Efficient explicit circuit for quantum state preparation of piecewise continuous functions

Nikita Guseynov [*]
Global College, Shanghai Jiao Tong University, Shanghai 200240, China

Nana Liu [†]
Institute of Natural Sciences, School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai 200240, China;
Ministry of Education Key Laboratory in Scientific and Engineering Computing, Shanghai Jiao Tong University, Shanghai 200240, China;
Shanghai Artificial Intelligence Laboratory, Shanghai, China;
and Global College, Shanghai Jiao Tong University, Shanghai 200240, China

(Received 17 October 2025; accepted 9 December 2025; published 2 January 2026)

Efficiently uploading data into quantum states is essential for many quantum algorithms to achieve advantage across various applications. In this paper, we address this challenge by developing a method to upload a polynomial function $f(x)$ on the interval $x \in [-1, 1]$ into a pure quantum state consisting of qubits, where a discretized $f(x)$ is the amplitude of this state. The preparation cost has $\mathcal{O}(n \log n)$ scaling in the number of qubits n and linear scaling with the degree of the polynomial Q. This efficiency allows the preparation of states whose amplitudes correspond to high-degree polynomials (up to 10^4), enabling accurate approximation of functions that admit efficient polynomial series representations and whose amplitude profiles are not extremely localized. We provide a fully explicit circuit realization, based on four real polynomials that meet specific parity and boundedness conditions. We extend this construction to cover piecewise polynomial functions, a case not previously addressed explicitly in the literature, the algorithm scaling linearly with the number of piecewise parts. Our method achieves efficient quantum circuit implementation and we present detailed gate counting and resource analysis.

DOI: 10.1103/plc3-2jyx

I. INTRODUCTION

Quantum computing [1] holds significant promise for revolutionizing various fields such as material science [2], computer science [3–7], and finance [8,9] due to its potential to solve some problems more efficiently than canonical methods on classical computers. The development of efficient quantum algorithms is crucial to harnessing this power, enabling quantum computers to outperform their classical counterparts [10,11]. However, the effectiveness of these algorithms can be severely undermined by inefficient data loading processes [12].

In this paper, we present an explicit and resource-transparent method to upload a continuous or piecewise continuous function $f(x)$ defined on the interval $x \in [-1, 1]$ (where x is discretized along this interval with $\Delta x \sim 1/N$) into a digital qubit-based quantum state as the amplitudes of a normalized vector $|\psi\rangle^n$,

$$|\psi\rangle^n := \frac{1}{N_f} \sum_{x=0}^{2^n-1} f(x_i)|x\rangle^n, \tag{1}$$

with a size $N = 2^n$, where n is the number of qubits. The domain $[-1, 1]$ can always be generalized to any interval $[a, b]$ by a simple rescaling of x.

Quantum states of this kind play a key role in numerous quantum algorithms. They are used in discrete-time quantum walk approaches [13], and modeling electromagnetic scattering cross sections [14]. They also appear in quantum algorithms for solving linear and nonlinear partial differential equations [15,16] and in specifying priors for quantum phase estimation [17].

While the digital quantum circuit model offers a universal platform, the fundamental operations available on current quantum hardware are typically restricted to one- and two-qubit gates, such as CNOTS and single-qubit rotations, which play a role analogous to the not and or gates in classical computing [11]. A nontrivial challenge in this framework is to determine how, starting from the initial state $|0\rangle^n$, one can efficiently combine only these fundamental gates to prepare a state whose amplitudes encode a prescribed set of function values, as in Eq. (1). This "uploading" task is far from automatic: it requires decomposing the functional map into a concrete gate sequence, while optimizing for circuit depth and hardware constraints. The difficulty is particularly

[*]Contact author: guseynov.nm@gmail.com
[†]Contact author: nana.liu@quantumlah.org

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

TABLE I. Polynomial degree and gate counts for various functions [18,19]. The polynomial degree means that we retain only k terms from the expansion for the function; the gate counts directly follow from the series expansions from Table II and Theorem C 2. The main discretization parameter here is $n = \log_2 N$ which is the number of qubits. The error term ϵ represents the maximum absolute value of the residual beyond k terms in the series expansion across the given domain. The δ value for the reciprocal functions is defined by its domain. For ReLU and Leaky ReLU we suggest using the piecewise approach from Appendix D.

Name	Notation	Polynomial degree	Gate counts	Domain
Exponent	$e^{\alpha x}$	$\mathcal{O}\left(\frac{\lvert\alpha\rvert+\log(1/\epsilon)}{\log(1+\frac{\log(1/\epsilon)}{\lvert\alpha\rvert})}\right)$	$\mathcal{O}\left(n\log n\times\frac{\lvert\alpha\rvert+\log(1/\epsilon)}{\log(1+\frac{\log(1/\epsilon)}{\lvert\alpha\rvert})}\right)$	$x\in[-1,1];\alpha\in\mathbb{C}$
Cosine	$\cos(tx)$	$\mathcal{O}\left(t+\frac{\log(1/\epsilon)}{\log(e+\frac{\log(1/\epsilon)}{t})}\right)$	$\mathcal{O}\left[n\log n\left(t+\frac{\log(1/\epsilon)}{\log(e+\frac{\log(1/\epsilon)}{t})}\right)\right]$	$x\in[-1,1];t\in\mathbb{R}$
Sine	$\sin(tx)$	$\mathcal{O}\left(t+\frac{\log(1/\epsilon)}{\log(e+\frac{\log(1/\epsilon)}{t})}\right)$	$\mathcal{O}\left[n\log n\left(t+\frac{\log(1/\epsilon)}{\log(e+\frac{\log(1/\epsilon)}{t})}\right)\right]$	$x\in[-1,1];t\in\mathbb{R}$
Sigmoid	$\frac{1}{1+e^{-2x}}$	$\mathcal{O}(\log(1/\epsilon))$	$\mathcal{O}p[n\log n\times\log(1/\epsilon)]$	$x\in[-\pi/2,\pi/2]$
Gaussian	$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(x/\sigma)^2}$	$\mathcal{O}\left(\frac{\log(1/\epsilon)+\log(1/\sigma)}{\log[1+2\sigma\log(1/\epsilon)]}\right)$	$\mathcal{O}\left(n\log n\times\frac{\log(1/\epsilon)+\log(1/\sigma)}{\log[1+2\sigma\log(1/\epsilon)]}\right)$	$x\in[-1,1];\sigma>0$
Bessel	$J_n(\alpha x)$	$\mathcal{O}\left(\frac{\log(1/\epsilon)}{\log[1+\frac{2}{\lvert\alpha\rvert}\log(1/\epsilon)]}\right)$	$\mathcal{O}\left(n\log n\times\frac{\log(1/\epsilon)}{\log[1+\frac{2}{\lvert\alpha\rvert}\log(1/\epsilon)]}\right)$	$x\in[-1,1];\alpha\in\mathbb{R}$
Reciprocal	$\frac{1}{x}$	$\mathcal{O}(\delta\sqrt{\log^2(1/\epsilon)+\log(1/\epsilon)\log(\delta)})$	$\mathcal{O}(\delta n\log n\times\sqrt{\log^2(1/\epsilon)+\log(1/\epsilon)\log(\delta)})$	$x\in\left[-1,-\frac{1}{\delta}\right]\cup\left[\frac{1}{\delta},1\right]$
ReLU	ReLU(x)	$\mathcal{O}(1)$	$\mathcal{O}(n\log n)$	$x\in\mathbb{R}$
Leaky ReLU	Leaky ReLU(x)	$\mathcal{O}(1)$	$\mathcal{O}(n\log n)$	$x\in\mathbb{R}$

pronounced for large n, where the direct uploading procedure would require storing 2^n array of data, an exponential classical cost. The subsequent quantum brute-force routine [20,21] to upload that array has gate cost $\mathcal{O}(\sqrt{2^n})$ which is inefficient.

Since polynomials are efficiently computable classically, one might expect quantum arithmetic circuits to be the natural choice for state preparation. However, in practice, such circuits are significantly more expensive in terms of gate complexity and ancilla requirements. In addition, quantum arithmetic algorithms often require the implementation of functions such as the arccosine, which is technically challenging and resource intensive. To avoid this costly overhead, we present a method that directly constructs state preparation unitaries using the polynomial structure itself, without relying on arithmetic subroutines [22,23], offering practical circuit synthesis for encoding piecewise continuous functions into quantum states.

The resource cost of our method scales as $\mathcal{O}(Qn\log n)$, where n is the number of qubits and Q is the degree of the polynomial. This linear dependence on Q enables the efficient construction of high-degree polynomials, allowing us to upload polynomial approximations of functions with known series expansions (such as Taylor or Jacobi-Anger expansions [19]) in regimes where these series are efficient and the filling ratio $\mathcal{F}$ remains appreciable. Furthermore, we extend the construction to accommodate piecewise polynomial functions while preserving the same scaling behavior. For a function divided into G segments, the total resource scaling becomes $\mathcal{O}(Qn\log n+Gn+QG)$.

To achieve this, we approximate the target complex function as a superposition of four real polynomials with different parities. For each of those polynomial we build an independent state preparation unitary using the alternating phase transition sequence (APTS) [24], also known as qubitization [25]. This procedure applies the target polynomial function on the eigenvalues λ_i of a block-encoded simple matrix. In particular, we use the matrix with linear eigenvalues $\lambda_i\sim Ai+B$ as an input for the APTS. Finally, we use the four prepared block-encoded polynomials as an input for a linear combination of unitaries (LCU) [25] to embed those polynomials into the desired function.

In addition, we suggest a method to construct a piecewise polynomial function, defined as an independent polynomial of different degrees in each sector. To realize this in practice, we developed an efficient procedure that uses an ancilla qubit as an indicator for each segment $-1\leqslant a_1\leqslant b_1\leqslant 1$ of the domain $[-1,1]$. For a given computational basis state $|i\rangle^n$, the unitary gate sets this ancilla to $|1\rangle$ if and only if the state belongs to the corresponding segment $x_i\in[a_1,b_1]$, effectively marking the target region. We then apply the controlled version of the state preparation algorithm described above, ensuring that the appropriate polynomial is uploaded only for the predetermined segment. After completing the segment-specific operation, we uncompute the indicator qubit, returning it to $|0\rangle$ so it can be reused later. This method enables independent polynomials of arbitrary degree to be assigned to each segment and preserves the linear scaling in terms of the polynomial degree Q.

This document is organized as follows. Section II provides the theoretical foundations, including definitions and essential mathematical tools. In Sec. III, we present the main results of the paper. Section IV reviews alternative methods from the literature and compares them with our approach. Appendixes C and D detail our explicit algorithm for function uploading, outlining each step of the process.

II. PRELIMINARIES

Now we introduce some preliminary mathematical and quantum computing concepts that we will use, and the system of indexes we will employ. We start with defining the multi-control operator.

Definition 1 (Multi-control operator). Let U be an m-qubit quantum unitary and b a bit string of length n. We define C_U^b as an $n+m$-qubit quantum unitary that applies U to an m-qubit quantum register if and only the n-qubit quantum register is in state $|b\rangle$. In Appendix A we show how this gate can be implemented using the C_U^1 operator, $16n-16$ single-qubit

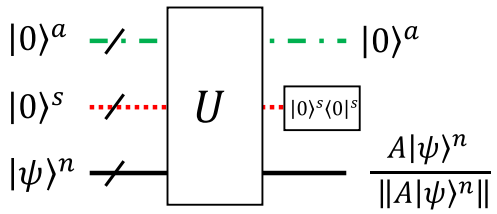


FIG. 1. The general scheme of block-encoding for matrix A. The last operation on the second register indicates measurement of zero, projecting the main n-qubit register onto the desired state. Wires for pure ancilla (Definition 2) are depicted as green dash-dotted lines; workspace ancilla (Definition 4) are shown as red dotted wires; and the system register is represented by a solid black wire.

operations, and $12n - 12$ CNOTS

$$C_U^b = |b\rangle^n\langle b|^n \otimes U + \sum_{\substack{i=0,\ldots,2^n-1 \\ i\neq b}} |i\rangle^n\langle i|^n \otimes I^{\otimes m}.$$

Definition 2 (Pure ancilla). Let U be an $n+m$-qubit unitary operator such that for any arbitrary $|\psi\rangle^n$,

$$U|0\rangle^m|\psi\rangle^n = |0\rangle^m|\phi\rangle^n, \tag{2}$$

where $|\phi\rangle^n$ is some quantum state. Then we say that the m-qubit quantum register is pure ancilla for operator U, if before and after the action it is in the state $|0\rangle^m$.

We note that from Eq. (2), there must exist an n-qubit unitary V such that $|\phi\rangle^n = V|\psi\rangle^n$, i.e., on the subspace where the ancilla is in the $|0\rangle^m$ state, U acts as $I^m \otimes V$.

Throughout this paper, in all figures for quantum schemes, we depict wires corresponding to such pure ancillas as green dash-dot lines.

Remark 1. A pure ancilla is simply a qubit that is required to be present in the quantum computer for intermediate computation but does not need to be measured at the end of the algorithm. Since it is returned to the $|0\rangle$ state after use, it can be safely reused in subsequent computations or circuits without resetting. In Fig. 3(c) we use $m-2$ auxiliary qubits for efficient CNOT construction. Exploitation of this qubits is divided into two epochs: (i) entangling stage that allows us to achieve some sophisticated quantum states and (ii) purification stage that sets ancillas back to the zero state.

Definition 3 [Block-encoding (modified Definition 43 from [24])]. Suppose that A is an n-qubit operator, $\alpha, \epsilon \in \mathbb{R}_+$; $a, s \in \mathbb{Z}_+$, and let $(a+s+n)$-qubit unitary U be so that for any arbitrary quantum state $|\psi\rangle^n$,

$$U|0\rangle^a|0\rangle^s|\psi\rangle^n = |0\rangle^a|\phi\rangle^{n+s}; \quad (\langle 0|^s \otimes I^{\otimes n})|\phi\rangle^{n+s} = \tilde{A}|\psi\rangle^n,$$

then we say that U is an (α, s, ϵ) block encoding of A, if

$$||A - \alpha\tilde{A}|| \leqslant \epsilon,$$

where the s-qubit register, referred to as the workspace ancilla (Definition 4), must be measured in the zero state to achieve the desired action on the system; if any other outcome is observed, the result is discarded.

We underline that the unitary U can exploit some auxiliary qubits setting them back to zero state. Later we call such qubits "pure ancillas" (Definition 2). The general scheme of block encoding is presented in Fig. 1.

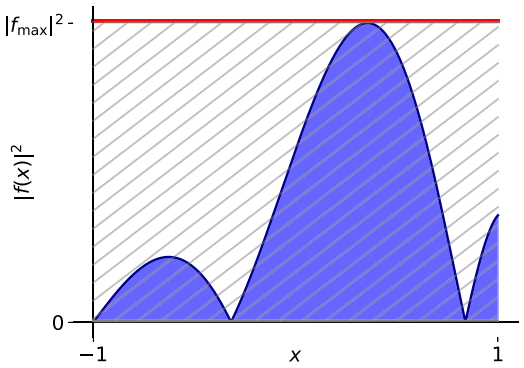


FIG. 2. The geometrical interpretation of the filling ratio. The filling ratio is the area under the function $|f(x)|^2$ (blue area) relative to its doubled absolute maximum value $2|f_{\max}|^2$ (area in the dashed rectangle).

Definition 4 (Workspace ancilla). Workspace ancilla are qubits initialized in $|0\rangle$ and added to the system register in block encoding. They enable embedding a nonunitary operator A as the top-left block of a unitary U:

$$U = \begin{pmatrix} A & * \\ * & * \end{pmatrix}.$$

After U is applied, workspace ancilla are not required to return to $|0\rangle$ and typically end in an unknown state (unlike the pure ancilla). The action of A is realized by projecting the workspace ancillas onto $|0\rangle^{\otimes a}$:

$$(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes |\psi\rangle) = A|\psi\rangle.$$

On quantum hardware, this corresponds to measuring all workspace ancillas and accepting only the all-zero result; otherwise, the system register state is discarded.

Throughout this paper, in all figures for quantum schemes, we depict wires corresponding to workspace ancillas as red dotted wires.

Definition 5 (Filling ratio [26]). Given the usual definition for L^p function norms of a Riemann integrable, bounded function f, $\|f\|_p = [\int_a^b |f(x)|^p dx]^{1/p}$, we define the *filling ratio* of f,

$$\mathcal{F} := \frac{\|f\|_2^2}{2\|f\|_{\max}^2}.$$

Note that for continuous functions over a closed interval $\|f\|_\infty =: \|f\|_{\max}$ represents the absolute largest value of the function. The filling ratio has a clear geometrical interpretation shown in Fig. 2 .

III. RESULTS

We begin with a continuous function $f(x)$ defined on the interval $[-1, 1]$; $f : \mathbb{R} \to \mathbb{C}$. We assume that it has a polynomial decomposition

$$f(x) = \sum_{i=0}^{Q} \sigma_i x^i; \quad \sigma_i \in \mathbb{C}. \tag{3}$$

Polynomial decomposition is a crucial tool in both theoretical and applied mathematics due to its flexibility and

TABLE II. Efficient series expansions and error terms [18,19]; as the error expression we use the absolute value of the number of terms retained $|\sum_{k+1}^{\infty}\ldots|$. $J_n(x)$: Bessel function of the first kind; $T_n(x)$: Chebyshev polynomials of the first kind; B_n: Bernoulli numbers; $\Gamma(x)$: Gamma function. We approximate each function using its series expansion up to k terms. For the reciprocal function, $K = \lceil \delta^2 \log(\delta\epsilon^{-1})\rceil$.

Name	Notation	Series expansion	Error expression	Domain
Exponent	$e^{\alpha x}$	$\sum_{s=0}^{k} \frac{(\alpha x)^s}{s!}$	$\epsilon \approx \frac{e^{\lvert\alpha\rvert+k+1}\lvert\alpha\rvert^{k+1}}{\sqrt{2\pi(k+1)}(k+1)^{k+1}}$	$x \in [-1,1]; \alpha \in \mathbb{C}$
Cosine	$\cos(tx)$	$J_0(t) + 2\sum_{s=1}^{k}(-1)^s J_{2s}(t) T_{2s}(x)$	$\epsilon \leqslant \frac{1.07}{\sqrt{2(k+1)}}\left(\frac{e\lvert t\rvert}{4(k+1)}\right)^{2(k+1)}$	$x \in [-1,1]; t \in \mathbb{R}$
Sine	$\sin(tx)$	$2\sum_{s=0}^{k}(-1)^s J_{2s+1}(t) T_{2s+1}(x)$	$\epsilon \leqslant \frac{1.07}{\sqrt{2(k+1)}}\left(\frac{e\lvert t\rvert}{4(k+1)}\right)^{2(k+1)}$	$x \in [-1,1]; t \in \mathbb{R}$
Gaussian	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$	$\frac{1}{\sigma\sqrt{2\pi}} \sum_{s=0}^{k} \frac{(-1)^s}{s!}\left(\frac{x^2}{2\sigma^2}\right)^s$	$\epsilon \approx \frac{1}{\sigma 2\pi\sqrt{k+1}}\left(\frac{e}{2(k+1)\sigma^2}\right)^{k+1}$	$x \in [-1,1]; \sigma > 0$
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	$\frac{1}{2} + \frac{1}{2}\sum_{s=1}^{k} \frac{2^{2s}(2^{2s}-1)B_{2s}}{(2s)!}\left(\frac{x}{2}\right)^{2s-1}$	$\epsilon \approx \left\lvert \frac{2^{2(k+1)}(2^{2(k+1)}-1)B_{2(k+1)}}{[2(k+1)]!}\right\rvert$	$x \in [-\pi/2, \pi/2]$
Reciprocal	$\frac{1}{x}$	$4\sum_{j=0}^{k}(-1)^j \sum_{s=j+1}^{K} \frac{\binom{2K}{K+s}}{2^{2K}} T_{2j+1}(x)$	$\epsilon \leqslant \exp\left(-\frac{-\log(\delta)+\sqrt{\log(\delta)^2+4(\frac{k}{\delta})^2}}{2}\right)$	$x \in [-1,-\frac{1}{\delta}] \cup [\frac{1}{\delta}, 1]$
Bessel	$J_n(\alpha x)$	$\sum_{s=0}^{k} \frac{(-1)^s}{s!\,\Gamma(s+n+1)}\left(\frac{\alpha x}{2}\right)^{2s+n}$	$\epsilon \approx \frac{(\frac{\lvert\alpha\rvert}{2})^{2(k+1)+n} e^{k+n+1}}{2\pi\,(k+1)^{k+\frac{3}{2}}(k+n+1)^{k+n+2}}$	$x \in [-1,1]; \alpha \in \mathbb{R}$
Hyp. Tan.	$\tanh(x)$	$\sum_{s=1}^{k} \frac{2^{2s}(2^{2s}-1)B_{2s}x^{2s-1}}{(2s)!}$	$\epsilon \approx \lvert\frac{2^{2(k+1)}(2^{2(k+1)}-1)B_{2(k+1)}x^{2(k+1)-1}}{[2(k+1)]!}\rvert$	$x \in [-\pi/2, \pi/2]$

effectiveness in approximating complex functions. As shown in Tables I and II, various functions can be effectively approximated using polynomial series expansions, each with specific error bounds and applicable domains.

Our main goal in this paper is to upload such a function into a digital qubit-based quantum computer as a normalized vector $|\psi\rangle^n \in \mathbb{C}^{2^n}$, defined as

$$|\psi\rangle^n = \frac{\sum_{k=0}^{2^n-1}\sum_{i=0}^{Q}\sigma_i x_k^i |k\rangle^n}{\sqrt{\sum_{\kappa=0}^{2^n-1}\left|\sum_{j=0}^{Q}\sigma_j x_\kappa^j\right|^2}}, \tag{4}$$

which is a discrete version of Eq. (3) with partitioning size $\Delta x = \frac{2}{2^n-1}$. Thus, given the discrete representation $x_k = -1 + k\Delta x$, we can rewrite Eq. (4) as

$$|\psi\rangle^n = \frac{\sum_{i=0}^{Q}\sum_{k=0}^{2^n-1}\sigma_i(a+k\Delta x)^i |k\rangle^n}{\sqrt{\sum_{\kappa=0}^{2^n-1}\left|\sum_{j=0}^{Q}\sigma_j(a+\kappa\Delta x)^j\right|^2}}. \tag{5}$$

Our next step is to show that any polynomial of the form in Eq. (3) can be written as a sum of four polynomials P_s with the following properties:

(1) $P_s(x)$ has parity (0 or 1 mod 2), which means that it has only odd or even degrees of x;

(2) $P_s(x)$ is a real polynomial, which means that all the coefficients are real;

(3) For all $x \in [-1,1]$: $|P_s(x)| < 1$.

Thus, we decompose Eq. (3) into four real polynomials

$$f(x) = \alpha_1 P_1(x) + \alpha_2 P_2(x) + i\alpha_3 P_3(x) + i\alpha_4 P_4(x); \quad \alpha_s \in \mathbb{R}, \tag{6}$$

where $P_1(x)$ and $P_3(x)$ have parity 1, and $P_2(x)$ and $P_4(x)$ have parity 0. The third condition can be met by tuning α_i. For such $f(x)$, we provide an explicit algorithm for preparing pure quantum states whose amplitudes are proportional to $f(x)$. We will also provide the exact gate count and show explicitly the required quantum circuits. The following theorems are the central theorems in this paper.

Theorem 1 (Linear combination of polynomials). Let $f(x)$ be a continuous function $f : \mathbb{R} \to \mathbb{C}$ with polynomial decomposition

$$f(x) = \sum_{i=0}^{Q}\sigma_i x^i = \alpha_1 P_1(x) + \alpha_2 P_2(x) + i\alpha_3 P_3(x) + i\alpha_4 P_4(x);$$
$$\times\ \alpha_s \in \mathbb{R}; \quad \sigma_i \in \mathbb{C},$$

where polynomials $P_s(x)$ satisfy the conditions 1, 2, 3. Then we can construct a unitary operation U_f that prepares a 2^n-discrete version of $f(x)$

$$|\psi\rangle^n = \frac{\sum_{i=0}^{Q}\sum_{k=0}^{2^n-1}\sigma_i x_k^i |k\rangle^n}{\sqrt{\sum_{\kappa=0}^{2^n-1}\left|\sum_{j=0}^{Q}\sigma_j x_\kappa^j\right|^2}},$$

with probability $P \sim \mathcal{F}$ (see Definition 5) and resources no greater than

(1) $\mathcal{O}(Qn\log_2 n)$ quantum gates (CNOTS and one-qubit rotations);

(2) $\lceil \log_2 n\rceil + 1$ pure ancillas.

Proof. The proof is provided in Appendix C. The central idea underlying the theorem is to apply the singular value transformation to a diagonal unitary operator with linear eigenvalues which represent coordinate values on the interval $[-1,1]$.

Remark 2. While our method does not rely directly on properties such as smoothness or Lipschitz continuity, these characteristics do influence the efficiency of the polynomial approximation. For example, smooth or Lipschitz-continuous functions can often be approximated to a given precision ϵ using lower-degree polynomials, which, in turn, reduces the gate complexity of our circuit. However, functions with discontinuities or sharp features typically require higher-degree polynomials or a piecewise representation.

Theorem 2 (Piece-wise polynomial function). Let $f(x)$ be a piecewise continuous function $f : \mathbb{R} \to \mathbb{C}$ with polynomial

decomposition

$$f(x) = \begin{cases} f_1(x) = \sum_{i=0}^{Q_1} \sigma_i^{(1)} x^i = \alpha_1^{(1)} P_1^{(1)}(x) + \alpha_2^{(1)} P_2^{(1)}(x) + i\alpha_3^{(1)} P_3^{(1)}(x) + i\alpha_4^{(1)} P_4^{(1)}(x), & \text{if } K_1 \geqslant x \geqslant a, \\ f_2(x) = \sum_{i=0}^{Q_2} \sigma_i^{(2)} x^i = \alpha_1^{(2)} P_1^{(2)}(x) + \alpha_2^{(2)} P_2^{(2)}(x) + i\alpha_3^{(2)} P_3^{(2)}(x) + i\alpha_4^{(2)} P_4^{(2)}(x), & \text{if } K_2 \geqslant x > K_1, \\ \vdots & \\ f_G(x) = \sum_{i=0}^{Q_G} \sigma_i^{(G)} x^i = \alpha_1^{(G)} P_1^{(G)}(x) + \alpha_2^{(G)} P_2^{(G)}(x) + i\alpha_3^{(G)} P_3^{(G)}(x) + i\alpha_4^{(G)} P_4^{(G)}(x), & \text{if } b \geqslant x > K_{G-1}, \end{cases}$$
$$\sigma_i^{(j)} \in \mathbb{C}; \quad \alpha_s^{(j)} \in \mathbb{R},$$

where G is the number of the pieces and Q_g is the polynomial degree in the piece g with $\max_g Q_g = Q_{\max}$. The polynomials $P_s^{(j)}(x)$ satisfy the conditions 1, 2, 3. Then we can construct a unitary operation U_f that prepares a 2^n-discrete version of $f(x)$ with probability $P \sim \mathcal{F}$ (see Definition 5) and resources no greater than

(1) $\mathcal{O}(Q_{\max} n \log_2 n + Gn + Q_{\max} G)$ CNOTS and one-qubit operations;

(2) $n + \lceil \log_2 G \rceil - 1$ pure ancillas.

Proof. The proof is provided in Appendix D. The main idea is to prepare an auxiliary register in the state $|1\rangle$ for the interval $K_i \geqslant x > K_{i-1}$, then apply the controlled version of the quantum circuit from Theorem C 2.

A subtle limitation of our approach arises when the filling ratio $\mathcal{F}$ (Definition 5) becomes exponentially small. For typical continuous or piecewise-continuous functions this does not occur, and the resource scaling in Theorems 1 and 2 remains valid. However, for highly localized functions whose weight concentrates on only a few grid points, one may have $\mathcal{F} = \mathcal{O}(2^{-n})$, in which case the postselection inherent in the block encoding must be repeated $1/\mathcal{F} = \mathcal{O}(2^n)$ times undermining the algorithm efficiency. These cases lie outside the main regime of interest for this work, but we list two strategies how the filling ratio can be boosted by simple preprocessing of the input amplitudes, exploiting the fact that our framework yields with unitary which acts on the state $|+\rangle^{\otimes n}$. We outline the representative strategies as follows:

(1) Sectoring. When $f(x)$ contains a narrow peak near s, one may replace the uniform input state $|+\rangle^{\otimes n}$ with a two-level, piecewise-constant amplitude profile $|g\rangle = \sum_k g(x_k)|k\rangle$, where $g(x)$ takes a small value C_1 outside $[s - \varepsilon, s + \varepsilon]$ and a larger value C_2 inside it. Such states correspond to a single rectangular "feature" and can be prepared efficiently using the feature-based state-preparation method [27]. Thus, the framework considered in our paper implements the effective function $f'(x) = f(x)/g(x)$, which has a larger filling ratio, though it requires a separate polynomial decomposition of f', while postselection still recovers the normalized state corresponding to the original function $f(x)$.

(2) Approximate-profile compensation. Suppose one can efficiently prepare a simple function $g(x)$ approximating $f(x)$, and has a polynomial decomposition of the ratio $f(x)/g(x)$. We first upload g as the input state and then apply our function-uploading procedure to f/g, thereby transforming $|g\rangle$ into the desired state for f. Since g captures the dominant shape, the effective filling ratio $\mathcal{F}$ of f/g is significantly increased, including the case of exponentially decaying functions. Sectoring corresponds to the special case where g is a two-level profile.

These techniques mitigate the exponentially small-$\mathcal{F}$ regime for certain cases, although a full treatment is beyond the scope of this paper.

IV. RELATED WORK

There are numerous methods addressing the problem at hand. The authors of Ref. [20] proposed an algorithm for uploading any vector (without an analytical expression) with circuit depth and width of $\mathcal{O}(\sqrt{2^n})$, which is crucial for fields like quantum machine learning [21], where a quantum computer operates on arbitrary states. Additionally, various optimization approaches exist; one promising method [28] used a genetic algorithm to iteratively build the desired quantum state. The authors conjectured this method may scale as $\mathcal{O}[\text{poly}(n)]$.

Among other approaches, the authors of Ref. [29] leveraged Schmidt decomposition. The complexity of this algorithm depended on the Schmidt rank and can significantly reduce the number of CNOT gates. The complexity is $\mathcal{O}(2^m)$, where m is the Schmidt measure.

Our approach not only prepares continuous functions that admit efficient polynomial approximations and reasonably large filling ratio $\mathcal{F}$, but also provides an explicit method for constructing piecewise polynomial functions. This is achieved with no change to the asymptotic scaling, offering a scalable solution for approximating nonsmooth or segmented data. To the best of our knowledge, none of the below-mentioned methods address this important instance explicitly, making our method a uniquely flexible and practical alternative.

A particularly promising method is proposed in Ref. [30], where the authors constructed functions using a linear combination of block encodings for Fourier and Chebyshev series. Their gate complexity is $\mathcal{O}(Qn \log n)$ for a Chebyshev series of degree Q, and $\mathcal{O}(dn \log d)$ for a Fourier series with harmonic d. The key difference with our method lies in how polynomial superpositions are formed. They employ LCU, while we rely on quantum singular value transformation (QSVT). Our method also requires fewer workspace ancillas: $\mathcal{O}(\log n)$ versus $\mathcal{O}(\log n + \log Q)$ or $\mathcal{O}(2 \log d + 1)$, simplifying implementation.

The findings in Ref. [31] are based on diagonal block encoding of amplitudes [32,33] applied to a linearly dependent circuit, and then modified using singular value transformation. While this algorithm is conceptually similar to ours, it has worse complexity: $\mathcal{O}(Qn^2)$ versus our $\mathcal{O}(Qn \log n)$. It also requires $\mathcal{O}(n)$ workspace ancillas.

Another important early method is the Grover-Rudolph algorithm [34], which prepares quantum states correspond-

ing to integrals of a given probability density function. This method scales as $\mathcal{O}[\text{poly}(n)]$ for certain smooth distributions and provides a strong foundation for amplitude encoding. Additionally, it does not depend on the filling ratio $\mathcal{F}$; its probability of success is 1.

The main disadvantage of the Grover-Rudolph approach is its reliance on quantum amplitude oracles that prepare conditional probabilities of the form

$$f(i) = \frac{\int_{\frac{x_R^i + x_L^i}{2}}^{x_R^i} p(x)\,dx}{\int_{x_L^i}^{x_R^i} p(x)\,dx}, \tag{7}$$

where x_L^i, x_R^i are subinterval endpoints and $p(x)$ is a target probability density. This formulation assumes access to efficient integration or a known cumulative distribution function (CDF), which limits the method's applicability. In each recursive step, the method requires the transformation

$$\sqrt{p_i^{(m)}}\,|i\rangle\,|0\cdots 0\rangle \longrightarrow \sqrt{p_i^{(m)}}\,|i\rangle\,|\theta_i\rangle, \quad \theta_i = \arccos(\sqrt{f(i)}). \tag{8}$$

Now assume the target amplitude function is a polynomial

$$\sqrt{p(x)} = \sum_{k=0}^{Q} a_k\,x^k; \quad p(x) = \left(\sum_{k=0}^{Q} a_k\,x^k\right)^2 = \sum_{m=0}^{2Q} b_m\,x^m. \tag{9}$$

Then the conditional probability becomes

$$f(i) = \frac{\text{poly}_{2Q+1}}{\text{poly}_{2Q+1}}; \quad \theta_i = \arccos\left(\sqrt{\frac{\text{poly}_{2Q+1}}{\text{poly}_{2Q+1}}}\right). \tag{10}$$

Thus, although we started with a seemingly simpler goal (quantum state preparation for a degree Q polynomial) the method ends up requiring the construction of oracles for much more complicated functions of degree $2Q+1$ and transcendental transformations. This significantly increases both classical and quantum implementation complexity. The authors do not specify how to construct such oracles, which is nontrivial in practice.

Earlier QSVT-based approach [35] requires approximating the composed function $f[\arcsin(\cdot)]$. This composition tends to make polynomial approximation harder: ensuring boundedness and parity on $[-1, 1]$ is more delicate, coefficients can be less well conditioned, and achieving the same approximation error may require a higher polynomial degree Q.

Other lines of work emphasize regimes with small heralded success probabilities [36] or concentrate on low-degree polynomial cases.

ACKNOWLEDGMENTS

N.L. acknowledges funding from the Science and Technology Program of Shanghai, China (No. 21JC1402900), NSFC Grants No. 12471411 and No. 12341104, the Shanghai Jiao Tong University 2030 Initiative, and the Fundamental Research Funds for the Central Universities. N. Guseynov acknowledges funding from NSFC Grant No. W2442002.

DATA AVAILABILITY

No data were created or analyzed in this study.

APPENDIX A: EXPLICIT QUANTUM CIRCUITS FOR THE MULTICONTROL UNITARIES

In this appendix, we explicitly demonstrate how we address multicontrol operations and gate counting. The general structure of a multicontrol operator is illustrated in Fig. 3. It is known [11] that the Toffoli gate can be realized using six CNOTS and eight single-qubit operations. Consequently, the resource requirement for a single multicontrol operator includes no more than one C_U^1 operator, $16n - 16$ single-qubit operations, and $12n - 12$ CNOTS, simplifying the complexity of C_X^b to $C_X^{11\cdots 1}$ for simplicity. Additionally, we utilize $n-1$ pure ancillas. In this paper, we compute the complexity of the controlled unitary C_U^1 based on the gate complexity of U by substituting all CNOT gates with Toffoli gates (each implemented using six CNOTS and eight single-qubit operations), and by replacing single-qubit gates with their controlled versions (requiring two CNOTS and two single-qubit rotations), following the standard construction in Ref. [11].

APPENDIX B: EFFICIENT SERIES EXPANSIONS AND ERROR TERMS

In this appendix, we provide an overview of efficient series expansions for common functions and their corresponding error terms. Table II presents the notation, series expansions, error expressions, and domains for a variety of functions such as exponent, trigonometric functions, and others. Below, we define the terms and explain how the error is computed.

(i) Series Expansion: Each function is approximated by truncating its series expansion up to k terms. For example, the expansion for the exponential function e^x is $\sum_{s=0}^{k} \frac{x^s}{s!}$. These expansions provide approximate values for the functions within a specific domain.

(ii) Error Expression: The error term ϵ represents the residual sum of the terms beyond k in the series expansion. For each function, the error provides a measure of the approximation accuracy.

(iii) Norm of the Error: The error terms are presented in the infinity norm, denoted as $\|\cdot\|_\infty$, which corresponds to the maximum absolute value of the error across the given domain.

APPENDIX C: COORDINATE-POLYNOMIAL-ORACLE

The primary objective of this paper is to construct a function defined on the interval $[-1, 1]$. We begin by constructing an amplitude oracle for the coordinate operator $\hat{x}$ on the interval $[-1, 1]$. The operator $\hat{O}_x$ acts as follows:

$$\hat{O}_x|0\rangle^\lambda|k\rangle^n = \frac{1}{\mathcal{N}_x}x_k|0\rangle^\lambda|k\rangle^n + J_x|\perp_0\rangle^{n+\lambda};$$
$$x_k = -1 + \frac{2k}{2^n - 1}, \tag{C1}$$

where $|\perp_0\rangle^{n+\lambda}$ denotes a state orthogonal to $|0\rangle^\lambda \otimes I^{\otimes n}$ meaning $\forall|\psi\rangle^n : \langle\perp_0|^{n+\lambda}(|0\rangle^\lambda \otimes |\psi\rangle^n) = 0$. The second term

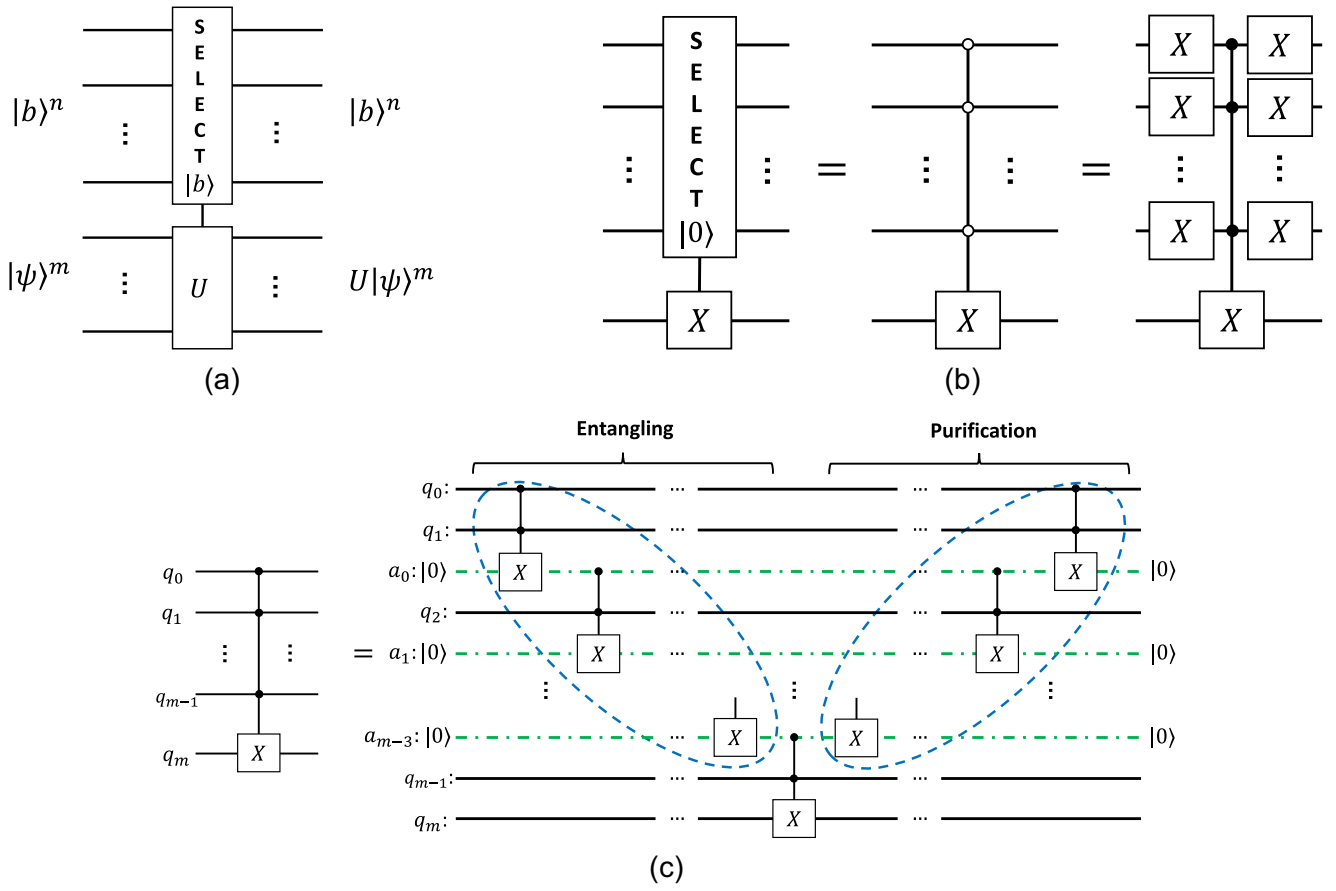


FIG. 3. (a) The general view of multicontrol operator C_U^b from Definition 1. Wires for pure ancilla are depicted as green dash-dotted lines; and the system register is represented by a solid black wire. Select operation means that the operator U is applied only if the state on the upper register is $|b\rangle^n$ (b) Explicit quantum circuit for implementation $C_X^{00\ldots0}$. (c) Explicit quantum circuit for implementation $C_X^{11\ldots1}$, which we use everywhere in this paper. We use $2m-3$ Toffoli application and $m-2$ pure ancillas.

involving J_x corresponds to the orthogonal component introduced by the block-encoding structure and ensures unitarity of the oracle $\hat{O}_x$. While this term can, in principle, be explicitly reconstructed from the following quantum circuits, we omit its full expression here as it is not important for the understanding of the algorithm and would be unnecessarily cumbersome.

Thus, $\hat{O}_x$ provides a $(\mathcal{N}_x, \lambda, 0)$ block encoding of the coordinate operator $\hat{x}$. In this paper we construct this operator using the LCU technique

$$\hat{x} = \begin{pmatrix} -1 & & & \\ & -1+\frac{2}{2^n-1} & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} = -\sum_{j=0}^{n-1} \frac{2^j}{2^n-1} Z_j; \quad Z_j = I \otimes \cdots \otimes I \otimes Z \otimes \underbrace{I \otimes \cdots \otimes I}_{j \text{ times}}, \tag{C2}$$

where $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ is the Pauli operator.

Lemma C 1 [Amplitude-oracle for coordinate operator (Appendix A from Ref. [37])]. Let $2^n \times 2^n$ matrix $\hat{x} = \mathrm{diag}(-1, -1+\frac{2}{2^n-1}, \ldots, 1)$ be a n-qubit finite-difference representation of coordinates as in the Eq. (C2), then we can construct a $(1, \lceil \log_2 n \rceil, 0)$ block encoding of the matrix $\hat{x}$:

$$\hat{O}_x |0\rangle^\lambda |k\rangle^n = x_k |0\rangle^\lambda |k\rangle^n + J_x |\perp_0\rangle^{n+\lambda};$$
$$x_k = -1 + \frac{2k}{2^n-1}; \quad \lambda = \lceil \log_2 n \rceil,$$

with resources no greater than

(1) $16n\lceil \log_2 n \rceil - 16n$ one-qubit operations;
(2) $12n\lceil \log_2 n \rceil - 16n$ CNOTS;
(3) $\lceil \log_2 n \rceil - 2$ pure ancillas.

Proof. We prove this lemma by demonstrating the circuit in Fig. 4, which implements the LCU technique to build the desired operator.

The implementation utilizes the auxiliary unitary

$$U_x = \frac{i}{\sqrt{2^n-1}} \sum_{j=0}^{n-1} 2^{j/2} |j\rangle^\lambda. \tag{C3}$$

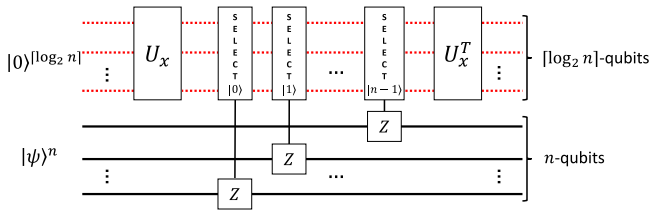


FIG. 4. A construction of the amplitude-oracle for $2^n \times 2^n$ matrix $\hat{x}$ which is $(1, \lceil \log_2 n \rceil, 0)$-block-encoding. The construction using LCU technique as in Eq. (C2). The unitary U_x is defined in the Eq. (C3) with its transposed version U_x^T. Workspace ancilla are shown as red dotted wires; and the system register is represented by a solid black wire.

The construction of U_x and U_x^T addresses the general problem of construction an arbitrary quantum state. The authors of [38,39] suggest an explicit implementation of U_x which consist of less than $\frac{23}{24}n$ CNOTS for an even number of qubits and n CNOTS for the odd number. Let's consider the number of one-qubit operations here. One can conclude that each CNOT can be accompanied by six Euler rotations; however, a rotation around the z axis commutes with control operation, and a rotation around x one commutes with Pauli X operation. Thus, the number of one-qubit Euler rotations cannot exceed the number of CNOTS by more than four times [40]. For simplicity in this paper we address a general one-qubit operation, so we merge two Euler rotations acting on each two qubits where the CNOT is applied. Thus, the complexity of the U_α^H implementation is less than: (i) $2n$ one-qubit rotations and (ii) n CNOTS.

From x to a polynomial

In this subsection we use the trick called qubitization [24] to transform the Amplitude-oracle for the $2^n \times 2^n$ matrix $\hat{x}$ denoted as $\hat{O}_x$ to the Amplitude-oracle for a polynomial of $\hat{x}$ to $\hat{O}_{P_s}$ (Coordinate-polynomial-oracle). The main building block in this trick is so called Alternating phase modulation sequence. This unitary operation is a key instrument for building $\hat{O}_{P_s}$.

Definition C 1 (Alternating phase modulation sequence). Let U be a $(1, m, 0)$ block encoding of the hermitian matrix A such that

$$A = (|0\rangle^m\langle 0|^m \otimes I^{\otimes n})U(I^{\otimes n} \otimes |0\rangle^m\langle 0|^m);$$

let $\Phi \in \mathbb{R}^q$, then we define the q-phased alternating sequence U_Φ as follows:

$$U_\Phi := \begin{cases} e^{i\phi_1(2\Pi - I)}U\prod_{j=1}^{(q-1)/2}(e^{i\phi_{2j}(2\Pi-I)}U^\dagger e^{i\phi_{2j+1}(2\Pi-I)}U) & \text{if } q \text{ is odd, and} \\ \prod_{j=1}^{q/2}(e^{i\phi_{2j-1}(2\Pi-I)}U^\dagger e^{i\phi_{2j}(2\Pi-I)}U) & \text{if } q \text{ is even,} \end{cases}$$

where $2\Pi - I = (2|0\rangle^m\langle 0|^m - I^{\otimes m}) \otimes I^{\otimes n}$.

Now we apply the Alternating phase modulation sequence on $\hat{O}_x$. In this case $m = 1$, $e^{i\phi(2\Pi - I)}$ turns to implementing the one-qubit rotation $C^{0\ldots0}_{\Phi(-2\phi)}$

$$\Phi(\theta) = \begin{pmatrix} e^{i\theta} & 0 \\ 0 & 1 \end{pmatrix}. \tag{C4}$$

The quantum circuit for U_Φ for this case is depicted in Fig. 5.

Now we have all the ingredients to build the Coordinate-polynomial-oracle $\hat{O}_{P_s}$ using singular value transformation.

Theorem C 1 [Singular value transformation by real polynomials (Corollary 18 from [24])]. Let $P_q(x)$ be a degree-Q polynomial as in Eq. (6) and U_Φ is the alternating phase modulation sequence as in Definition C 1 based on Amplitude-oracle for the n-qubit matrix $\hat{x} = \mathrm{diag}(-1, -1 + \frac{2}{2^n-1}, \ldots, 1)$ as in Lemma C 1. Then there exist $\Phi \in \mathbb{R}^n$,

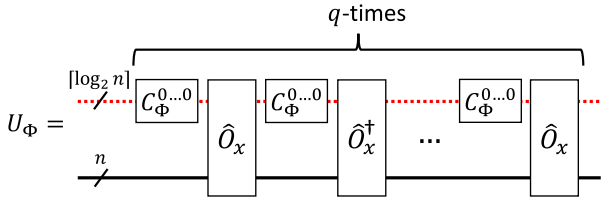


FIG. 5. Alternating phase modulation sequence U_Φ for $\hat{O}_x$ with odd q. Workspace ancilla are shown as red dotted wires; and the system register is represented by a solid black wire.

such that

$$\hat{O}_{P_s} = (H_W \otimes I^{\otimes n+\lceil \log_2 n \rceil + 1})(|0\rangle\langle 0| \otimes U_\Phi + |1\rangle\langle 1| \otimes U_{-\Phi}) \times (H_W \otimes I^{\otimes n+\lceil \log_2 n \rceil + 1}),$$

where $H_W = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. $\hat{O}_{P_s}$ is Coordinate-polynomial-oracle which is a $(1, \lceil \log_2 n \rceil + 1, 0)$ block encoding of $P_s(\hat{x})$:

$$\hat{O}_{P_s}|0\rangle^{\lceil \log_2 n \rceil + 1}|i\rangle^n = (P_s(x))_i|0\rangle^{\lceil \log_2 n \rceil + 1}|i\rangle^n + J^{(i)}_{P_s}|\perp_0\rangle^{n+\lceil \log_2 n \rceil + 1}. \tag{C5}$$

The total complexity of construction $\hat{O}_{P_s}$ does not exceed the following:

(1) $16Qn\lceil \log_2 n \rceil + 8Qn + 32Q\lceil \log_2 n \rceil - 48q + 2$ one-qubit operations;

(2) $12Qn\lceil \log_2 n \rceil + 4Qn + 24Q\lceil \log_2 n \rceil - 36q$ CNOTS;

(3) $\lceil \log_2 n \rceil - 1$ pure ancillas.

Moreover, given $P_s(x)$ and $\delta \geqslant 0$ we can find a $P'_s(x)$ and a corresponding Φ, such that $|P'_s(x) - P_s(x)| \leqslant \delta$ for all $x \in [-1, 1]$, using a classical computer in time $\mathcal{O}\{\mathrm{poly}[Q, \log(1/\delta)]\}$.

Proof. The rule of choosing Φ and the general proof is given in Ref. [24]. We compute complexities using Amplitude-oracle for x (Lemma C 1) as an input for U_Φ depicted in Fig. 5.

Remark C 1. To estimate the gate complexity of the Coordinate-polynomial-oracle, we note that the LCU implementing the controlled unitaries U_Φ and $U_{-\Phi}$ admits a useful simplification. As illustrated in Fig. 5, the only distinction between $|0\rangle\langle 0| \otimes U_\Phi$ and $|0\rangle\langle 0| \otimes U_{-\Phi}$ lies in the phase

operations $C_\Phi^{00\cdots0}$; the Amplitude-oracle $\hat{O}_x$ is identical in both branches. Hence the calls to $\hat{O}_x$ can be shared and need not be controlled by the selector qubit (whether it is in $|0\rangle$ or $|1\rangle$). Dropping these redundant controls on $\hat{O}_x$ reduces the overall LCU overhead and yields the gate counts stated above.

Our construction relies on precomputed QSP phase angles for the polynomials in Theorem C 1. We do not design new algorithms for this step, but instead assume access to existing classical solvers for quantum signal processing. Optimization-based methods such as Ref. [41] and stable factorization schemes such as Ying [42] provide numerically robust phase synthesis in double precision and have been demonstrated for polynomial degrees up to 10^4. Open-source implementations, for example, the python library `pyqsp` [43], can be used as the classical preprocessing backend for our scheme. A detailed analysis of the classical cost of phase computation is beyond the scope of this work, and we treat it as an assumed primitive.

Now we use LCU to combine four coordinate-polynomial-oracles to build the discrete version of $f(x)$ from Eq. (5).

Theorem C 2 (Linear combination of polynomials {Same as Theorem 1}). Let $f(x)$ be a continuous function $f:\mathbb{R}\to\mathbb{C}$ with polynomial decomposition

$$\begin{aligned} f(x) &= \sum_{i=0}^{Q} \sigma_i x^i \\ &= \alpha_1 P_1(x) + \alpha_2 P_2(x) + i\alpha_3 P_3(x) + i\alpha_4 P_4(x); \\ &\quad \times \sigma_i \in \mathbb{C}; \quad \alpha_s \in \mathbb{R}, \end{aligned}$$

where polynomials $P_s(x)$ satisfy

(1) $P_s(x)$ has parity-(s mod 2);
(2) $P_s(x) \in \mathbb{R}[x]$;
(3) for all $x \in [-1,1]$: $|P_s(x)| < 1$.

Then we can construct a unitary operation U_f which is $(\sum_{\kappa=0}^{4}|\alpha_s|, \lceil\log_2 n\rceil+3, 0)$ block encoding of $f(\hat{x})$. U_f prepares a 2^n-discrete version of $f(x)$:

$$|\psi\rangle^n = \frac{\sum_{i=0}^{Q}\sum_{k=0}^{2^n-1}\sigma_i x_k^i |k\rangle^n}{\sqrt{\sum_{\kappa=0}^{2^n-1}\left|\sum_{j=0}^{Q}\sigma_j x_\kappa^j\right|^2}},$$

with probability $P\sim\mathcal{F}$ (see Definition 5) and resources no greater than

(1) $16Qn\lceil\log_2 n\rceil - 16Qn + 32Q\lceil\log_2 n\rceil + 16n\lceil\log_2 n\rceil + 33n + 120Q + 2$ one-qubit operations;
(2) $12Qn\lceil\log_2 n\rceil - 16Qn + 24Q\lceil\log_2 n\rceil + 12n\lceil\log_2 n\rceil + 26n + 92Q$ CNOTS;
(3) $\lceil\log_2 n\rceil + 1$ pure ancillas.

Proof. We demonstrate the circuit design in Fig. 6, where

$$\begin{aligned} U_\alpha|0\rangle^2 = \frac{1}{\sqrt{\sum_{s=1}^{4}|\alpha_s|}}&[\sqrt{\alpha_1}|00\rangle^2 + \sqrt{\alpha_2}|01\rangle^2 \\ &+ e^{i\pi/4}\sqrt{\alpha_3}|10\rangle^2 + e^{i\pi/4}\sqrt{\alpha_4}|11\rangle^2]. \end{aligned} \tag{C6}$$

The operation U_α is an arbitrary two-qubit operation (three CNOTS, six one-qubit operations); the complexity of $\hat{O}_x$ is derived in Lemma 1. We highlight that during the creation of Coordinate-polynomial-oracles (see Theorem C 1) for all four polynomials it is necessary to apply $\hat{O}_x$ $Q-1$ times, so there is no need with their control versions; we apply the controlled version of $\hat{O}_x$ only once due to different parities, which is similar to Remark C 1.

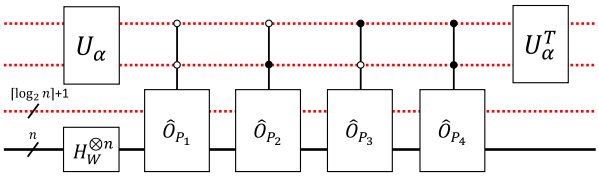


FIG. 6. Quantum circuit for the 2^n-discrete version of $f(x)$. The desired function is prepared if the first three registers from the top are measured in the zero state $|0\rangle$. The gate U_α is clarified in Eq. (C6) with a transposed version U_α^T; the H_W is Hadamard gate $H_W = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$. Workspace ancilla are shown as red dotted wires; and the system register is represented by a solid black wire.

The filling ratio $\mathcal{F}$ is an important quantity that measures how the total probability amplitude is distributed after applying the block encodings $\hat{O}_{P_s}$ to the constant superposition $H_W^{\otimes n}|0\rangle^n = |+\rangle^n$, as shown in Fig. 6, where $H_W = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$. The block encodings $\hat{O}_{P_s}$ have matrix elements bounded by 1, meaning that after their action, the total probability tends to decrease; the Fig. 2 demonstrated how the applying of the $\hat{O}_{P_s}$ cut the rectangle leaving the blue area under the plot. The total probability $P\sim\mathcal{F}$ is derived from the first term in the Eq. (C5).

APPENDIX D: PIECEWISE POLYNOMIAL FUNCTION

Now we generalize the pipeline described in Appendix C for the more general case of noncontinuous functions. We start with the simplest case of two pieces

$$f(x) = \begin{cases} \sum_{i=0}^{Q_1}\sigma_i^{(1)}x^i, & \text{if } a \leqslant x \leqslant K, \\ \sum_{i=0}^{Q_2}\sigma_i^{(2)}x^i, & \text{if } b \geqslant x > K, \end{cases} \quad \sigma_i^{(j)} \in \mathbb{C}. \tag{D1}$$

As in the previous section, our aim is to build a 2^n-discrete version of $f(x)$ as a quantum state on a digital- or qubit-based quantum computer [1].

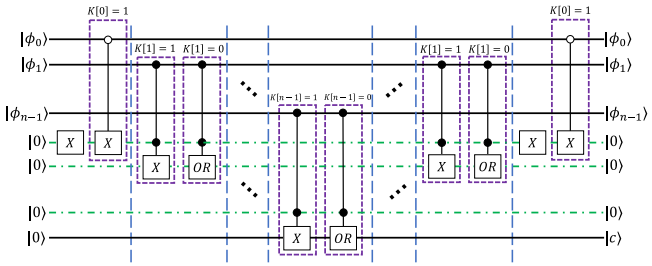


FIG. 7. Circuit design for initializing the indicator qubit. The whole circuit can be understood as a classical comparator as it compares the state (computational basis) in the upper register with K, setting the last qubit in the state $|c\rangle = |1\rangle$ if $\phi \geqslant K_n$. The quantum circuit uses $n-1$ pure ancillas. A classical array $K[i]$ holds the binary representation of K_n; the value of $K[i]$ determines which gate is applied. Wires for pure ancilla are depicted as green dash-dotted lines; and the system register is represented by a solid black wire.

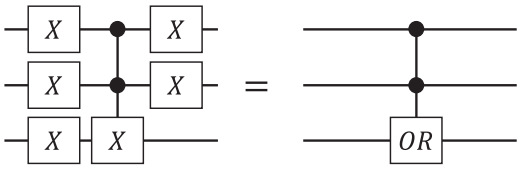


FIG. 8. The representation of the *OR* gate used in Fig. 7.

We aim to reduce the problem to the previous case; for this purpose, we introduce one ancillary qubit which keeps the information whether $k \geqslant K_n$ or not, where k is the number of a state in the computational basis and $K_n = \lceil (K-a)/\Delta x \rceil$ is a discrete coordinate of K. The initialization of this qubit (indicator qubit) is depicted in Fig. 7 and clarified in Fig. 8. The complexity of this procedure is no greater than

(1) $16n + 34$ one-qubit operations,
(2) $12n - 4$ CNOTS,
(3) $n - 1$ pure ancillas.

Using this auxiliary feature, we can prepare the 2^n-discrete version of $f(x)$ from Eq. (D1). The corresponding circuit is depicted in Fig. 9. By induction we prove Theorem D 1.

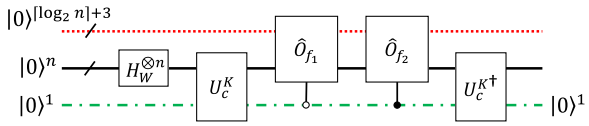


FIG. 9. Circuit design for preparing a piecewise function (two pieces) from Eq. (D1). Here the unitary U_c^K sets the last qubit in state $|1\rangle$ if the state in the middle n-qubit register is greater than K_n; the circuit design is depicted in Fig. 7. The oracles $\hat{O}_{f_j}$ are a controlled version of quantum circuits from Theorem C 2 but without Hadamard gates $H_W^{\otimes n}$ (see Fig. 6). The unitary $U_c^{K\dagger}$ returns the ancilla back to the zero state, making it a pure ancilla (see Definition 2). Wires for pure ancilla are depicted as green dash-dotted lines; workspace ancilla are shown as red dotted wires; and the system register is represented by a solid black wire.

Theorem D 1 (Piecewise polynomial function {Same as Theorem 2}). Let $f(x)$ be a piecewise continuous function $f : \mathbb{R} \to \mathbb{C}$ with polynomial decomposition

$$f(x) = \begin{cases} f_1(x) = \sum_{i=0}^{Q_1} \sigma_i^{(1)} x^i = \alpha_1^{(1)} P_1^{(1)}(x) + \alpha_2^{(1)} P_2^{(1)}(x) + i\alpha_3^{(1)} P_3^{(1)}(x) + i\alpha_4^{(1)} P_4^{(1)}(x), & \text{if } a \leqslant x \leqslant K_1, \\ f_2(x) = \sum_{i=0}^{Q_2} \sigma_i^{(2)} x^i = \alpha_1^{(2)} P_1^{(2)}(x) + \alpha_2^{(2)} P_2^{(2)}(x) + i\alpha_3^{(2)} P_3^{(2)}(x) + i\alpha_4^{(2)} P_4^{(2)}(x), & \text{if } K_2 \geqslant x > K_1, \\ \vdots & \\ f_G(x) = \sum_{i=0}^{Q_G} \sigma_i^{(G)} x^i = \alpha_1^{(G)} P_1^{(G)}(x) + \alpha_2^{(G)} P_2^{(G)}(x) + i\alpha_3^{(G)} P_3^{(G)}(x) + i\alpha_4^{(G)} P_4^{(G)}(x), & \text{if } b \geqslant x > K_{G-1}, \end{cases}$$

$$\sigma_i^{(j)} \in \mathbb{C}; \quad \alpha_s^{(j)} \in \mathbb{R},$$

where the polynomials $P_s^{(j)}(x)$ satisfy

(1) $P_s^{(j)}(x)$ has parity-$(s \bmod 2)$;
(2) $P_s^{(j)}(x) \in \mathbb{R}[x]$;
(3) For all $x \in [-1, 1]$: $|P_s^{(j)}| < 1$.

Then we can construct a unitary operation U_f which is a block encoding of $f(\hat{x})$ that prepares a 2^n-discrete version of $f(x)$ with probability $P \sim \mathcal{F}$ and resources no greater than

(1) $\mathcal{O}(Q_{\max} n \log_2 n + Gn + Q_{\max} G)$ quantum gates;
(2) $n + \lceil \log_2 G \rceil - 1$ pure ancillas.

The number of workspace ancillas is $\lceil \log_2 n \rceil + 3$; and $Q_{\max} = \max_g Q_g$.

Remark D 1 (Why the costs split as $Q\, n \log n + Gn + Q_{\max} G$). A key point is that all piece oracles $\hat{O}_{f_1}, \hat{O}_{f_2}, \ldots, \hat{O}_{f_G}$ *reuse the same* Amplitude-oracle for the coordinate operator $\hat{O}_x$. Within each $\hat{O}_{f_i}$ (see Fig. 9), the calls to $\hat{O}_x$ appear as a fixed subsequence that does *not* depend on the state of the final indicator qubit. Hence these $\hat{O}_x$ invocations need not be controlled by the indicator; only their *presence* and ordering are fixed per piece.

This observation explains the decomposition of our gate complexity as follows:

(1) $Q_{\max} n \log n$ term. Because the same $\hat{O}_x$ sequence is reused across all pieces, the total cost to implement G alternating phase modulation sequences (Definition C 1) is dominated by repeatedly applying the longest *uncontrolled* coordinate oracle sequence, yielding the $Q_{\max} n \log n$ factor.

(2) Gn term. For piecewise application we execute the segment selector U_c^K once per segment to prepare or unprepare the indicator (see Fig. 9), incurring an additional linear overhead in the number of segments, i.e., Gn.

(3) $Q_{\max} G$ term. The only place where the indicator qubit must control anything is on the *phase angles* of the controlled-phase operator $C_\Phi^{00\cdots 0}$ inside the alternating sequence (Definition C 1). Controlling these angles per segment contributes an additive $Q_{\max} G$ overhead, where $Q_{\max}$ is the largest polynomial degree used by any piece.

[1] R. P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. **21**, 467 (1982).
[2] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, and M. Head-Gordon, Simulated quantum computation of molecular energies, Science **309**, 1704 (2005).
[3] P. C. S. Costa, S. Jordan, and A. Ostrander, Quantum algorithm for simulating the wave equation, Phys. Rev. A **99**, 012323 (2019).
[4] F. Gaitan, Finding flows of a Navier–Stokes fluid through quantum computing, npj Quantum Inf. **6**, 61 (2020).

[5] S. Jin, N. Liu, and Y. Yu, Time complexity analysis of quantum algorithms via linear representations for nonlinear ordinary and partial differential equations, J. Comput. Phys. **487**, 112149 (2023).

[6] N. Linden, A. Montanaro, and C. Shao, Quantum vs. classical algorithms for solving the heat equation, Commun. Math. Phys. **395**, 601 (2022).

[7] S. Jin, N. Liu, and Y. Yu, Quantum simulation of partial differential equations: Applications and detailed analysis, Phys. Rev. A **108**, 032603 (2023).

[8] N. Stamatopoulos, D. J. Egger, Y. Sun, C. Zoufal, R. Iten, N. Shen, and S. Woerner, Option pricing using quantum computers, Quantum **4**, 291 (2020).

[9] J. Gonzalez-Conde, A. Rodríguez-Rozas, E. Solano, and M. Sanz, Efficient Hamiltonian simulation for solving option price dynamics, Phys. Rev. Res. **5**, 043220 (2023).

[10] S. Zhang and L. Li, A brief introduction to quantum algorithms, CCF Transactions on High Performance Computing **4**, 53 (2022).

[11] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, Cambridge, England, 2010).

[12] E. Tang, Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions, Phys. Rev. Lett. **127**, 060503 (2021).

[13] A. M. Childs, On the relationship between continuous- and discrete-time quantum walk, Commun. Math. Phys. **294**, 581 (2010).

[14] A. Scherer, B. Valiron, S.-C. Mau, S. Alexander, E. van den Berg, and T. E. Chapuran, Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2D target, Quantum Inf. Process. **16**, 60 (2017).

[15] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, and S. Kais, Quantum algorithm and circuit design solving the Poisson equation, New J. Phys. **15**, 013021 (2013).

[16] S. K. Leyton and T. J. Osborne, A quantum algorithm to solve nonlinear differential equations, arXiv:0812.4423.

[17] D. W. Berry, Y. Su, C. Gyurik, R. King, J. Basso, A. D. T. Barba, A. Rajput, N. Wiebe, V. Dunjko, and R. Babbush, Analyzing prospects for quantum advantage in topological data analysis, PRX Quantum **5**, 010319 (2024).

[18] A. M. Childs, R. Kothari, and R. D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision, SIAM J. Comput. **46**, 1920 (2017).

[19] Edited by M. Abramowitz and I. A. Stegun, in *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, National Bureau of Standards Applied Mathematics Series Vol. 55 (United States Government Printing Office, Washington, D.C., 1964).

[20] I. F. Araujo, D. K. Park, T. B. Ludermir, W. R. Oliveira, F. Petruccione, and A. J. da Silva, Configurable sublinear circuits for quantum state preparation, Quantum Inf. Process. **22**, 123 (2023).

[21] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature (London) **549**, 195 (2017).

[22] E. Muñoz Coreas and H. Thapliyal, T-count and qubit optimized quantum circuit design of the non-restoring square root algorithm, J. Emerg. Technol. Comput. Syst. **14**, 1 (2018).

[23] T. Häner, M. Roetteler, and K. Svore, Optimizing quantum circuits for arithmetic, arXiv:1805.12445.

[24] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019 (Association for Computing Machinery, New York, 2019), pp. 193–204.

[25] A. M. Dalzell, S. McArdle, M. Berta, P. Bienias, C. Chen, A. Gilyén, C. T. Hann, M. J. Kastoryano, E. T. Khabiboulline, A. Kubica, G. Salton, S. Wang, and F. G. S. L. Brandão, *Quantum Algorithms: A Survey of Applications and End-to-End Complexities* (Cambridge University Press, Cambridge, England, 2025).

[26] A. G. Rattew and B. Koczor, Preparing arbitrary continuous functions in quantum registers with logarithmic complexity, arXiv:2205.00519.

[27] A. N. Soklakov and R. Schack, Efficient state preparation for a register of quantum bits, Phys. Rev. A **73**, 012307 (2006).

[28] F. M. Creevey, C. D. Hill, and L. C. L. Hollenberg, Gasp: A genetic algorithm for state preparation on quantum computers, Sci. Rep. **13**, 11956 (2023).

[29] I. F. Araujo, C. Blank, I. C. S. Araújo, and A. J. da Silva, Low-rank quantum state preparation, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **43**, 161 (2024).

[30] M. Rosenkranz, E. Brunner, G. Marin-Sanchez, N. Fitzpatrick, S. Dilkes, Y. Tang, Y. Kikuchi, and M. Benedetti, Quantum state preparation for multivariate functions, Quantum **9**, 1703 (2025).

[31] J. Gonzalez-Conde, T. W. Watts, P. Rodriguez-Grasa, and M. Sanz, Efficient quantum amplitude encoding of polynomial functions, Quantum **8**, 1297 (2024).

[32] A. G. Rattew and P. Rebentrost, Non-linear transformations of quantum amplitudes: Exponential improvement, generalization, and applications, arXiv:2309.09839.

[33] N. Guo, K. Mitarai, and K. Fujii, Nonlinear transformation of complex amplitudes via quantum singular value transformation, Phys. Rev. Res. **6**, 043227 (2024).

[34] L. K. Grover and T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions, arXiv:quant-ph/0208112.

[35] S. McArdle, A. Gilyén, and M. Berta, Quantum state preparation without coherent arithmetic, arXiv:2210.14892.

[36] H. Li, H. Ni, and L. Ying, On efficient quantum block encoding of pseudo-differential operators, Quantum **7**, 1031 (2023).

[37] N. M. Guseynov, A. A. Zhukov, W. V. Pogosov, and A. V. Lebedev, Depth analysis of variational quantum algorithms for the heat equation, Phys. Rev. A **107**, 052422 (2023).

[38] M. Plesch and I. C. V. Brukner, Quantum-state preparation with universal gate decompositions, Phys. Rev. A **83**, 032302 (2011).

[39] V. Bergholm, J. J. Vartiainen, M. Möttönen, and M. M. Salomaa, Quantum circuits with uniformly controlled one-qubit gates, Phys. Rev. A **71**, 052330 (2005).

[40] V. V. Shende, I. L. Markov, and S. S. Bullock, Minimal universal two-qubit controlled-not-based circuits, Phys. Rev. A **69**, 062321 (2004).

[41] Y. Dong, X. Meng, K. B. Whaley, and L. Lin, Efficient phase-factor evaluation in quantum signal processing, Phys. Rev. A **103**, 042419 (2021).

[42] Y. Dong, L. Lin, H. Ni, and J. Wang, Infinite quantum signal processing, Quantum **8**, 1558 (2024).

[43] Uchuang community, PYQSP: PYTHON quantum signal processing phase-angle solver (2025), https://github.com/ichuang/pyqsp.