

Theorem 2. In Case 2, suppose that the $(n+1)$ -qubits PQC $U(\theta)$ satisfies Eqs. (8) and (9). Then, if the measurement result of the last qubit in the computational basis for the state

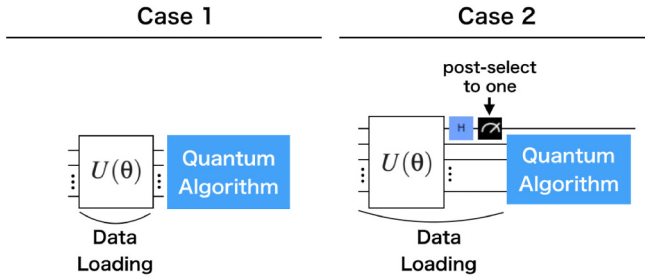


FIG. 2. Overview of the data loading in Case 1 and Case 2.

$(I^{\otimes n} \otimes H)U(\theta)|0\rangle^{\otimes n+1}$ is $|1\rangle, |\text{Data}\rangle$ is generated. That is,

$$(I^{\otimes n} \otimes |1\rangle\langle 1|)(I^{\otimes n} \otimes H)U(\theta)|0\rangle^{\otimes n+1} \propto |\text{Data}\rangle|1\rangle.$$

This theorem implies that, if $U(\theta)$ is trained so that conditions (8) and (9) are satisfied, $|\text{Data}\rangle$ can be obtained by the above postselection procedure, with success probability nearly $1/2$. Note that, by applying the extra postprocessing, we can obtain $|\text{Data}\rangle$ with success probability 1 instead of $1/2$, which is shown in Appendix B. The overview of the data-loading circuits in Case 1 and Case 2 are summarized in Fig. 2. As seen from the figure, $U(\theta)$ is directly used as the data-loading circuit in Case 1, while we need the postprocessing after $U(\theta)$ in Case 2.

2. Optimization of $U(\theta)$

Here we provide a training method for optimizing $U(\theta)$, so that Eqs. (4) and (5) are nearly satisfied in Case 1, and Eqs. (8) and (9) are nearly satisfied in Case 2, with as small approximation error as possible. For this purpose, we employ the strategy to decrease the maximum mean discrepancy (MMD) cost [48,49], which was previously proposed for training the Quantum Born Machine [9,50]. Note that the other costs, the Stein discrepancy (SD) [9] and the Sinkhorn divergence (SHD) [9] can also be taken, but in this paper we use MMD for its ease of use.

The MMD is a cost of the discrepancy between two probability distributions: $q_\theta(j)$, the model probability distribution, and $p(j)$, the target distribution. The cost function $\mathcal{L}_{MMD}(q_\theta, p)$ is defined as

$$\mathcal{L}_{MMD}(q_\theta, p) \equiv \gamma_{MMD}(q_\theta, p)^2, \quad \gamma_{MMD}(q_\theta, p) = \left| \sum_{j=0}^{N-1} q_\theta(j)\Phi(j) - \sum_{j=0}^{N-1} p(j)\Phi(j) \right|, \quad (11)$$

where $\Phi(j)$ is a function that maps j to a feature space. Thus, given the kernel $\kappa(j, k)$ as $\kappa(j, k) = \Phi(j)^T \Phi(k)$, it holds

$$\mathcal{L}_{MMD}(q_\theta, p) = \mathbf{E}_{\substack{j \sim q_\theta \\ k \sim q_\theta}}[\kappa(j, k)] - 2\mathbf{E}_{\substack{j \sim q_\theta \\ k \sim p}}[\kappa(j, k)] + \mathbf{E}_{\substack{j \sim p \\ k \sim p}}[\kappa(j, k)], \quad (12)$$

where, for example, one of the expectation values is defined by

$$\mathbf{E}_{\substack{j \sim q_\theta \\ k \sim q_\theta}}[\kappa(j, k)] = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \kappa(j, k)q_\theta(j)q_\theta(k). \quad (13)$$

Note that, even though the index of the sum goes till $N - 1$ in Eq. (13), we can efficiently estimate the expectation value without $O(N)$ computation by sample averaging as follows. First, given N_{shot} as the number of samples for each index, we sample $\{j_\ell\}_{\ell=0}^{N_{\text{shot}}-1}$ and $\{k_\ell\}_{\ell=0}^{N_{\text{shot}}-1}$ according to the probability distribution $q_\theta(\cdot)$; note that, in our case, $q_\theta(j)$ is the probability to obtain $|j\rangle$ as a result of measuring the final state of PQC, and thus we can obtain samples just by measuring the final state multiple times. Then, using the samples $\{j_\ell\}_{\ell=0}^{N_{\text{shot}}-1}$ and $\{k_\ell\}_{\ell=0}^{N_{\text{shot}}-1}$, we can approximate the expectation value as

$$\mathbf{E}_{\substack{j \sim q_\theta \\ k \sim q_\theta}}[\kappa(j, k)] \simeq \frac{1}{N_{\text{shot}}} \sum_{\ell=0}^{N_{\text{shot}}-1} \kappa(j_\ell, k_\ell). \quad (14)$$

The approximation error is bounded by $O(1/\sqrt{N_{\text{shot}}})$ with high probability; this fact can be proven by using the bound for probability distributions such as Chernoff bound [51] combined with the technique to derive the error bound; see, e.g., [9,52]. Similarly, we can efficiently estimate the other expectation values in \mathcal{L}_{MMD} by the sample-averaging technique; as a result, we can estimate \mathcal{L}_{MMD} with guaranteed error $O(1/\sqrt{N_{\text{shot}}})$, via $O(N_{\text{shot}}) < O(N)$ computation.

It should also be noted that when the kernel is *characteristic* [48,49], then $\mathcal{L}_{MMD}(q_\theta, p) = 0$ means $q_\theta(j) = p(j)$ for all j . In this paper, we take a one-dimensional Gaussian kernel $\kappa(j, k) = C \exp[-(j - k)^2/2\sigma^2]$ with a positive constant C , which is characteristic.

In Case 1, the goal is to train the model distributions

$$q_\theta(j) = |\langle j|U(\theta)|0\rangle^{\otimes n}|^2, \quad q_\theta^H(j) = |\langle j|H^{\otimes n}U(\theta)|0\rangle^{\otimes n}|^2,$$

so that they approximate the target distributions

$$p(j) = \mathbf{d}_j^2, \quad p^H(j) = (\mathbf{d}_j^H)^2, \quad (15)$$

respectively. In Case 2, the model distributions

$$q_\theta(j) = |\langle j|U(\theta)|0\rangle^{\otimes n+1}|^2, \quad q_\theta^H(j) = |\langle j|H^{\otimes n+1}U(\theta)|0\rangle^{\otimes n+1}|^2$$

are trained so that they approximate the target distributions

$$p(j) = \bar{\mathbf{d}}_j^2, \quad p^H(j) = \bar{\mathbf{d}}_j^{H^2}, \quad (16)$$

respectively. In both cases, our training policy is to minimize the following cost function:

$$\mathcal{L}(\theta) = \frac{\mathcal{L}_{MMD}(q_\theta, p) + \mathcal{L}_{MMD}(q_\theta^H, p^H)}{2}. \quad (17)$$

Actually, $\mathcal{L}(\theta)$ becomes zero if and only if $\mathcal{L}_{MMD}(q_\theta, p) = 0$ and $\mathcal{L}_{MMD}(q_\theta^H, p^H) = 0$, or equivalently $q_\theta(j) = p(j)$ and $q_\theta^H(j) = p^H(j)$ for all j as long as we use a characteristic kernel.

To minimize the cost function (17), we take the standard gradient descent algorithm. In particular as we note at the end of Sec. II A, we consider the PQC where each parameter θ_r is embedded in the quantum circuit in the form $\exp(-i\theta_r\sigma_y/2)$. In this case, the gradients of q_θ and q_θ^H with respect to θ_r can be computed by using the parameter shift rule [53] as

$$\frac{\partial q_\theta(j)}{\partial \theta_r} = q_{\theta_r^+}^+(j) - q_{\theta_r^-}^-(j), \quad \frac{\partial q_\theta^H(j)}{\partial \theta_r} = q_{\theta_r^+}^{H+}(j) - q_{\theta_r^-}^{H-}(j), \quad (18)$$

where $q_{\theta_r}^{\pm}(j) = |\langle j|U_{r\pm}(\boldsymbol{\theta})|0\rangle|^2$, $q_{\theta_r}^{H\pm}(j) = |\langle j|HU_{r\pm}(\boldsymbol{\theta})|0\rangle|^2$. The shifted unitary operator is defined by

$$U_{r\pm}(\boldsymbol{\theta}) = U_{r\pm}(\{\theta_1, \dots, \theta_{r-1}, \theta_r, \theta_{r+1}, \dots, \theta_R\}) \\ = U(\{\theta_1, \dots, \theta_{r-1}, \theta_r \pm \pi/2, \theta_{r+1}, \dots, \theta_R\}), \quad (19)$$

with R as the number of the parameters, which can be written as $R = \ell n$ (recall that ℓ is the depth of PQC). Then, by differentiating (12) and using (18), the gradient of \mathcal{L} can be explicitly computed [50] as

$$2 \frac{\partial \mathcal{L}}{\partial \theta_r} = \mathbf{E}_{\substack{j \sim q_{\theta_r}^+ \\ k \sim q_{\theta}^+}} [\kappa(j, k)] - \mathbf{E}_{\substack{j \sim q_{\theta_r}^- \\ k \sim q_{\theta}^-}} [\kappa(j, k)] \\ - \mathbf{E}_{\substack{j \sim q_{\theta_r}^+ \\ k \sim p}} [\kappa(j, k)] + \mathbf{E}_{\substack{j \sim q_{\theta_r}^- \\ k \sim p}} [\kappa(j, k)] \\ + \mathbf{E}_{\substack{j \sim q_{\theta_r}^{H+} \\ k \sim q_{\theta}^{H+}}} [\kappa(j, k)] - \mathbf{E}_{\substack{j \sim q_{\theta_r}^{H-} \\ k \sim q_{\theta}^{H-}}} [\kappa(j, k)] \\ - \mathbf{E}_{\substack{j \sim q_{\theta_r}^{H+} \\ k \sim p^H}} [\kappa(j, k)] + \mathbf{E}_{\substack{j \sim q_{\theta_r}^{H-} \\ k \sim p^H}} [\kappa(j, k)]. \quad (20)$$

We can approximately compute the gradient (20) by sampling j and k from the distributions q_{θ} , $q_{\theta_r}^+$, $q_{\theta_r}^-$, q_{θ}^H , $q_{\theta_r}^{H+}$, $q_{\theta_r}^{H-}$, p , and p^H similar to the case of Eq. (14). Then, using the gradient descent algorithm with Eq. (20), we can update the vector $\boldsymbol{\theta} = (\theta_1, \dots, \theta_R)$ to the direction that minimizes $\mathcal{L}(\boldsymbol{\theta})$. Note that, in the above sampling approach, the estimation error of the gradient vectors does not depend on N but only on the number of samples N_{shot} . This is because each gradient is written as the sum of the expectation values (20) and each expectation value can be estimated with the error $O(1/\sqrt{N_{\text{shot}}})$, as discussed around Eq. (14).

Lastly we remark that we may be able to utilize the classical shadow technique [54,55] and its extension [56], to significantly reduce the number of measurements as follows. Recall that the density matrix of an n -qubit quantum state ρ is a linear combination of 4^n Pauli terms written as $\rho = \sum_P \alpha_P P$ where $\alpha_P \in \mathbf{R}$ and $P \in \{I, X, Y, Z\}^{\otimes n}$. Also in the proposed method, the quantum state generated from the PQC is measured either in the computational basis (i.e., the eigenstates of $Z^{\otimes n}$) or in the rotated computational basis via the Hadamard gate (i.e., the eigenstates of $X^{\otimes n}$). Applying the classical shadow technique allows us to estimate all coefficients α_P for $P \in \{I, Z\}^n$ and for $P \in \{I, X\}^n$ within an additive error ϵ by spending $\propto \text{poly}(n)/\epsilon^2$ number of measurements. We can also extend the measurement basis by the eigenstates of Y , and by applying the classical shadow technique to probabilistically check if the coefficients of α_P 's are correct. We leave the details of the analysis for future work.

C. Computational complexity of the AAE algorithm

We have seen above that the number of measurements required for estimating the cost and the gradient vector does not scale exponentially with the number of qubits. However, we must still solve the issues that often appear in the standard variational quantum algorithms (VQAs), for ensuring the scalability of our algorithm. Below we pose a few typical issues and describe how we could handle them.

Firstly, it is theoretically shown that the general VQA with highly expressive PQC has the barren plateau issue [57]; that

is, the gradient of the cost function becomes exponentially small as the number of qubits increases. Our algorithm may also have the same issue, even though $U(\boldsymbol{\theta})$ is limited to be real in the computational basis. For mitigating this issue, several approaches have been proposed: e.g., circuit initialization [58], special structured ansatz [45], and parameter embedding [59], while further studies are necessary to examine the validity of these approaches. The methods [58] and [59] are applicable to our algorithm; on the other hand, the approach of [45] depends on the detail of the cost function (the MMD cost in our case), and we need further investigation to see the applicability of this method to our algorithm. Related to this point, we also need to carefully address the issue that the landscape of the cost function in VQA may have many local minima, which may result in many trials of the training for PQC. Because this local minima issue is ubiquitous in classical optimization problems, we can employ several established classical optimizers [60]. Other approaches may also be applicable to our algorithm, but their theoretical understanding, particularly the convergence proof, are still to be investigated.

The next typical issue that our algorithm shares in common with the general VQA is that the depth of circuit tends to become bigger in order to reduce the value of cost function below a certain specified value [61]. Various attempts to reduce the circuit depth have been made in the literature [62–65], and some of those are applicable to our algorithm. Also, Refs. [66,67] provide methods to split a large quantum circuit to several small quantum circuits. Even with those assistance, we need further theoretical development to conclude that $O(\text{poly}(n))$ depth is actually enough for training any data-loading circuit. Nevertheless, in contrast to the problems that require near-perfect cost minimization such as VQE in chemistry, the depth needed for our algorithm is smaller as long as very precise data loading is not necessary. Computation of the singular value decomposition entropy, which we will discuss in Sec. III, is an example of such quantum algorithms.

Table I summarizes the computational complexity of AAE, under the assumption that the PQC with the number of gates $O(\text{poly}(n))$ achieves the approximate data loading with sufficient precision for the given problem. We show two cases: (1) the case when using AAE (denoted by ‘‘AAE’’ in the table) and (2) the case when exactly encoding data (denoted by ‘‘exact encoding’’) [15–26,68]; for each case we show both the results when the data vector is dense or sparse. For exact encoding with sparse data, we show the result in Ref. [68]. Also, the number of nonzero elements in sparse data is denoted by K . We divide the computational complexity into two stages: the training stage (training) and the execution stage of the quantum algorithm (execution). In the case of exact encoding, there is no training stage. The total number of gate operations required in the training stage of AAE is denoted by N_{train} , and that for the main quantum algorithm is denoted by N_{alg} . The total number of measurements required to retrieve the output of the algorithm, satisfying sufficient precision for the given problem, is denoted by N_{mes} . We emphasize the total computational complexity in the execution stage by bold letters.

Regarding the computational complexity, the merit of using AAE exists in the computational complexity in execution

TABLE I. Overview of the computational complexity for our algorithm (AAE) and the case for exact encoding [15–26,68]; for each case we show both the results when the data vector is dense or sparse. For exact encoding with sparse data, we show the result in Ref. [68]. We divide the computational complexity into two stages: the training stage (training) and the execution stage of the quantum algorithm (execution). In the case of exact encoding, there is no training stage. The total number of gate operations required in the training stage of AAE is denoted by N_{train} (as the gate operations, we include both the single qubit operations and the two-qubit operations). The number of gates required for the main quantum algorithm is denoted by N_{alg} , and the total number of measurements required to retrieve the output satisfying a sufficient precision for each problem is denoted by N_{mes} .

Strategy	(1) AAE		(2) Exact encoding	
	Dense	Sparse	Dense	Sparse
No. of nonzero elements in the data	N	K	N	K
No. of gates in the data-loading circuit	$O(\text{poly}(\log N))$		$O(N)$	$O(K \log N)$
Computational complexity (training)	Classical (Walsh Hadamard Transform)	$O(N \log N)$	$O[K \log K \times \log(\frac{N}{K})]$	—
	Quantum (total no. of gate operations)	N_{train}		
Computational complexity (execution)	(a) No. of gate operations for the data loading per one measurement	$O(\text{poly}(\log N))$		$O(N)$
	(b) No. of gate operations for quantum algorithm per one measurement			N_{alg}
	(c) No. of measurements			N_{mes}
	Total = [(a) + (b)] × (c)	$O((\text{poly}(\log N) + N_{\text{alg}}) \times N_{\text{mes}})$		$O(N + N_{\text{alg}}) \times N_{\text{mes}}$
			$O(K + N_{\text{alg}}) \times N_{\text{mes}}$	

stage. In particular, the computational complexity of AAE is $O(N)$ times smaller in the execution stage than that of the exact encoding method, when N_{alg} is of the order of $\text{poly}(\log N)$. Such a merit is favorable in particular when the data-loading circuit is used repeatedly. One example is when N_{mes} is as large as $\text{poly}(\log N)$. Another example is when the data-loading circuit is usable in various problems; for example, once we load a training data set for a particular quantum machine learning problem, it can be utilized in other machine learning models.

For the training stage of AAE, however, we need further discussion. Firstly, the $O(N \log N)$ classical computation for the case of dense data, which comes from the Walsh-Hadamard transform in AAE, seems costly. However, other data-loading methods implicitly contains processes to register the data. For instance, the exact encoding method shown in the table needs the process that compiles the data into $O(N)$ quantum gates, which at least requires $O(N)$ computational complexity. Also, even when a quantum random access memory (QRAM) [69] is ideally available, registering $O(N)$ data into the QRAM requires at least $O(N)$ computational complexity; e.g., in the QRAM proposed in Ref. [70], $O(N \log N)$ gate operations are necessary for the data registration.

Secondly, related with the above-mentioned trainability issues in VQA, N_{train} might become large, in the absence of some elaborated techniques for VQA. A promising approach is to take the convex relaxation on the target cost function, in which case the total number of iterations to achieve $\tilde{\mathcal{L}}(\theta) < \epsilon$ is $O(\text{poly}(\log N)/\epsilon^2)$ [71], where $\tilde{\mathcal{L}}$ is the relaxed convex cost function. Furthermore, it was shown in [71] that the total

number of measurements for realizing $\tilde{\mathcal{L}}(\theta) < \epsilon$ is $O(GS/\epsilon^2)$, where G is the upper bound of $|\partial \tilde{\mathcal{L}}/\partial \theta_r|$ over the parameter space and S is a constant that relates with the size of the parameter space. Thus, combining these two complexities, we find that the total number of gate operations N_{train} to achieve $\tilde{\mathcal{L}}(\theta) < \epsilon$ is $O(\text{poly}(\log N)\text{poly}(1/\epsilon))$, provided that the number of parameters is of the order of $\text{poly}(\log N)$. Note that, of course, the convex relaxation appends an additional error related with the gap between the original cost function and the relaxed one. Nonetheless, we hope that, in our case, this gap could be minor compared to the target precision of the cost function (SVD entropy in our case); we will study this problem as an important future work for having scalability.

D. Some modification on the AAE algorithm

Before concluding this section, we consider four types of modifications on the AAE algorithm. The first two are the change of the cost function, and the next one discusses the change of the conditions for perfect encoding; the fourth one is on the possibility to formulate the AAE algorithm in the GAN framework.

The first one is simple; we may be able to build the cost function as the weighted average of $\mathcal{L}_{\text{MMD}}(q_\theta, p)$ and $\mathcal{L}_{\text{MMD}}(q_\theta^H, p^H)$ instead of the current equally weighted average (17). It is worth investigating the effect of this modification.

The second possible change is taking a cost function other than MMD. That is, as mentioned above, Stein discrepancy (SD) or Sinkhorn divergence (SHD) can serve as a cost

for measuring the difference of two probability distributions. Also, as another type of cost function, readers may wonder if the Kullback-Leibler divergence (KL divergence)

$$\mathcal{L}_{\text{KL}}(p, q_\theta) = \sum_{j=0}^{N-1} \{p(j) \log[p(j)] - p(j) \log[q_\theta(j)]\} \quad (21)$$

would be a more natural cost function for comparing a target distribution $p(j)$ and a model distribution $q_\theta(j)$ with parameter θ .

The gradient $\partial \mathcal{L}_{\text{KL}} / \partial \theta_r$ is given by

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{KL}}}{\partial \theta_r} &= - \sum_{j=0}^{N-1} \frac{p(j)}{q_\theta(j)} \frac{\partial q_\theta(j)}{\partial \theta_r} \\ &= - \sum_{j=0}^{N-1} \frac{p(j)}{q_\theta(j)} [q_{\theta_r}^+(j) - q_{\theta_r}^-(j)] \\ &= - \mathbf{E}_{j \sim q_{\theta_r}^+} \left[\frac{p(j)}{q_\theta(j)} \right] + \mathbf{E}_{j \sim q_{\theta_r}^-} \left[\frac{p(j)}{q_\theta(j)} \right]. \end{aligned} \quad (22)$$

However, we cannot efficiently compute this quantity by sample averaging unlike the case of MMD. For example, we sample $\{j_\ell\}_{\ell=0}^{N_{\text{shot}}-1}$ from $q_{\theta_r}^+(\cdot)$ and may compute the first term of the gradient as

$$\mathbf{E}_{j \sim q_{\theta_r}^+} \left[\frac{p(j)}{q_\theta(j)} \right] \simeq \frac{1}{N_{\text{shot}}} \sum_{\ell=0}^{N_{\text{shot}}-1} \frac{p(j_\ell)}{q_\theta(j_\ell)}, \quad (23)$$

similar to the case of Eq. (14). Then we need to compute the value of $q_\theta(j) = |\langle j|U(\theta)|0\rangle|^2$ for each ℓ , which, however, requires $O(2^n)$ measurements. Therefore, the gradient of the KL-divergence cannot be efficiently computed in our setting, unlike the case of MMD (see [50] for more detailed explanation).

To the contrary, the gradient of SD and SHD as well as MMD are efficiently computable, because the gradient vector is written in terms of the averages of efficiently computable statistical quantities as in Eq. (20) [9].

The third possible change is altering conditions (5) and (9), which is used for characterizing the perfect encoding. In Case 1, we train $U(\theta)$ so that Eqs. (4) and (5) are approximately satisfied; the complexity for computing the right-hand side of Eq. (5) is $O(N \log N)$. However, as seen in the proof of Theorem 1, even if condition (5) is replaced by

$$|\langle 0|H^{\otimes n}U(\theta)|0\rangle|^2 = \left(\sum_{k=0}^{N-1} \mathbf{d}_k \langle 0|H^{\otimes n}|k\rangle \right)^2, \quad (24)$$

the perfect encoding is still achieved; that is, $U(\theta)|0\rangle = \sum_j \mathbf{d}_j |j\rangle$ or $U(\theta)|0\rangle = -\sum_j \mathbf{d}_j |j\rangle$ holds. This implies that we can obtain the data-loading circuit by training $U(\theta)$ so that Eqs. (4) and (24) are approximately satisfied. Then the complexity for computing the right hand side is reduced to $O(N)$. In Case 2, the situation is the same. Therefore, the modified algorithm with the use of conditions (4) and (24) may also work.

Now, as another possibility of changing conditions (5) and (9), readers may wonder that, if we carefully choose an operator X instead of $H^{\otimes n}$, the conditions

$$|\langle j|U(\theta)|0\rangle|^2 = \mathbf{d}_j^2, \quad (25)$$

$$|\langle j|XU(\theta)|0\rangle|^2 = \left(\sum_{k=0}^{N-1} \mathbf{d}_k \langle j|X|k\rangle \right)^2 \quad (26)$$

would also result in $U(\theta)|0\rangle = \sum_j \mathbf{d}_j |j\rangle$ or $U(\theta)|0\rangle = -\sum_j \mathbf{d}_j |j\rangle$ for arbitrary real vector \mathbf{d} . This is clearly favorable because we do not need Case 2; namely, we need neither auxiliary qubits nor the postselection. However, as shown in Appendix C, it seems to be difficult to find such X for arbitrary \mathbf{d} . This is why we consider the two cases depending on \mathbf{d} .

The final possible change is utilizing GAN [72], which was originally proposed as the method to train a generative model. GAN consists of two components: a generator and a discriminator. The generator generates samples (fake data), and the discriminator receives either a real data from a data source or fake data from the generator. The discriminator is trained so that it exclusively classifies the fake data as fake and the real data as real. The generator is trained so that the samples generated by the generator are classified as real by the discriminator. If the training is successfully conducted, we will have a good generative model; i.e., the probability distribution that governs the samples of the generator well approximates the source distribution. As mentioned in Sec. I, the motivating work [40] applied GAN composed of the quantum generator implemented by the PQC and the classical (neural network) discriminator, and demonstrated that the trained PQC approximates the probability distribution $p(j) = \mathbf{d}_j^2$. In our work, in contrast, we do not take the GAN formulation. The main reason is that, in our case, the PQC is trained to learn *two* probability distributions [Eqs. (4) and (5) in Case 1], which cannot be formulated in the ordinary GAN that handles only one generator and one discriminator.

However, customizing GAN to fit into our setting may be doable as follows; we will discuss only Case 1, but the same argument applies to Case 2. The customized GAN is composed of two quantum generators (Generator A and Generator B) and two classical discriminators (Discriminator A and Discriminator B). In particular, we use one PQC to realize the two generators. First, Generator A and Discriminator A correspond to condition (4); output samples of Generator A (fake data A) are obtained by measuring the output state of PQC in computational basis, and Discriminator A receives the real data sampled from $p(j) = \mathbf{d}_j^2$ or the fake data A generated from Generator A. Also, Generator B and Discriminator B correspond to condition (5); output samples of Generator B (fake data B) are generated by measuring the output state of the same PQC yet in the Hadamard basis, and Discriminator B receives the real data sampled from $p^H(j) = \mathbf{d}_j^{H^2}$ or the fake data B generated by Generator B. With this setting, the discriminators are trained so that they will exclusively classify the real and fake data. On the other hand, the PQC is trained so that the outputs of the generators are to be classified as real by the discriminators. Ideally, as a result of the training, we will obtain the generator that almost satisfies (4) and (5).

III. APPLICATION TO SVD ENTROPY CALCULATION FOR FINANCIAL MARKET INDICATOR

This section is devoted to describe the quantum algorithm composed of our AAE and the variational qSVD algorithm [41] for computing the SVD entropy for stock price dynamics [42]. We first give the definition of SVD entropy and then describe the quantum algorithm, with particular emphasis on how the AAE algorithm well fits into the problem of computing the SVD entropy. Finally the in-depth numerical simulation is provided.

A. SVD entropy

The SVD entropy is used as one of the good indicators for forewarning the financial crisis, which is computed by the singular value decomposition of the correlation matrix between stock prices. Let $s_{j,t}$ be the price of the j th stock at time t . Then we define the logarithmic rate of return as follows:

$$r_{jt} = \log(s_{j,t}) - \log(s_{j,t-1}). \quad (27)$$

Also, the correlation matrix C of the set of stocks $j = 1, 2, \dots, N_s$ over the term $t = 1, 2, \dots, T$ is defined as

$$C_{jk} = \sum_{t=1}^T a_{jt} a_{kt}, \quad (28)$$

where

$$a_{jt} = \frac{r_{jt} - \langle r_j \rangle}{\sigma_j \sqrt{N_s T}}. \quad (29)$$

The average $\langle r_j \rangle$ and the standard deviation σ_j over the whole period of term are defined as

$$\langle r_j \rangle = \frac{1}{T} \sum_{t=1}^T r_{jt}, \quad \sigma_j^2 = \frac{1}{T} \sum_{t=1}^T (r_{jt} - \langle r_j \rangle)^2. \quad (30)$$

The correlation matrix C is positive semidefinite, and thus its eigenvalues are nonnegative. In addition, C satisfies

$$\text{Tr}(C) = \sum_{j=1}^{N_s} \sum_{t=1}^T a_{jt}^2 = 1. \quad (31)$$

Now for the positive eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_U$ of the correlation matrix, which satisfy $\sum_{u=1}^U \lambda_u = 1$ from Eq. (31), the SVD entropy S is defined as

$$S = - \sum_{u=1}^U \lambda_u \log \lambda_u. \quad (32)$$

Computation of S in any classical means requires the diagonalization of the $N_s \times N_s$ matrix C , hence its computational complexity is $O(N_s^3)$.

The SVD entropy S has been proposed as an indicator to detect financial crises, such as financial crashes and bubbles, based on the methodology of information theory [42]. In the information theory, entropy measures the randomness of random variables [73]. According to the Efficient Market Hypothesis [74,75], financial markets during normal periods show highly random behavior, which lead to large entropy. In

fact, it is known that the eigenvalue distribution of the correlation matrix C can be well explained by the chiral random matrix theory, except for some large eigenvalues [76–79]. The eigenvector associated with the largest eigenvalue is interpreted as the market portfolio, which consists of the entire stocks, and the eigenvectors associated with the other large eigenvalues are interpreted as the portfolios of stocks belonging to different industrial sectors. The structural relation of these eigenvalues does not change a lot during the normal periods, but it changes drastically at a point related to financial crisis. Actually, the stock prices across the entire market or the clusters of several industrial sectors have been reported to show collective behavior when financial crisis occurred [80–86]; mathematically, this means that the eigenvalues become nearly degenerate, or roughly speaking the probability distribution of the eigenvalues visibly becomes sharp. As a result, the SVD entropy S takes a relatively small value, indicating the financial crisis. This behavior is analogous to that of the statistical mechanical systems near a critical point, where spins form a set of clusters due to the very large correlation length [87]. Lastly we add a remark that a similar and detailed analysis for image processing can be found in Ref. [88].

B. Computation on quantum devices

Computation of the SVD entropy on a quantum device can be performed by first training the PQC $U(\theta)$ by the AAE algorithm, to generate a target state in which the stock data a_{jt} is suitably embedded. Next, the PQC $U_{\text{SVD}}(\xi)$ is variationally trained so that it performs the SVD. Finally, the SVD entropy is estimated from the output state of the entire circuit $U_{\text{SVD}}(\xi)U(\theta)$. In the following, we show the detail discussions of the procedures.

1. The first part: Data loading

The AAE serves as the first part of the entire algorithm; that is, it is used to load the normalized logarithmic rate of return of the stock price data a_{jt} given in Eq. (29) into a quantum state. The target state that the AAE aims to approximate is the following bipartite state:

$$|\text{Data}\rangle = \sum_{j=1}^{N_s} \sum_{t=1}^T a_{jt} |j\rangle_{\text{stock}} |t\rangle_{\text{time}}, \quad (33)$$

where $\{|j\rangle_{\text{stock}}\}_{j=1, \dots, N_s}$ and $\{|t\rangle_{\text{time}}\}_{t=1, \dots, T}$ are the computational basis set constructing the stock index Hilbert space $\mathcal{H}_{\text{stock}}$ and the time index Hilbert space $\mathcal{H}_{\text{time}}$, respectively. The number of qubits needed to prepare this state is $n_s + n_t$, where $n_s = O(\log N_s)$ and $n_t = O(\log T)$, meaning that the quantum approach has an exponential advantage in the memory resource. Also note that the state (33) is normalized because of Eq. (31). The partial trace over $\mathcal{H}_{\text{time}}$ gives rise to

$$\rho_{\text{stock}} = \text{Tr}_{\mathcal{H}_{\text{time}}}(|\text{Data}\rangle\langle\text{Data}|) = \sum_{jk} C_{jk} |j\rangle_{\text{stock}} \langle k|_{\text{stock}}, \quad (34)$$

where C_{jk} is the (j, k) element of the correlation matrix given in Eq. (28); that is, $\rho_{\text{stock}} = C$ is realized on a quantum device. Hence, we need an efficient algorithm to diagonalize ρ_{stock} and

eventually compute the SVD entropy S on a quantum device, and this is the reason why we use the qSVD algorithm.

2. The second part: Quantum singular value decomposition

We apply the qSVD algorithm [41] to achieve the above-mentioned diagonalization task. The point of this algorithm lies in the fact that diagonalizing $\rho_{\text{stock}} = C$ is equivalent to realizing the Schmidt decomposition of $|\text{Data}\rangle$:

$$|\text{Data}\rangle = \sum_{m=1}^M c_m |v_m\rangle_{\text{stock}} |v'_m\rangle_{\text{time}}, \quad (35)$$

where $\{c_m\}_{m=1}^M$ are the Schmidt coefficients, and $\{|v_m\rangle_{\text{stock}}\}_{m=1}^M$ and $\{|v'_m\rangle_{\text{time}}\}_{m=1}^M$ are set of orthogonal states, with $M \leq \min(N_s, T)$; note that in general these are not the computational basis. Actually, in this representation, ρ_{stock} is calculated as

$$\begin{aligned} \rho_{\text{stock}} &= \text{Tr}_{\mathcal{H}_{\text{time}}} (|\text{Data}\rangle\langle\text{Data}|) \\ &= \sum_{m=1}^M |c_m|^2 |v_m\rangle_{\text{stock}}\langle v_m|_{\text{stock}}, \end{aligned} \quad (36)$$

which is exactly the diagonalization of $\rho_{\text{stock}} = C$. This equation tells us that the eigenvalue of the correlation matrix C is now found to be $\lambda_j = |c_j|^2$ for all $j = 1, \dots, M = U$, and thus we end up with the expression

$$S = - \sum_{m=1}^M |c_m|^2 \log |c_m|^2. \quad (37)$$

This coincides with the entanglement entropy between $\mathcal{H}_{\text{stock}}$ and $\mathcal{H}_{\text{time}}$, i.e., von Neumann entropy of ρ_{stock} , $S = -\text{Tr}(\rho_{\text{stock}} \log \rho_{\text{stock}})$.

Note now that we cannot efficiently extract the values of $|c_m|^2$ from the state $|\text{Data}\rangle$, because $\{|v_m\rangle_{\text{stock}}\}_{m=1}^M$ and $\{|v'_m\rangle_{\text{time}}\}_{m=1}^M$ are not the computational basis. Thus, as the next step, we need to transform the basis $\{|v_m\rangle_{\text{stock}}\}_{m=1}^M$ and $\{|v'_m\rangle_{\text{time}}\}_{m=1}^M$ to the computational basis, which is done by using qSVD [41].

The qSVD is a variational algorithm for finding the transformation that transforms Schmidt basis to the computational basis. For simplicity, let us assume $n_s = n_t$, which is the case in our numerical demonstration in Sec. III D. Let $|\text{Data}\rangle$ be the output of the AAE circuit, which approximates the target $|\text{Data}\rangle$. We train PQC's $U_1(\xi)$ and $U_2(\xi')$ with parameters ξ and ξ' , so that, ideally, they realize

$$U_1(\xi) \otimes U_2(\xi') |\widetilde{\text{Data}}\rangle = \sum_{m=1}^M c_m |\bar{m}\rangle_{\text{stock}} |\bar{m}\rangle_{\text{time}}. \quad (38)$$

Here $\{|\bar{m}\rangle_{\text{stock}}\}_{m=1}^M$ and $\{|\bar{m}\rangle_{\text{time}}\}_{m=1}^M$ are subset of the computational basis states, which thus satisfy $\langle \bar{m} | \bar{\ell} \rangle = \delta_{m,\ell}$ (here we omit the subscript ‘‘stock’’ or ‘‘time’’ for simplicity). Clearly, then, the Schmidt basis is identified as $|v_m\rangle_{\text{stock}} = U_1^\dagger(\xi) |\bar{m}\rangle_{\text{stock}}$ and $|v'_m\rangle_{\text{time}} = U_2^\dagger(\xi') |\bar{m}\rangle_{\text{time}}$. The training policy is chosen so that $U_1(\xi) \otimes U_2(\xi') |\widetilde{\text{Data}}\rangle$ is as close to the Schmidt form in the computational basis as possible. The cost function to be minimized, proposed in [41], is the sum of Hamming distances between the stock bit sequence and the time bit sequence, obtained as the result of

computational-basis measurement on $\mathcal{H}_{\text{stock}}$ and $\mathcal{H}_{\text{time}}$; actually, if we measure the right-hand side of Eq. (38), the outcomes are perfectly correlated, e.g., 010 on $\mathcal{H}_{\text{stock}}$ and 010 on $\mathcal{H}_{\text{time}}$. The cost function is represented as

$$\mathcal{L}_{\text{SVD}}(\xi, \xi') = \sum_{q=1}^{n_s} \frac{1 - \langle \sigma_z^q \sigma_z^{q+n_s} \rangle}{2}, \quad (39)$$

where the expectation $\langle \cdot \rangle$ is taken over $U_1(\xi) \otimes U_2(\xi') |\widetilde{\text{Data}}\rangle$. The operator σ_z^q is the Pauli Z operator that acts on the q th qubit. We see that $\mathcal{L}_{\text{SVD}}(\xi, \xi') = 0$ holds, if and only if $U_1(\xi) \otimes U_2(\xi') |\widetilde{\text{Data}}\rangle$ takes the form of the right-hand side of Eq. (38). Therefore, by training $U_1(\xi)$ and $U_2(\xi')$ so that $\mathcal{L}_{\text{SVD}}(\xi, \xi')$ is minimized, we obtain the state that best approximates the Schmidt decomposed state.

Lastly, we gain the information on the amplitude of the output of qSVD circuit (i.e., the values approximating $|c_m|^2$), via the computational basis measurements, and then compute the SVD entropy S . For example, we take the method proposed in [89], which effectively estimates S from the state $\sum_{m=1}^M c_m |\bar{m}\rangle_{\text{stock}} |\bar{m}\rangle_{\text{time}}$; more specifically, this algorithm utilizes the amplitude estimation [90] to estimate S with complexity $\tilde{O}(\sqrt{\min(N_s, T)}/\epsilon^2)$, where ϵ is the estimation error and the \tilde{O} hides the polylog factor. The computational complexity of estimating S is negligible if the quantum state after the qSVD is sparse, or when T is small. This is indeed the case in our problem for computing the SVD entropy in the financial example, because in practice only a few large eigenvalues are associated with the market sectors and carry important information, especially in an abnormal period [76–79]. This situation is well suited to the spirit of the qSVD algorithm, which aims to estimate only large eigenvalues. Hence, taking those fact into consideration, we may be able to reduce the complexity for estimating the value of the SVD entropy.

C. Complexity of the algorithm

The complexity for computing the SVD entropy can be obtained by setting $N = N_s T$, $N_{\text{alg}} = O(\text{poly}(\log N_s T))$, and $N_{\text{mes}} = \tilde{O}(\sqrt{\min(N_s, T)}/\epsilon^2)$ in Table I, where the data is dense. As noted in Sec. II C, even though we need $O(N_s T \log N_s T)$ computation in a classical computing device, the exact data loading method also requires the same amount of classical computation for compiling the data into gate operations. On the other hand, for the execution stage, AAE requires $O(\text{poly}[\log(N_s T)]) \tilde{O}(\sqrt{\min(N_s, T)}/\epsilon^2)$ gate operations in a quantum computing device. In contrast, for exactly encoding the data we need $O(N_s T)$ gates (say, with the technique in [15,24]), and the total gate operations on a quantum computing device is $O(N_s T) \tilde{O}(\sqrt{\min(N_s, T)}/\epsilon^2)$, which is much larger than the one using the AAE method.

D. Demonstration

Here we give a numerical demonstration to show the performance of our algorithm composed of AAE and qSVD, in the problem of computing the SVD entropy for the following stock data found in the Dow Jones Industrial Average at the end of 2008: Exxon Mobil Corporation (XOM), Walmart

TABLE II. Stock prices for Exxon Mobil Corporation (XOM), Walmart (WMT), Procter & Gamble (PG), and Microsoft (MSFT) between April 2008 and March 2009.

Symbol	Apr 08	May 08	Jun 08	Jul 08	Aug 08	Sep 08	Oct 08	Nov 08	Dec 08	Jan 09	Feb 09	Mar 09
XOM	84.80	90.10	88.09	87.87	80.55	78.04	77.19	73.45	77.89	80.06	76.06	67.00
WMT	53.19	58.20	57.41	56.00	58.75	59.90	59.51	56.76	55.37	55.98	46.57	48.81
PG	70.41	67.03	65.92	60.55	65.73	70.35	69.34	64.72	63.73	61.69	54.00	47.32
MSFT	28.83	28.50	28.24	27.27	25.92	27.67	26.38	22.48	19.88	19.53	17.03	15.96

(WMT), Procter & Gamble (PG), and Microsoft (MSFT). They are top four stocks included in Dow Jones Industrial Average by market capitalization at the end of 2008. For each stock, we use the one-year monthly data from April 2008 to March 2009, which is shown in Table II. Data were taken from Yahoo Finance (in every month, the opening price is used). Figure 3 shows the logarithmic rate of return (27) for each stock at every month computed with the data in Table II.

The goal is to compute the SVD entropy at each term, with the length $T = 5$ months. For example, we compute the SVD entropy at August 2008, using the data from April 2008 to August 2008. The stock indices $j = 1, 2, 3, 4$ correspond to XOM, WMT, PG, and MSFT, respectively. Also, the time indices $t = 0, 1, 2, 3, 4$ identify the month in which the SVD entropy is computed; for instance, the SVD entropy in August 2008 is computed, using the data of April 2008 ($t = 0$), May 2008 ($t = 1$), June 2008 ($t = 2$), July 2008 ($t = 3$), and August 2008 ($t = 4$). As a result, s_{jt} has totally $20 = 4$ (stocks) \times 5 (terms) components, and thus, from Eq. (27), both r_{jt} and a_{jt} have $16 = 4 \times 4$ components, where the indices run over $j = 1, 2, 3, 4$ and $t = 1, 2, 3, 4$; that is, $\mathcal{H}_{\text{stock}} \otimes \mathcal{H}_{\text{time}} = \mathbf{C}^4 \otimes \mathbf{C}^4$. Note that $\{a_{jt}\}$ contain both positive and negative quantities, and thus AAE algorithm for Case 2 is used for the data loading. Hence, we need an additional ancilla qubit, meaning that the total number of qubit is 5. The extended target state (7) is now given by

$$|\bar{\psi}\rangle = \sum_{k=0}^{31} \bar{\psi}_k |k\rangle, \tag{40}$$

where

$$\bar{\psi}_k = \begin{cases} a_{jt} & \text{if } k = 8(j-1) + 2(t-1), & a_{jt} \geq 0 \\ 0 & \text{if } k = 8(j-1) + 2(t-1), & a_{jt} < 0 \\ -a_{jt} & \text{if } k = 8(j-1) + 2(t-1) + 1, & a_{jt} < 0 \\ 0 & \text{if } k = 8(j-1) + 2(t-1) + 1, & a_{jt} \geq 0. \end{cases}$$

The binary representation of k corresponds to the state of the qubits, e.g., $|2\rangle \equiv |00010\rangle$. Then the conditions of perfect data loading, given by Eqs. (8) and (9), are represented as

$$\begin{aligned} |\langle k|U(\theta)|0\rangle|^{\otimes 5}|^2 &= \bar{\psi}_k^2, \\ |\langle k|H^{\otimes 5}U(\theta)|0\rangle|^{\otimes 5}|^2 &= \left(\sum_{\ell=0}^{31} \bar{\psi}_\ell \langle \ell|H^{\otimes 5}|k\rangle \right)^2. \end{aligned} \tag{41}$$

The right-hand side of these equations are the target probability distributions to be approximated by the output probability distributions (left-hand side) of the trained PQC $U(\theta)$; that is, $q_\theta(k) = |\langle k|U(\theta)|0\rangle|^{\otimes 5}|^2$, $q_\theta^H(k) = |\langle k|H^{\otimes 5}U(\theta)|0\rangle|^{\otimes 5}|^2$, $p(k) = \bar{\psi}_k^2$, and $p^H(k) = (\sum_{\ell=0}^{31} \bar{\psi}_\ell \langle \ell|H^{\otimes 5}|k\rangle)^2$.

In this work, we execute the AAE algorithm as follows. The PQC is the eight-layer ansatz $U(\theta)$ illustrated in Fig. 1. Each layer is composed of the set of parameterized single-qubit rotational gate $R_y(\theta_r) = \exp(-i\theta_r\sigma_y/2)$ and CNOT gates that connect adjacent qubits; θ_r is the r th parameter and σ_y is the Pauli Y operator (hence $U(\theta)$ is a real matrix). We randomly initialize all θ_r at the beginning of each training. As the kernel function, $\kappa(x, y) = \exp[-(x - y)^2/0.25]$ is used. To compute the r th gradient of \mathcal{L} given in Eq. (20), we generate

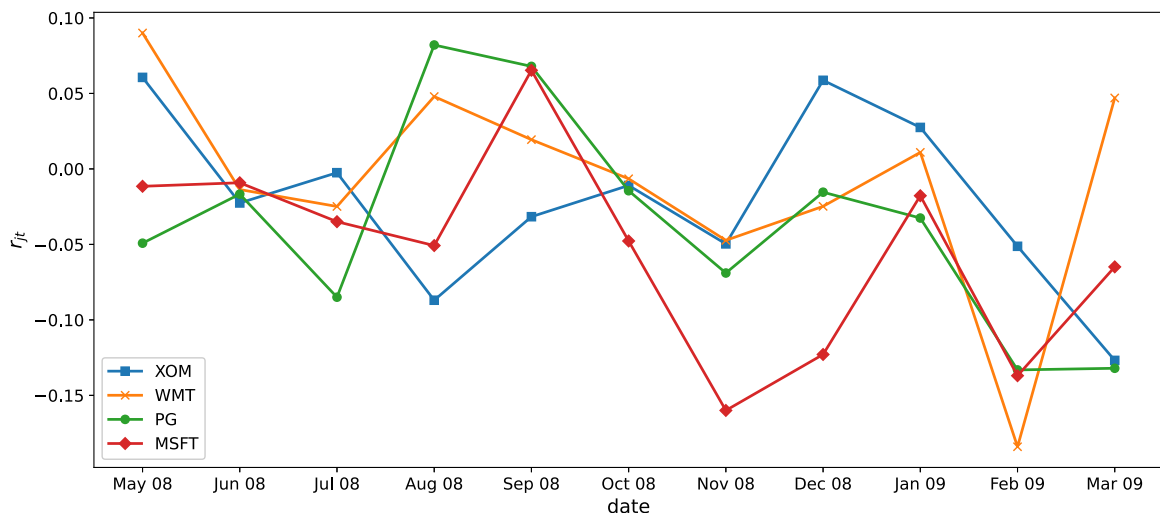


FIG. 3. Logarithmic rate of return (r_{jt}) for each stock and each moment that is computed with the data in Table II.

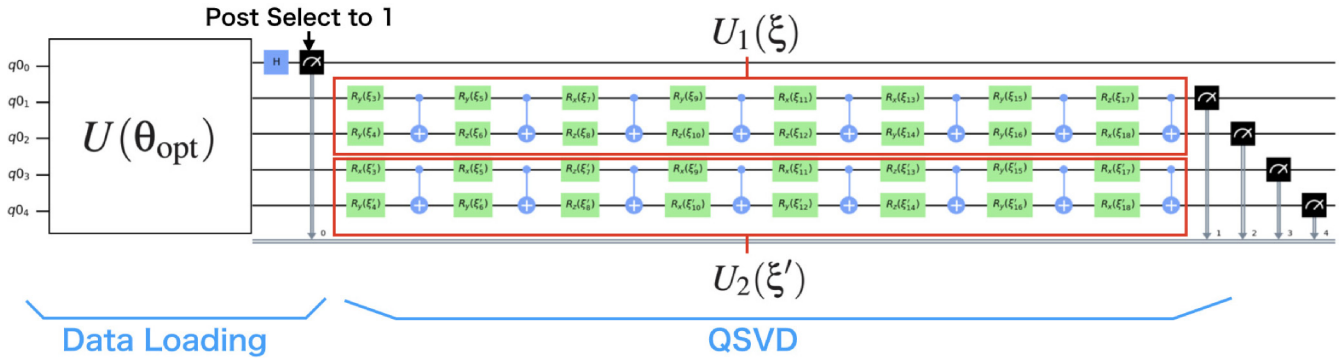


FIG. 4. Structure of the qSVD circuit. The parameters θ_{opt} in the AAE circuit are fixed. Each layer of U_1 is composed of parameterized single-qubit rotational gates $\exp(-i\xi_r\sigma_{a_r}/2)$ and CNOT gates that connect adjacent qubits, where ξ_r is the r th parameter and σ_{a_r} is the Pauli operator ($a_r = x, y, z$). The circuit U_2 has the same structure as U_1 . For each trial, we randomly initialize the gate types and parameters, e.g., as for U_1 , we choose the gate types σ_{a_r} ($a_r = x, y, z$) at the beginning of each trial and fix them during training, and we initialize parameters ξ_r . We initialize U_2 in the same way.

400 samples for each $q_{\theta_r}^+$, $q_{\theta_r}^-$, $q_{\theta_r}^{H+}$, and $q_{\theta_r}^{H-}$. As the optimizer, Adam [91] is used; the learning rate is 0.1 for the first 100 iterations and 0.01 for the other iterations. The number of iterations (i.e., the number of the updates of the parameters) is set to 200 for training $U(\theta)$. We performed 10 trials for training $U(\theta)$ and then chose the one which best minimizes the cost \mathcal{L} at the final iteration step.

Suppose that the above AAE algorithm generated the quantum state |Data>, which approximates Eq. (33). Then the next step is to apply the qSVD circuit to |Data> and then compute the SVD entropy. The PQC $U_1(\xi)$ and $U_2(\xi')$, which respectively act on the stock state $|j\rangle$ and the time state $|t\rangle$, are set to two-qubit eight-layer ansatz illustrated in Fig. 4. Each layer of U_1 is composed of parameterized single-qubit rotational gates $\exp(-i\xi_r\sigma_{a_r}/2)$ and CNOT gates that connect adjacent qubits, where ξ_r is the r th parameter and σ_{a_r} is the Pauli operator ($a_r = x, y, z$). As seen in the figure, U_2 has the same structure as U_1 . For each trial, we randomly initialize the

gate types and parameters, e.g., as for U_1 , we choose the gate types σ_{a_r} ($a_r = x, y, z$) at the beginning of each trial and fix them during the training, and we initialize parameters ξ_r . We initialize U_2 in the same way. Also we used Adam optimizer, with learning rate 0.01. For simulating the quantum circuit, we used Qiskit [92]. To focus on the net approximation error that stems from qSVD algorithm, we assume that the gradient of \mathcal{L}_{SVD} can be exactly computed (equivalently, infinite number of measurements are performed to compute this quantity). The number of iterations for training $U_1(\xi) \otimes U_2(\xi')|Data\rangle$ is 500. Unlike the case of AAE, we performed qSVD only once, to determine the optimal parameter set $(\xi_{\text{opt}}, \xi'_{\text{opt}})$. Finally, we compute the SVD entropy based on the amplitude of the final state $U_1(\xi_{\text{opt}}) \otimes U_2(\xi'_{\text{opt}})|Data\rangle$, under the assumption that the ideal quantum state tomography can be executed.

The SVD entropy in each term, computed through AAE and qSVD algorithms, is shown by the orange line with square dots in Fig. 5. As a reference, the exact value of SVD

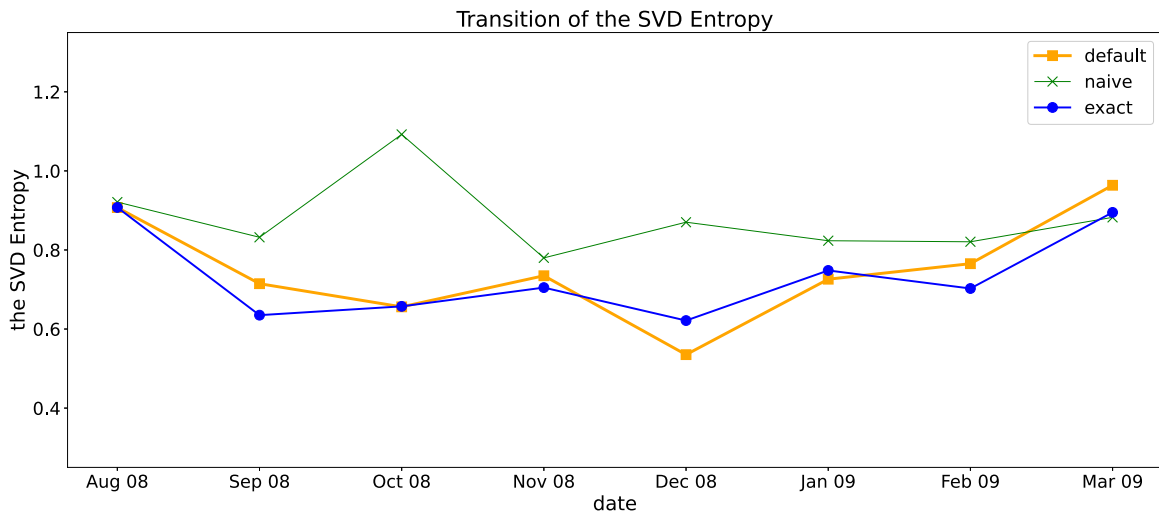


FIG. 5. Change of the SVD entropy for each term, with different computing method. The SVD entropy computed via AAE and qSVD algorithms, is shown by the orange line with square dots. The exact value of SVD entropy, computed by diagonalizing the correlation matrix, is shown by the blue line with circle dots. The SVD entropy computed with the method [40] is shown by the green line with cross marks.

TABLE III. The mean value and the maximum value of the overlap $\mathcal{O} = |\langle 1 | \langle \text{Data} | VU(\theta) | 0 \rangle^{\otimes 5} |$ for each term, depending on the values of the cost functions \mathcal{L}_1 and \mathcal{L}_2 . The number of trials that satisfy each condition out of 10 trials is also listed.

Term	$\mathcal{L}_1 < 0.01$ and $\mathcal{L}_2 < 0.01$ at the final iteration			Otherwise: $\mathcal{L}_1 \geq 0.01$ or $\mathcal{L}_2 \geq 0.01$ at the final iteration		
	No. of trials satisfying the condition	\mathcal{O}		No. of trials satisfying the condition	\mathcal{O}	
		Mean	Max		Mean	Max
Apr 08–Aug 08	2	0.977	0.981	8	0.591	0.851
May 08–Sep 08	4	0.948	0.973	6	0.435	0.646
Jun 08–Oct 08	4	0.960	0.977	6	0.284	0.718
Jul 08–Nov 08	3	0.973	0.981	7	0.439	0.875
Aug 08–Dec 08	2	0.968	0.972	8	0.437	0.648
Sep 08–Jan 08	3	0.955	0.968	7	0.203	0.441
Oct 08–Feb 08	7	0.957	0.980	3	0.578	0.829
Nov 08–Mar 08	7	0.969	0.979	3	0.613	0.655

entropy, computed by diagonalizing the correlation matrix, is shown by the blue line with circle dots. Also, to see a distinguishing property of AAE, we study the naive data-loading method [40] that trains the PQC $U_{\text{naive}}(\theta)$ so that it learns only the absolute value of the data by minimizing the cost function $\mathcal{L}_{\text{MMD}}(|\langle j | \langle t | U_{\text{naive}}(\theta) | 0 \rangle^{\otimes 4}|^2, a_{jt}^2)$; namely, $U_{\text{naive}}(\theta)$ loads the data so that the absolute values of the amplitudes of $U_{\text{naive}}(\theta) | 0 \rangle^{\otimes 4}$ is close to $|a_{jt}|$ yet without taking into account the signs. The resulting value of SVD entropy computed with this naive method is shown by the green line with cross marks. Importantly, the SVD entropies computed with our AAE algorithm well approximate the exact values, while the naive method poorly works at some point of term. Note that the estimation errors in the case of AAE is within the acceptable range for application, because the SVD entropy usually fluctuates by several percent during the normal period, while it can change drastically by a few tens of percent at around financial events [42,93–95].

Now, to see the quality of the data-loading circuit for each trial in detail, we compute the overlap between the target state $|\text{Data}\rangle$ and the generated state $VU(\theta) | 0 \rangle^{\otimes 5}$ after each training (10 trials for each term). The overlap can be measured by using the value

$$\mathcal{O} \equiv |\langle 1 | \langle \text{Data} | VU(\theta) | 0 \rangle^{\otimes 5} | \quad (42)$$

at the final iteration of each trial. In fact, in terms of \mathcal{O} , the generated state can be represented as

$$VU(\theta) | 0 \rangle = (\mathcal{O} |\text{Data}\rangle + \sqrt{1 - \mathcal{O}^2} |\text{Data}^\perp\rangle) | 1 \rangle, \quad (43)$$

where $|\text{Data}^\perp\rangle$ is a state that is orthogonal to $|\text{Data}\rangle$. Namely, the closer the value of \mathcal{O} is to 1, the more accurately $VU(\theta)$ generates $|\text{Data}\rangle$. To evaluate the statistics of the overlap in each trial, we divide the 10 trials for each term into the following two patterns of conditions satisfied by the cost function at the final iteration step: $\mathcal{L}_1, \mathcal{L}_2 < 0.01$ or otherwise, where we simply denote

$$\mathcal{L}_1 = \mathcal{L}_{\text{MMD}}(q_\theta, p), \quad \mathcal{L}_2 = \mathcal{L}_{\text{MMD}}(q_\theta^H, p^H).$$

Recall that $(q_\theta, p, q_\theta^H, p^H)$ are given below Eq. (41). In Table III we show the mean value and the maximum value of the overlap \mathcal{O} for each pattern and for each term. The number of trials that satisfy each condition out of 10 trials is also listed in the table. We then find that, as long as the condition $\mathcal{L}_1, \mathcal{L}_2 < 0.01$ is satisfied, the mean of \mathcal{O} is larger than 0.94, and there are at least two out of 10 trials that satisfy this condition. Also, the maximum value of \mathcal{O} is larger than 0.96 in all terms; such large overlaps between the target state and the generated state will lead to a successful computation of the SVD entropy for each term. When $\mathcal{L}_1 \geq 0.01$ or $\mathcal{L}_2 \geq 0.01$, on the other hand, \mathcal{O} takes a relatively small value; in this case the subsequent qSVD algorithm may yield an imprecise value of SVD entropy, hence this trial should be discarded. A notable point here is that the success probability is relatively high; a thorough examination for a larger system is an important future work.

In Fig. 6 we show an example of set of the training results of $q_\theta(k)$ and $q_\theta^H(k)$ for four terms Apr 08–Aug 08, May 08–Sep 08, Jun 08–Oct 08, and Jul 08–Nov 08 (green

TABLE IV. Stock prices for Exxon Mobil Corporation (XOM), Walmart (WMT), Procter & Gamble (PG), Microsoft (MSFT), General Electronic (GE), AT&T (T), Johnson & Johnson (JNJ), and Chevron (CVX) between April 2008 and March 2009.

Symbol	Apr 08	May 08	Jun 08	Jul 08	Aug 08	Sep 08	Oct 08	Nov 08	Dec 08	Jan 09	Feb 09	Mar 09
XOM	84.80	90.10	88.09	87.87	80.55	78.04	77.19	73.45	77.89	80.06	76.06	67.00
WMT	53.19	58.20	57.41	56.00	58.75	59.90	59.51	56.76	55.37	55.98	46.57	48.81
PG	70.41	67.03	65.92	60.55	65.73	70.35	69.34	64.72	63.73	61.69	54.00	47.32
MSFT	28.83	28.50	28.24	27.27	25.92	27.67	26.38	22.48	19.88	19.53	17.03	15.96
GE	35.92	31.54	29.57	25.40	27.34	27.44	23.08	19.02	15.73	15.88	11.57	7.970
T	38.70	39.29	39.67	33.41	31.01	32.53	28.15	26.87	28.00	28.74	24.97	22.80
JNJ	65.13	67.13	66.55	63.75	68.50	71.09	69.07	61.49	57.66	60.13	57.25	49.03
CVX	85.08	94.86	98.82	98.26	83.98	84.49	81.51	73.44	76.50	74.23	69.52	59.37

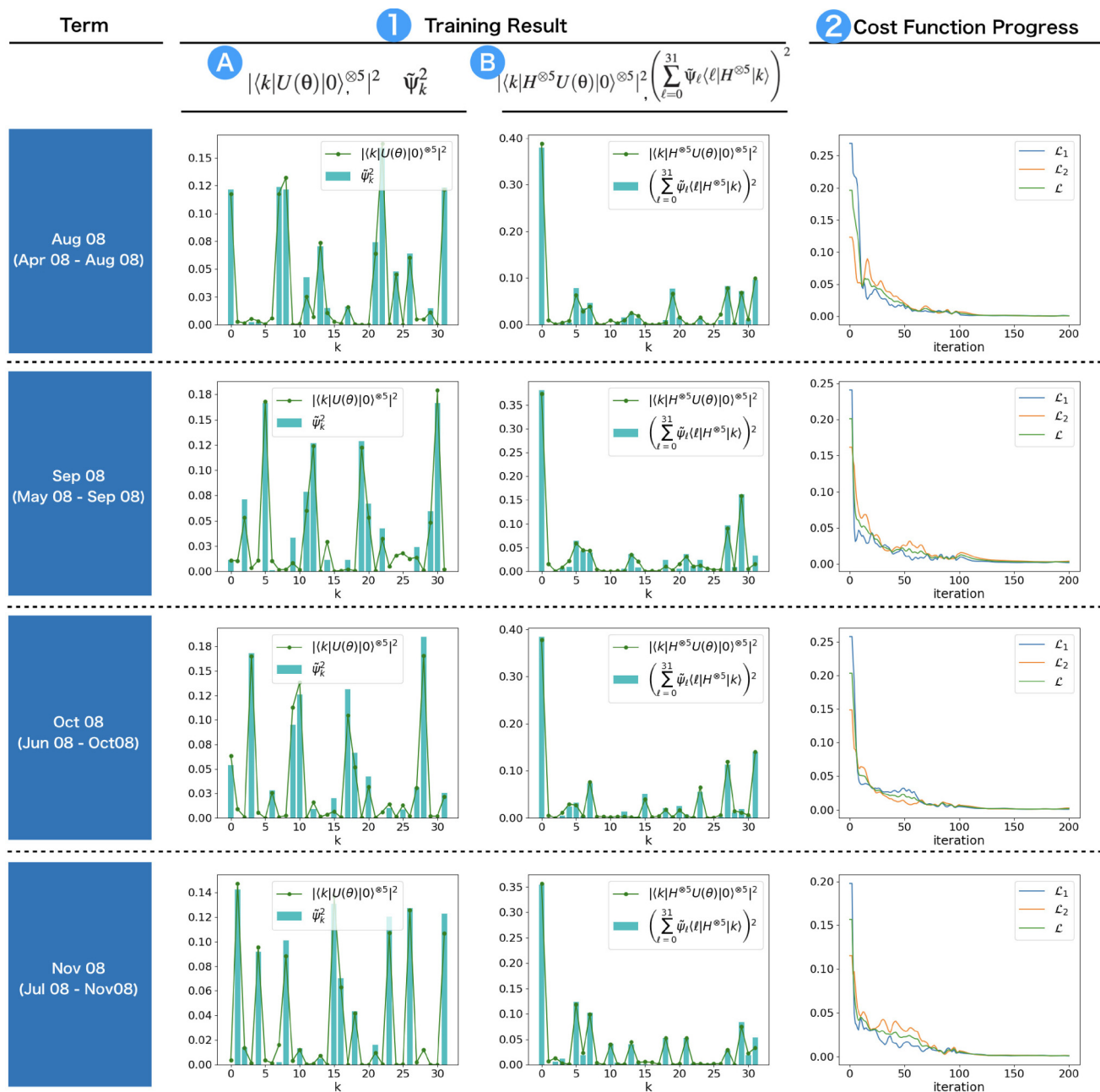


FIG. 6. (1) Example of the training results of $q_{\theta}(k) = | \langle k | U(\theta) | 0 \rangle^{\otimes 5} |^2$ and $q_{\theta}^H(k) = | \langle k | H^{\otimes 5} U(\theta) | 0 \rangle^{\otimes 5} |^2$ for each term (green lines), and the corresponding target distributions $p(k) = \tilde{\psi}_k^2$ and $p^H(k) = \left(\sum_{\ell=0}^{31} \tilde{\psi}_{\ell} | \langle \ell | H^{\otimes 5} | k \rangle \right)^2$ (green bars). These distributions are the best one in the sense that the cost function \mathcal{L} at the 200th epoch takes the smallest in 10 trials. (2) The change of the cost function \mathcal{L} in the same trial as (1) for each term. We also show the change of the cost functions for the two distributions $q_{\theta}(k)$ and $q_{\theta}^H(k)$ that contribute to \mathcal{L} .

lines); this set of distributions is the best one out of 10 trials in the sense that it minimizes the cost $\mathcal{L}_{\text{MMD}}(\theta)$ at the 200th iteration step, corresponding to the case when the overlap is maximized for each term. Also the target distributions $p(k)$ and $p^H(k)$ are illustrated with green bars. The right column of Fig. 6 plots the change of the costs \mathcal{L}_1 , \mathcal{L}_2 , and $\mathcal{L} = (\mathcal{L}_1 + \mathcal{L}_2)/2$ for each term. These results confirm that the AAE algorithm realizes near perfect data loading; that is, the resulting model distributions q_{θ} and q_{θ}^H well approximate the target distributions p and p^H , respectively, which eventually leads to the successful computation of SVD entropy as discussed above.

Here we point out the interesting feature of the SVD entropy, which can be observed from Figs. 3 and 5. Figure 3 shows that, until August 2008, the stocks did not strongly correlate with each other, which leads to the relatively large value of SVD entropy (~ 0.9) as seen in Fig. 5. In September 2008, the Lehman Brothers bankruptcy ignited the global financial crisis. As a result, from October 2008 to February 2009, the stocks became strongly correlated with each other. In such a case, many of stocks cooperatively moved as seen in Fig. 3. This strong correlation led to the small SVD entropy (~ 0.7) from October to February, which is an evidence of the financial crisis. In March 2009, Fig. 5 shows that the SVD

entropy again takes relatively large value (~ 0.9), indicating that the market returned to normal and each stock moved differently. Interestingly, according to the S&P index, it is argued that the financial crisis ended in March 2009 (e.g., see [96]), which is consistent to the result of SVD entropy. We would like to emphasize that AAE algorithm correctly computed the SVD entropy and enables us to capture the above-mentioned financial trends.

Lastly, it is surely important to assess the performance of AAE for other example problems with different size and data set. Appendix D gives such a demonstration where the number of stocks is eight.

IV. CONCLUSIONS

This paper provides the Approximate Amplitude Encoding (AAE) algorithm that effectively loads a given classical data into a shallow parameterized quantum circuit. The point of the AAE algorithm is in the formulation of a valid cost function composed of two types of maximum mean discrepancy measures, based on the perfect encoding condition (Theorem 1 for Case 1 and Theorem 2 for Case 2); training of the circuit is executed by minimizing this cost function, which enables encoding the signs of the data components unlike the previous proposal (that can only load the absolute values). We also provide an algorithm composed of AAE and the existing quantum singular value decomposition (qSVD) algorithm, for computing the SVD entropy in the stock market. A thorough numerical study was performed, showing that the approximation error of AAE was found to be sufficiently small in this case and, as a result, the subsequent qSVD algorithm yields a good approximation solution.

To show that the proposed AAE algorithm will be practically useful to implement various quantum algorithms that need classical data loading, it is important to examine a larger system, e.g., a 20-qubit problem with a 20-layer ansatz. In fact in this case the number of parameters is 400, while the degree of freedom of the state vector is $2^{20} \approx 1\,000\,000$, reflecting that the polynomial-size circuit could deal with an exponential-size problem. However, even in this potentially classically doable size setting, there are several practical problems to be resolved. For instance, we expect that the gradient vanishing issue will arise, which needs careful application of several (existing) methods such as circuit initialization [58], special structured ansatz [45], and parameter embedding [59]. Moreover, recently we find some approaches for approximating a large circuit with set of small circuits [66,67]; these methods are worth investigating to address the scalability of our method. At the same time, a notable point of the problem of calculating the SVD entropy is that it does not require a very precise calculation but only a global trend over a certain time period. Hence we need to carefully determine the number of layers as well as the iteration steps of the variational algorithm to have necessary precision; in particular the former might be further reduced using existing techniques, e.g., [62–65]. With these elaboration, furthermore, we are also interested in testing the algorithm with a real quantum computing device. Overall, these additional tasks are all important and yet not straightforward, so we will study this problem as a separate work.

ACKNOWLEDGMENTS

This work was supported by Grant-in-Aid for JSPS Research Fellow Grant No. 22J01501, and MEXT Quantum Leap Flagship Program Grants No. JPMXS0118067285 and No. JPMXS0120319794.

APPENDIX A: PROOF OF THEOREM 1

Theorem 1. In Case 1, if (4) and (5) are exactly satisfied, $U(\boldsymbol{\theta})|0\rangle = \sum_j \mathbf{d}_j |j\rangle$ or $U(\boldsymbol{\theta})|0\rangle = -\sum_j \mathbf{d}_j |j\rangle$.

Proof. Let us denote \mathbf{a}_j by $\langle j|U(\boldsymbol{\theta})|0\rangle$. Then (4) and (5) are rewritten as

$$\mathbf{a}_j^2 = \mathbf{d}_j^2 \quad (\forall j), \quad (\text{A1})$$

$$\left(\sum_{k=0}^{N-1} H_{jk}^{\otimes n} \mathbf{a}_k \right)^2 = \left(\sum_{k=0}^{N-1} H_{jk}^{\otimes n} \mathbf{d}_k \right)^2 \quad (\forall j), \quad (\text{A2})$$

where $H_{jk}^{\otimes n} \equiv \langle j|H^{\otimes n}|k\rangle$. For $j = 0$, the left-hand side of (A2) becomes

$$\left(\sum_{k=0}^{N-1} H_{0k}^{\otimes n} \mathbf{a}_k \right)^2 = \frac{1}{2^n} \left(\sum_{k=0}^{N-1} \mathbf{a}_k \right)^2 \leq \frac{1}{2^n} \left(\sum_{k=0}^{N-1} |\mathbf{a}_k| \right)^2, \quad (\text{A3})$$

where the equality holds only when $\mathbf{a}_k \geq 0$ ($\forall k$) or $\mathbf{a}_k \leq 0$ ($\forall k$). The equality condition is equivalent to $\mathbf{a} = \mathbf{d}$ or $\mathbf{a} = -\mathbf{d}$, because of (A1) and the condition of Case 1: $\mathbf{d}_j \geq 0$ ($\forall j$) or $\mathbf{d}_j \leq 0$ ($\forall j$). Conversely, if the equality condition is not satisfied, we find that

$$\begin{aligned} \frac{1}{2^n} \left(\sum_{k=0}^{N-1} \mathbf{a}_k \right)^2 &< \frac{1}{2^n} \left(\sum_{k=0}^{N-1} |\mathbf{a}_k| \right)^2 \\ &= \frac{1}{2^n} \left(\sum_{k=0}^{N-1} \mathbf{d}_k \right)^2 = \left(\sum_{k=0}^{N-1} H_{0k}^{\otimes n} \mathbf{d}_k \right)^2, \end{aligned} \quad (\text{A4})$$

which contradicts to (A2) for $j = 0$. Thus, the equality condition of (A3) is satisfied, i.e., $\mathbf{a} = \mathbf{d}$ or $\mathbf{a} = -\mathbf{d}$.

APPENDIX B: AMPLIFICATION OF THE SUCCESS PROBABILITY IN CASE 2

In Case 2, the encoding can be carried out with success probability 1/2 in the ideal case [i.e., the case where Eqs. (8) and (9) are exactly satisfied]. But by applying the amplitude amplification operation [90], we obtain |Data⁺⟩ with success probability 1, instead of 1/2, although more gates to implement this extra operation are required. The method is described as follows. By adding another qubit, it holds

$$\begin{aligned} &\pm (I_n \otimes H)U(\boldsymbol{\theta})|0\rangle^{\otimes n+1}H|0\rangle \\ &= \frac{|\text{Data}^+\rangle - |\text{Data}^-\rangle}{2}|00\rangle + \frac{|\text{Data}^+\rangle - |\text{Data}^-\rangle}{2}|01\rangle \\ &\quad + \frac{|\text{Data}^+\rangle + |\text{Data}^-\rangle}{2}|10\rangle + \frac{|\text{Data}^+\rangle + |\text{Data}^-\rangle}{2}|11\rangle. \end{aligned} \quad (\text{B1})$$

Note that $\mathbf{d}_\uparrow, \mathbf{d}_\downarrow \neq 0$ and

$$\mathbf{d} = Q \begin{pmatrix} \mathbf{d}_\uparrow \\ \mathbf{d}_\downarrow \end{pmatrix}. \quad (\text{C10})$$

Substituting (C8) and (C9) into (C7), we get

$$P\mathbf{c} = \begin{pmatrix} X_{\uparrow\uparrow}\mathbf{d}_\uparrow \\ X_{\downarrow\downarrow}\mathbf{d}_\downarrow \end{pmatrix}, \quad (\text{C11})$$

and therefore,

$$\mathbf{c} = P \begin{pmatrix} X_{\uparrow\uparrow}\mathbf{d}_\uparrow \\ X_{\downarrow\downarrow}\mathbf{d}_\downarrow \end{pmatrix} \quad (\text{C12})$$

holds because $P^2 = I$. If we set

$$\mathbf{d}' = Q \begin{pmatrix} \mathbf{d}_\uparrow \\ -\mathbf{d}_\downarrow \end{pmatrix}, \quad (\text{C13})$$

then

$$\mathbf{c}' = X\mathbf{d}' = P(PXQ) \begin{pmatrix} \mathbf{d}_\uparrow \\ -\mathbf{d}_\downarrow \end{pmatrix} = P \begin{pmatrix} X_{\uparrow\uparrow}\mathbf{d}_\uparrow \\ -X_{\downarrow\downarrow}\mathbf{d}_\downarrow \end{pmatrix}. \quad (\text{C14})$$

Because P, Q are matrices that interchange rows, comparing (C13) and (C10) and (C14) and (C12), we see that

$$\mathbf{d}'_j{}^2 = \mathbf{d}_j{}^2, \quad \mathbf{c}'_j{}^2 = \mathbf{c}_j{}^2, \quad \mathbf{d}' \neq \mathbf{d}, -\mathbf{d}, \quad (\text{C15})$$

which concludes the proof of the sufficient condition of the theorem.

Next, we prove the necessity condition of the theorem. If the necessity assumption holds, there exist \mathbf{d}' and \mathbf{c}' that satisfy $\mathbf{c}' = X\mathbf{d}'$ and (C5). We set

$$\mathbf{c}^+ = \frac{\mathbf{c} + \mathbf{c}'}{2}, \quad \mathbf{c}^- = \frac{\mathbf{c} - \mathbf{c}'}{2}. \quad (\text{C16})$$

Then for each element of $\mathbf{c}^-, \mathbf{c}^+$, it holds that

$$\mathbf{c}_j^+ = \begin{cases} \mathbf{c}_j & (\text{if } \mathbf{c}_j = \mathbf{c}'_j) \\ 0 & (\text{if } \mathbf{c}_j = -\mathbf{c}'_j) \end{cases} \quad (\text{C17})$$

$$\mathbf{c}_j^- = \begin{cases} 0 & (\text{if } \mathbf{c}_j = \mathbf{c}'_j) \\ \mathbf{c}_j & (\text{if } \mathbf{c}_j = -\mathbf{c}'_j) \end{cases}.$$

We see that \mathbf{c}_j^- is nonzero only if \mathbf{c}_j^+ is zero, and vice versa. Therefore, by using a row swap matrix $P \in \mathcal{A}(N)$, \mathbf{c}_i^- and \mathbf{c}_i^+ can be transformed as

$$P\mathbf{c}^+ = \begin{pmatrix} \mathbf{c}^+ \\ \mathbf{0} \end{pmatrix}, \quad P\mathbf{c}^- = \begin{pmatrix} \mathbf{0} \\ \mathbf{c}^- \end{pmatrix}, \quad (\text{C18})$$

where $\mathbf{dim}(\mathbf{c}^+) + \mathbf{dim}(\mathbf{c}^-) = N$. Similarly, we set

$$\mathbf{d}^+ = \frac{\mathbf{d} + \mathbf{d}'}{2}, \quad \mathbf{d}^- = \frac{\mathbf{d} - \mathbf{d}'}{2}. \quad (\text{C19})$$

Then \mathbf{d}_j^- is nonzero only if \mathbf{d}_j^+ is zero, and vice versa. Thus, by using another row swap matrix $Q \in \mathcal{A}(N)$,

$$Q\mathbf{d}^+ = \begin{pmatrix} \mathbf{d}^+ \\ \mathbf{0} \end{pmatrix}, \quad Q\mathbf{d}^- = \begin{pmatrix} \mathbf{0} \\ \mathbf{d}^- \end{pmatrix}, \quad (\text{C20})$$

where $\mathbf{dim}(\mathbf{d}^+) + \mathbf{dim}(\mathbf{d}^-) = N$. From $\mathbf{c}' = X\mathbf{d}'$ and (C6),

$$\mathbf{c}^+ = X\mathbf{d}^+. \quad (\text{C21})$$

By multiplying P from left and using $Q^2 = I$, we get

$$P\mathbf{c}^+ = PXQQ\mathbf{d}^+. \quad (\text{C22})$$

Substituting the first equality in (C18) and that in (C20) into (C22),

$$\begin{pmatrix} \mathbf{c}^+ \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} X_{\uparrow\uparrow} & X_{\uparrow\downarrow} \\ X_{\downarrow\uparrow} & X_{\downarrow\downarrow} \end{pmatrix} \begin{pmatrix} \mathbf{d}^+ \\ \mathbf{0} \end{pmatrix}, \quad (\text{C23})$$

where we split PXQ into submatrices so that the number of rows and columns of $X_{\uparrow\uparrow}$ is $\mathbf{dim}(\mathbf{d}^+)$ and $\mathbf{dim}(\mathbf{c}^+)$ respectively. Writing the equality in (C23) explicitly, we get

$$\mathbf{c}_\uparrow = X_{\uparrow\uparrow}\mathbf{d}_\uparrow, \quad (\text{C24})$$

$$\mathbf{0} = X_{\downarrow\uparrow}\mathbf{d}_\uparrow. \quad (\text{C25})$$

Similarly, we obtain

$$\begin{pmatrix} \mathbf{0} \\ \mathbf{c}_\downarrow \end{pmatrix} = \begin{pmatrix} X_{\uparrow\uparrow} & X_{\uparrow\downarrow} \\ X_{\downarrow\uparrow} & X_{\downarrow\downarrow} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \mathbf{d}_\downarrow \end{pmatrix} \quad (\text{C26})$$

and as a result,

$$\mathbf{0} = X_{\uparrow\downarrow}\mathbf{d}_\downarrow, \quad (\text{C27})$$

$$\mathbf{c}_\downarrow = X_{\downarrow\downarrow}\mathbf{d}_\downarrow. \quad (\text{C28})$$

Since

$$\begin{pmatrix} \mathbf{d}_\uparrow \\ \mathbf{d}_\downarrow \end{pmatrix} = Q\mathbf{d}, \quad PXQ = \begin{pmatrix} X_{\uparrow\uparrow} & X_{\uparrow\downarrow} \\ X_{\downarrow\uparrow} & X_{\downarrow\downarrow} \end{pmatrix}, \quad (\text{C29})$$

Eqs. (C25) and (C27) indicate that (\mathbf{d}, X) is pair-block-diagonal.

From the theorem, it seems that when a data set (hence \mathbf{d}) is given, we should find X that (\mathbf{d}, X) is *not* pair-block-diagonal; then by training $U(\theta)$ so that (C1) and (C2) with the X , the goal (3) is achieved. However, as far as our knowledge, for general \mathbf{d} , it is difficult to check if (\mathbf{d}, X) is pair-block-diagonal or not. On the other hand, if $\mathbf{d}_j \geq 0 (\forall j)$ or $\mathbf{d}_j \leq 0 (\forall j)$, we can show that $(\mathbf{d}, H^{\otimes n})$ is *not* pair-block-diagonal; although we already know the fact from the Theorem 1, we can also prove it by directly showing that the condition for pair-block-diagonal is not satisfied when $\mathbf{d}_j \geq 0 (\forall j)$ or $\mathbf{d}_j \leq 0 (\forall j)$ and $X = H^{\otimes n}$. Therefore, instead of finding X for general \mathbf{d} , we build our algorithm depending on the values of \mathbf{d} .

APPENDIX D: COMPUTATION OF THE SVD ENTROPY IN THE CASE OF EIGHT STOCKS

Here we show the SVD entropy computation in a larger size setting. In addition to XOM, WMT, PG, and MSFT, we use the stock data of General Electric (GE), AT&T (T), Johnson & Johnson (JNJ), and Chevron (CVX); they are top eight stocks included in the Dow Jones Industrial Average at the end of 2008. As in the case of Table II, we use the one-year monthly data from April 2008 to March 2009. Data were taken from Yahoo Finance (in every month, the opening price is used). We show the stock price data in Table IV.

The goal is to compute the SVD entropy at each term, with the length $T = 5$ months, which is the same as Sec. III D. The stock indices $j = 1, 2, 3, 4, 5, 6, 7, 8$ correspond to XOM, WMT, PG, MSFT, GE, T, JNJ, and CVX, respectively. Also, the time indices $t = 0, 1, 2, 3, 4$ identify the month in which the SVD entropy is computed; for instance, the SVD entropy in August 2008 is computed, using the data of April

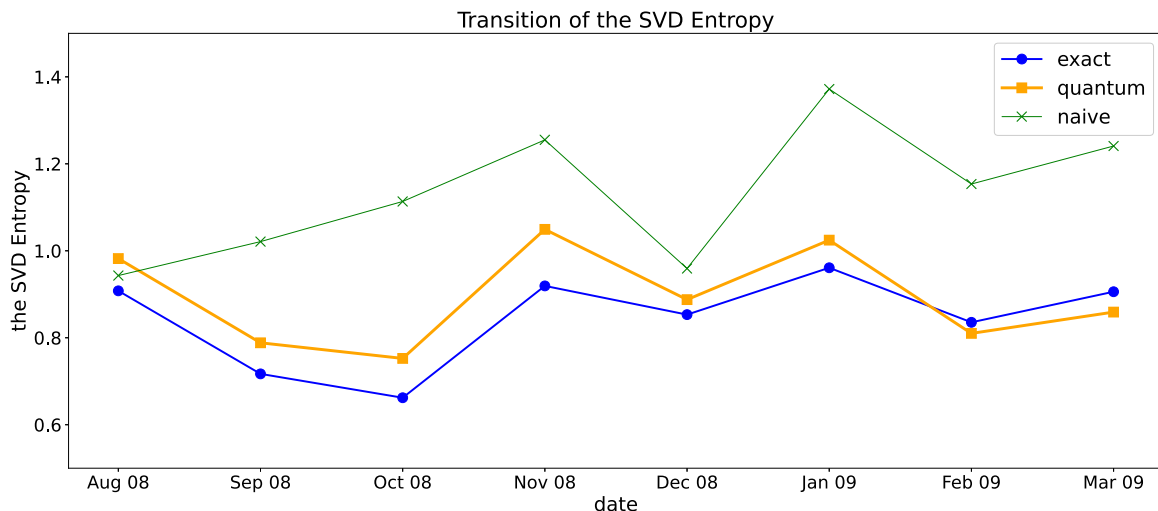


FIG. 7. Change of the SVD entropy for each term, with different computing method when using stock data in Table IV. The SVD entropy computed via AAE and qSVD algorithms, is shown by the orange line with square dots. The exact value of SVD entropy, computed by diagonalizing the correlation matrix, is shown by the blue line with circle dots. The SVD entropy computed with the naive data-loading method is shown by the green line with cross marks.

2008 ($t = 0$), May 2008 ($t = 1$), June 2008 ($t = 2$), July 2008 ($t = 3$), and August 2008 ($t = 4$). As a result, s_{jt} has a total of $40 = 8$ (stocks) \times 5 (terms) components. Thus, from Eq. (27), both r_{jt} and a_{jt} have $32 = 8 \times 4$ components, where the indices run over $j = 1, 2, 3, 4, 5, 6, 7, 8$ and $t = 1, 2, 3, 4$; that is, $\mathcal{H}_{\text{stock}} \otimes \mathcal{H}_{\text{time}} = \mathbb{C}^8 \otimes \mathbb{C}^4$. We use AAE algorithm in Case 2 for the data loading. Hence, we need an additional ancilla qubit, meaning that the total number of qubits is six.

Note that unlike the experiment in Sec. III D, n_s ($=3$) and n_t ($=2$) are different, which requires a little modification for the cost function of the qSVD algorithm. Recall that the purpose of training PQCs $U_1(\xi)$ and $U_2(\xi')$ in qSVD is finding unitary operators that transform the Schmidt basis $\{|v_m\rangle_{\text{stock}}\}_{m=1}^M$ and $\{|v'_m\rangle_{\text{time}}\}_{m=1}^M$ to the computational basis. We have freedom of the choice to which computational basis we transform the Schmidt basis, but we limit our goal of the training as $U_1(\xi)$ and $U_2(\xi')$ ideally operates as follows:

$$U_1(\xi) \otimes U_2(\xi') |\widetilde{\text{Data}}\rangle = \sum_{m=1}^M c_m (|\bar{m}\rangle|0\rangle)_{\text{stock}} |\bar{m}\rangle_{\text{time}}, \quad (\text{D1})$$

where $\{(|\bar{m}\rangle|0\rangle)_{\text{stock}}\}_{m=1}^M$ is a subset of the computational basis in $\mathcal{H}_{\text{stock}}$ with the last qubit equal to zero and $\{|\bar{m}\rangle_{\text{time}}\}_{m=1}^M$ is the subset of the computational basis in $\mathcal{H}_{\text{time}}$. The corresponding cost function is given by

$$\mathcal{L}_{\text{SVD}}(\xi, \xi') = \frac{1 - \sigma_z^3}{2} + \sum_{q=1}^2 \frac{1 - \langle \sigma_z^q \sigma_z^{q+3} \rangle}{2}, \quad (\text{D2})$$

where the expectation $\langle \cdot \rangle$ is taken over $U_1(\xi) \otimes U_2(\xi') |\widetilde{\text{Data}}\rangle$. The operator σ_z^q is the Pauli Z operator that acts on the q -th qubit. We can see that $\mathcal{L}_{\text{SVD}}(\xi, \xi') = 0$ holds, if and only if $U_1(\xi) \otimes U_2(\xi') |\widetilde{\text{Data}}\rangle$ takes the form of the right hand side of Eq. (D1). Therefore, by training $U_1(\xi)$ and $U_2(\xi')$ so that $\mathcal{L}_{\text{SVD}}(\xi, \xi')$ is minimized, we obtain the state that best approximates the Schmidt decomposed state.

The settings of AAE training is similar to the ones in Sec. III D. As the PQC $U(\theta)$ for the data loading, we use the hardware-efficient ansatz with six qubits. The composition of each layer, the way of initialization, the kernel function, the optimizer, and the number of samples for computing $q_{\theta_r}^{\pm}$ and $q_{\theta_r}^{H\pm}$ are the same as the previous case in Sec. III D. The learning rate is 0.1 for the first 100 iterations and 0.01 for the other iterations. We chose the number of layers from 12 or 13. We performed 10 trials of training $U(\theta)$ for each number of layers (20 trials in total). Note that for each trial we saved the model at the {200, 250, 300, 350, 400}-th iterations and then the model which minimizes the cost \mathcal{L} is chosen as the output of the trial. Among the outputs of each trial, the minimizer of \mathcal{L} is adopted as the data-loading circuit used for the computation of the SVD entropy in the next step. As a result, for computing the SVD entropy in August 2008, February 2009, and March 2009, 12-layer data-loading circuits are used and for computing that in September 2008, October 2008, November 2008, December 2008, and January 2009, 13-layer data-loading circuits are used.

The qSVD is performed with the cost function (D2). The PQC $U_1(\xi)$, which acts on the stock state $|j\rangle$, is set to a three-qubit, 12-layer ansatz; also $U_2(\xi')$, which acts on the time state $|t\rangle$, is set to a two-qubit, 12-layer ansatz. The composition of the circuit, the way of initialization, the optimizer, the learning rate, and the number of iterations are the same as the previous qSVD experiment. The computation of the SVD entropy is also performed in the same way as in Sec. III D.

We show the SVD entropy in each term computed through AAE and qSVD algorithms by the orange line with square dots in Fig. 7. As a reference, the exact value of SVD entropy, computed by diagonalizing the correlation matrix, is shown by the blue line with circle dots. Also, to see a distinguishing property of AAE, we study the naive data-loading method that trains the PQC $U_{\text{naive}}(\theta)$ so that it learns only the absolute value of the data by minimizing the cost function $\mathcal{L}_{\text{MMD}}(|\langle j|t|U_{\text{naive}}(\theta)|0\rangle|^2, a_{jt}^2)$. The resulting value of

SVD entropy computed with this naive method is shown by the green line with cross marks.

Similar to the results in Sec. III D, the SVD entropies computed with our AAE algorithm well approximate the exact values, while the naive method poorly works. Notably, despite the increase in the number of data, we see that the estimation errors in Fig. 7 are about the same as those in Fig. 5 (the maximum estimation error is about 10% in both figures). Also, the number of layers for the data-loading circuit in this section (=12 or 13) is smaller than twice that

in Sec. III D (=8) even though the number of data doubles and the number of qubits increases, which infers that the number of layers for AAE does not increase exponentially in conjunction with the increase of the number of qubits as expected. Still, as the number of qubits increases, the issues in the optimization discussed in Sec. II C may become severer; a larger size experiment is necessary to study how those issues affect our algorithm and how we can avoid them, which is beyond the scope of this paper and left for future work.

-
- [1] P. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (IEEE, Piscataway, NJ, 1994), pp. 124–134.
- [2] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing '96* (ACM Press, New York, NY, 1996), pp. 124–134.
- [3] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum Algorithm for Linear Systems of Equations, *Phys. Rev. Lett.* **103**, 150502 (2009).
- [4] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature (London)* **549**, 195 (2017).
- [5] S. Srinivasan, C. Downey, and B. Boots, Learning and inference in Hilbert space with quantum graphical models, in *Advances in Neural Information Processing Systems* (Curran Associates, Red Hook, NY, 2018), Vol. 31, pp. 10338–10347.
- [6] M. Schuld, I. Sinayskiy, and F. Petruccione, Prediction by linear regression on a quantum computer, *Phys. Rev. A* **94**, 022342 (2016).
- [7] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, *Nature (London)* **567**, 209 (2019).
- [8] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, Quantum machine learning: A classical perspective, *Proc. R. Soc. A* **474**, 20170551 (2018).
- [9] C. Blank, D. K. Park, J.-K. K. Rhee, and F. Petruccione, Quantum classifier with tailored quantum kernel, *npj Quantum Inf.* **6**, 41 (2020).
- [10] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Circuit-centric quantum classifiers, *Phys. Rev. A* **101**, 032308 (2020).
- [11] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum Support Vector Machine for Big Data Classification, *Phys. Rev. Lett.* **113**, 130503 (2014).
- [12] I. Kerenidis and A. Prakash, Quantum recommendation systems, [arXiv:1603.08675](https://arxiv.org/abs/1603.08675).
- [13] N. Wiebe, D. Braun, and S. Lloyd, Quantum Algorithm for Data Fitting, *Phys. Rev. Lett.* **109**, 050505 (2012).
- [14] M. Schuld, M. Fingerhuth, and F. Petruccione, Implementing a distance-based classifier with a quantum interference circuit, *Europhys. Lett.* **119**, 60002 (2017).
- [15] M. Plesch and Č. Brukner, Quantum-state preparation with universal gate decompositions, *Phys. Rev. A* **83**, 032302 (2011).
- [16] V. V. Shende, S. S. Bullock, and I. L. Markov, Synthesis of quantum-logic circuits, *IEEE Trans. Comput.-Aided Design Integrated Circuits Syst.* **25**, 1000 (2006).
- [17] L. Grover and T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions, [arXiv:quant-ph/0208112](https://arxiv.org/abs/quant-ph/0208112).
- [18] M. Möttönen, J. Vartiainen, V. Bergholm, and M. Salomaa, Transformation of quantum states using uniformly controlled rotations, *Quantum Inf. Computat.* **5**, 467 (2005).
- [19] V. Shende and I. Markov, Quantum circuits for incompletely specified two-qubit operators, *Quantum Inf. Computat.* **5**, 49 (2005).
- [20] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Elementary gates for quantum computation, *Phys. Rev. A* **52**, 3457 (1995).
- [21] J. J. Vartiainen, M. Möttönen, and M. M. Salomaa, Efficient Decomposition of Quantum Gates, *Phys. Rev. Lett.* **92**, 177902 (2004).
- [22] V. V. Shende, I. L. Markov, and S. S. Bullock, Minimal universal two-qubit controlled-not-based circuits, *Phys. Rev. A* **69**, 062321 (2004).
- [23] G.-L. Long and Y. Sun, Efficient scheme for initializing a quantum register with an arbitrary superposed state, *Phys. Rev. A* **64**, 014303 (2001).
- [24] V. Bergholm, J. J. Vartiainen, M. Mottonen, and M. M. Salomaa, Quantum circuits with uniformly controlled one-qubit gates, *Phys. Rev. A* **71**, 052330 (2005).
- [25] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl, Quantum circuits for isometries, *Phys. Rev. A* **93**, 032318 (2016).
- [26] X. Sun, G. Tian, S. Yang, P. Yuan, and S. Zhang, Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis, [arXiv:2108.06150](https://arxiv.org/abs/2108.06150).
- [27] J. Zhao, Y.-C. Wu, G.-C. Guo, and G.-P. Guo, State preparation based on quantum phase estimation, [arXiv:1912.05335](https://arxiv.org/abs/1912.05335).
- [28] G. Rosenthal, Query and depth upper bounds for quantum unitaries via Grover search, [arXiv:2111.07992](https://arxiv.org/abs/2111.07992).
- [29] X.-M. Zhang, M.-H. Yung, and X. Yuan, Low-depth quantum state preparation, *Phys. Rev. Research* **3**, 043200 (2021).
- [30] Z. Zhang, Q. Wang, and M. Ying, Parallel quantum algorithm for Hamiltonian simulation, [arXiv:2105.11889](https://arxiv.org/abs/2105.11889).
- [31] I. Kerenidis, A method for loading classical data into quantum states for applications in machine learning and optimization, U.S. Patent Application 16/986,553.
- [32] S. Ramos-Calderer, A. Pérez-Salinas, D. García-Martín, C. Bravo-Prieto, J. Cortada, J. Planaguma, and J. I. Latorre,

- Quantum unary approach to option pricing, *Phys. Rev. A* **103**, 032414 (2021).
- [33] S. Johri, S. Debnath, A. Mocherla, A. Singk, A. Prakash, J. Kim, and I. Kerenidis, Nearest centroid classification on a trapped ion quantum computer, *npj Quantum Inf.* **7**, 122 (2021).
- [34] N. Mathur, J. Landman, Y. Li, M. Strahm, S. Kazdaghli, A. Prakash, and I. Kerenidis, Medical image classification via quantum neural networks, [arXiv:2109.01831](https://arxiv.org/abs/2109.01831).
- [35] L. K. Grover, Synthesis of Quantum Superpositions by Quantum Computation, *Phys. Rev. Lett.* **85**, 1334 (2000).
- [36] Y. R. Sanders, G. H. Low, A. Scherer, and D. W. Berry, Black-Box Quantum State Preparation without Arithmetic, *Phys. Rev. Lett.* **122**, 020502 (2019).
- [37] S. Wang, Z. Wang, G. Cui, S. Shi, R. Shang, L. Fan, W. Li, Z. Wei, and Y. Gu, Fast black-box quantum state preparation based on linear combination of unitaries, *Quantum Inf. Proc.* **20**, 270 (2021).
- [38] J. Bausch, Fast black-box quantum state preparation, [arXiv:2009.10709](https://arxiv.org/abs/2009.10709).
- [39] A. N. Soklakov and R. Schack, Efficient state preparation for a register of quantum bits, *Phys. Rev. A* **73**, 012307 (2006).
- [40] C. Zoufal, A. Lucchi, and S. Woerner, Quantum generative adversarial networks for learning and loading random distributions, *npj Quantum Inf.* **5**, 103 (2019).
- [41] C. Bravo-Prieto, D. García-Martín, and J. I. Latorre, Quantum singular value decomposer, *Phys. Rev. A* **101**, 062310 (2020).
- [42] P. Caraianni, The predictive power of singular value decomposition entropy for stock market dynamics, *Physica A* **393**, 571 (2014).
- [43] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, *Nature (London)* **549**, 242 (2017).
- [44] K. Nakaji and N. Yamamoto, Expressibility of the alternating layered ansatz for quantum computation, *Quantum* **5**, 434 (2021).
- [45] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, *Nat. Commun.* **12**, 1791 (2021).
- [46] N. Ahmed and K. R. Rao, Walsh-Hadamard transform, in *Orthogonal Transforms for Digital Signal Processing* (Springer, Berlin, Heidelberg, 1975), pp. 99–152.
- [47] R. Scheibler, S. Haghightshoar, and M. Vetterli, A fast Hadamard transform for signals with sublinear sparsity in the transform domain, *IEEE Trans. Inf. Theory* **61**, 2115 (2015).
- [48] B. K. Sriperebudur, A. Gretton, K. Fukumizu, G. Lanckriet, and B. Schölkopf, Injective Hilbert space embeddings of probability measures, in *Proceedings of the 21st Annual Conference on Learning Theory (COLT 2008)* (Omni Press, Madison, WI, 2008), pp. 111–122.
- [49] K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf, Kernel measures of conditional dependence, in *Advances in Neural Information Processing Systems 20* (Curran Associates, Red Hook, NY, 2007), pp. 489–496.
- [50] J.-G. Liu and L. Wang, Differentiable learning of quantum circuit Born machines, *Phys. Rev. A* **98**, 062324 (2018).
- [51] H. Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, *Ann. Math. Stat.* **23**, 493 (1952).
- [52] B. K. Sriperebudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. Lanckriet, On integral probability metrics, ϕ -divergences and binary classification, [arXiv:0901.2698](https://arxiv.org/abs/0901.2698).
- [53] G. Crooks, Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition, [arXiv:1905.13311](https://arxiv.org/abs/1905.13311).
- [54] H.-Y. Huang, R. Kueng, and J. Preskill, Predicting many properties of a quantum system from very few measurements, *Nat. Phys.* **16**, 1050 (2020).
- [55] H.-Y. Huang, R. Kueng, and J. Preskill, Efficient Estimation of Pauli Observables by Derandomization, *Phys. Rev. Lett.* **127**, 030503 (2021).
- [56] S. Hillmich, C. Hadfield, R. Raymond, A. Mezzacapo, and R. Wille, Decision diagrams for quantum measurements with shallow circuits, in *Proceedings of the 2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, Piscataway, NJ, 2021), pp. 24–34.
- [57] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, *Nat. Commun.* **9**, 4812 (2018).
- [58] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, An initialization strategy for addressing barren plateaus in parametrized quantum circuits, *Quantum* **3**, 214 (2019).
- [59] T. Volkoff and P. J. Coles, Large gradients via correlation in random parametrized quantum circuits, *Quantum Sci. Technol.* **6**, 025008 (2021).
- [60] D. Wierichs, C. Gogolin, and M. Kastoryano, Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer, *Phys. Rev. Research* **2**, 043246 (2020).
- [61] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, Variational quantum algorithms, *Nat. Rev. Phys.* **3**, 625 (2021).
- [62] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, An adaptive variational algorithm for exact molecular simulations on a quantum computer, *Nat. Commun.* **10**, 3007 (2019).
- [63] H. L. Tang, V. Shkolnikov, G. S. Barron, H. R. Grimsley, N. J. Mayhall, E. Barnes, and S. E. Economou, Qubit-ADAPT-VQE: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor, *PRX Quantum* **2**, 020310 (2021).
- [64] I. G. Ryabinkin, T.-C. Yen, S. N. Genin, and A. F. Izmaylov, Qubit coupled cluster method: A systematic approach to quantum chemistry on a quantum computer, *J. Chem. Theory Comput.* **14**, 6317 (2018).
- [65] N. V. Tkachenko, J. Sud, Y. Zhang, S. Tretiak, P. M. Anisimov, A. T. Arrasmith, P. J. Coles, L. Cincio, and P. A. Dub, Correlation-informed permutation of qubits for reducing ansatz depth in the variational quantum eigensolver, *PRX Quantum* **2**, 020337 (2021).
- [66] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, CutQC: Using small quantum computers for large quantum circuit evaluations, in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (ACM Press, New York, NY, 2021), pp. 473–486.
- [67] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, Simulating Large Quantum Circuits on a Small Quantum Computer, *Phys. Rev. Lett.* **125**, 150504 (2020).
- [68] E. Malvetti, R. Iten, and R. Colbeck, Quantum circuits for sparse isometries, *Quantum* **5**, 412 (2021).

- [69] V. Giovannetti, S. Lloyd, and L. Maccone, Quantum Random Access Memory, *Phys. Rev. Lett.* **100**, 160501 (2008).
- [70] D. K. Park, F. Petruccione, and J.-K. K. Rhee, Circuit-based quantum random access memory for classical data, *Sci. Rep.* **9**, 3949 (2019).
- [71] A. W. Harrow and J. C. Napp, Low-Depth Gradient Measurements Can Improve Convergence in Variational Hybrid Quantum-Classical Algorithms, *Phys. Rev. Lett.* **126**, 140502 (2021).
- [72] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Advances in Neural Information Processing Systems*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Curran Associates, Red Hook, NY, 2014), Vol. 27, pp. 2672–2680.
- [73] T. M. Cover, *Elements of Information Theory* (John Wiley & Sons, New York, 1999).
- [74] P. A. Samuelson, Proof that properly anticipated prices fluctuate randomly, in *The World Scientific Handbook of Futures Markets* (1965), pp. 25–38.
- [75] E. F. Fama, Efficient capital markets: A review of theory and empirical work, *J. Finance* **25**, 383 (1970).
- [76] L. Laloux, P. Cizeau, J.-P. Bouchaud, and M. Potters, Noise Dressing of Financial Correlation Matrices, *Phys. Rev. Lett.* **83**, 1467 (1999).
- [77] V. Plerou, P. Gopikrishnan, B. Rosenow, L. A. Nunes Amaral, and H. E. Stanley, Universal and Nonuniversal Properties of Cross Correlations in Financial Time Series, *Phys. Rev. Lett.* **83**, 1471 (1999).
- [78] V. Plerou, P. Gopikrishnan, B. Rosenow, L. A. Nunes Amaral, T. Guhr, and H. E. Stanley, Random matrix approach to cross correlations in financial data, *Phys. Rev. E* **65**, 066126 (2002).
- [79] A. Utsugi, K. Ino, and M. Oshikawa, Random matrix theory analysis of cross correlations in financial markets, *Phys. Rev. E* **70**, 026110 (2004).
- [80] J.-P. Onnela, A. Chakraborti, K. Kaski, and J. Kertesz, Dynamic asset trees and black Monday, *Physica A* **324**, 247 (2003).
- [81] W. A. Rizzo, The informational efficiency and the financial crashes, *Res. Int. Business Finance* **22**, 396 (2008).
- [82] D. Y. Kenett, Y. Shapira, A. Madi, S. Bransburg-Zabary, G. Gurgershoren, and E. Ben-Jacob, Index cohesive force analysis reveals that the us market became prone to systemic collapses since 2002, *PLoS ONE* **6**, e19378 (2011).
- [83] A. Nobi, S. Lee, D. H. Kim, and J. W. Lee, Correlation and network topologies in global and local stock indices, *Phys. Lett. A* **378**, 2482 (2014).
- [84] F. Ren and W.-X. Zhou, Dynamic evolution of cross-correlations in the Chinese stock market, *PLoS ONE* **9**, e97711 (2014).
- [85] L. Zhao, W. Li, and X. Cai, Structure and dynamics of stock market in times of crisis, *Phys. Lett. A* **380**, 654 (2016).
- [86] K. Yin, Z. Liu, and P. Liu, Trend analysis of global stock market linkage based on a dynamic conditional correlation network, *J. Business Econ. Manage.* **18**, 779 (2017).
- [87] A. Dutta, G. Aeppli, B. K. Chakrabarti, U. Divakaran, T. F. Rosenbaum, and D. Sen, *Quantum Phase Transitions in Transverse Field Spin Models: From Statistical Physics to Quantum Information* (Cambridge University Press, Cambridge, 2015).
- [88] H. Matsueda, Renormalization group and curved spacetime, [arXiv:1106.5624](https://arxiv.org/abs/1106.5624).
- [89] T. Li and X. Wu, Quantum query complexity of entropy estimation, *IEEE Trans. Inf. Theory* **65**, 2899 (2018).
- [90] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, Quantum amplitude amplification and estimation, *Contemp. Math.* **305**, 53 (2002).
- [91] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [92] M. S. Anis, Abby-Mitchell, H. Abraham, AduOffei, R. Agarwal, G. Agliardi, M. Aharoni, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, M. Amy, S. Anagolum *et al.*, Qiskit: An open-source framework for quantum computing (2019).
- [93] R. Gu, W. Xiong, and X. Li, Does the singular value decomposition entropy have predictive power for stock market? Evidence from the Shenzhen stock market, *Physica A* **439**, 103 (2015).
- [94] J. Civitarese, Volatility and correlation-based systemic risk measures in the US market, *Physica A* **459**, 55 (2016).
- [95] P. Caraianni, Modeling the comovement of entropy between financial markets, *Entropy* **20**, 417 (2018).
- [96] K. Manda *et al.*, Stock market volatility during the 2008 financial crisis, Leonard N. Stern School of Business, Glucksman Institute for Research in Securities Markets, List of Websites, Working Paper No. 16976 (2021).
- [97] S. Aaronson and P. Rall, Quantum approximate counting, simplified, in *Symposium on Simplicity in Algorithms* (SIAM, Philadelphia, 2020), pp. 24–32.
- [98] K. Nakaji, Faster amplitude estimation, *Quantum Inf. Computat.* **20**, 1109 (2020).