# Learning PDEs for Portfolio Optimization with Quantum Physics-Informed Neural Networks

Letao Wang, Abdel Lisser, Sreejith Sreekumar, Zeno Toffano

# Learning PDEs for Portfolio Optimization with Quantum Physics-Informed Neural Networks

Letao Wang*, Abdel Lisser, Sreejith Sreekumar, Zeno Toffano

Laboratory of Signals and Systems, CentraleSupélec, CNRS, Paris-Saclay University, 3 Rue Joliot Curie, Gif-sur-Yvette, 91190, Île-de-France, France.

*Corresponding author(s). E-mail(s): letao.wang@centralesupelec.fr;

Contributing authors: abdel.lisser@centralesupelec.fr; sreejith.sreekumar@centralesupelec.fr; zeno.toffano@centralesupelec.fr;

**Abstract**

Partial differential equations (PDEs) play a crucial role in financial mathematics, particularly in portfolio optimization, and solving them using classical numerical or neural network methods has always posed significant challenges. Here, we investigate the potential role of quantum circuits for solving PDEs. We design a parameterized quantum circuit (PQC) for implementing a polynomial based on tensor rank decomposition, reducing the quantum resource complexity from exponential to polynomial when the corresponding tensor rank is moderate. Building on this circuit, we develop a Quantum Physics-Informed Neural Network (QPINN) and a Quantum-inspired PINN, both of which guarantee the existence of an approximation of the PDE solution, and this approximation is represented as a polynomial that incorporates tensor rank decomposition. Despite using 80 times fewer parameters in experiments, our quantum models achieve higher accuracy and faster convergence than a classical fully connected PINN when solving the PDE for the Merton portfolio optimization problem, which determines the optimal investment fraction between a risky and a risk-free asset. Our quantum models further outperform a classical PINN constructed to share the same inductive bias, providing experimental evidence of quantum-induced improvement and highlighting a resource-efficient pathway toward classical and near-term quantum PDE solvers.

**Keywords:** Quantum machine learning, Quantum-inspired machine learning, Quantum Physics-Informed Neural Network, Portfolio optimization.

# 1 Introduction

In financial mathematics, Partial Differential Equations (PDE) are fundamental to modeling and valuation of financial derivatives, risk management, and optimal portfolio strategies. A cornerstone in this domain is the Merton portfolio optimization problem (Merton 1969), which aims to determine an optimal investment strategy that maximizes the expected utility of the terminal wealth of an investor under stochastic market dynamics. Directly finding the optimal strategy is often difficult due to the randomness of dynamics, so it is commonly transformed into a Hamilton–Jacobi–Bellman (HJB) PDE, which characterizes both the maximum expected utility and the corresponding optimal investment strategy. Therefore, the Merton portfolio optimization problem can be solved by finding the solution to its corresponding HJB PDE (Gonzalo et al. 2025; Di Persio and Fraccarolo 2025; Wang and Hu 2022; Kilianová and Trnovska 2014; Gollier 2001).

Classical numerical PDE solvers, such as the finite difference method (FDM) and the finite element method (FEM) can be used to approximate PDE solutions. However, achieving high accuracy with these traditional deterministic methods often requires a significant computational cost (Luo et al. 2025). To address this limitation, non-deterministic algorithms have also been explored. For example, Physics-Informed Neural Networks (PINNs) have emerged as a promising mesh-free alternative that uses deep learning to solve PDEs by embedding physical laws into the loss function (Raissi et al. 2019). Despite their promise, PINNs also face several limitations, such as slow convergence, difficulty capturing high-frequency features, and scalability issues in high-dimensional problems (Luo et al. 2025). These limitations of classical PDE solvers motivate the exploration of a new computational paradigm as a possible route to solve PDEs more efficiently.

Quantum computing, with its ability to encode and process information in exponentially large-dimensional Hilbert spaces, offers a new frontier for accelerating scientific computing tasks. It has been considered as a new computational model with the potential to solve problems beyond the reach of classical computers. Although the construction of practical quantum hardware remains challenging, theoretical results have shown that quantum algorithms can achieve significant complexity advantages over classical methods, for example factoring (Shor 1997) and solving linear systems (Harrow et al. 2009) to address linear ordinary differential equations (Berry et al. 2017). Such results encourage people to investigate the potential of quantum computers for solving PDEs, which can be broadly categorized into two types (Hunout et al. 2025): fully quantum algorithms that directly encode differential equations within quantum circuits and hybrid quantum-classical algorithms that integrate quantum components into classical optimization loops.

The first fully quantum algorithms for differential equations were encoded by vector amplitudes built on the quantum amplitude amplification algorithm (QAAA) and the quantum amplitude estimation algorithm (QAEA) (Brassard et al. 2002). These methods typically applied differentiation schemes over discretized variable spaces. Subsequently, the Kacewicz quantum algorithm (Kacewicz 2006) was proposed to solve initial value problems (IVP), i.e., ordinary differential equations (ODEs) with initial conditions using the QAAA and QAEA algorithms (Brassard et al. 2002). Moreover, certain works extended the Kacewicz quantum algorithm to differential equations, including the Navier–Stokes equations (Gaitan 2020), the Burgers equation (Oz et al. 2022), and the heat equation (Oz et al. 2023).

However, fully quantum algorithms for solving PDEs typically require fault-tolerant quantum hardware and efficient quantum random access memory (qRAM) (Giovannetti et al. 2008), which remain infeasible in the near term (Matteo et al. 2020). This has motivated the development of hybrid quantum–classical approaches. In parallel, quantum machine learning (QML) has emerged as a broad research field between quantum computing and machine learning (Biamonte et al. 2017). A popular approach to QML is based on parameterized quantum circuits (PQCs) (Benedetti et al. 2019), whose training can be formulated as variational quantum algorithms (VQAs) (Cerezo et al. 2021) within the hybrid approach, making them suitable for near-term quantum devices that lack full fault tolerance or error correction capabilities.

Building on the idea of VQAs, Kyriienko et al. (2021) proposed differentiable quantum circuits (DQCs) as a framework for solving PDE, where a PQC is trained so that its expected measurement value serves as the output of the model, thus approximating the solution while minimizing the loss and enforcing the target PDE. Inspired by DQCs and the success of classical PINNs, several groups extended the DQC framework into Quantum Physics-Informed Neural Networks (QPINNs) (Markidis 2022; Trahan et al. 2024; Hegde and Markidis 2024; Berger et al. 2025; Panichi et al. 2026; Farea et al. 2025; Chen et al. 2026; Lantigua et al. 2026), which can be seen as the quantum analog of PINNs. QPINNs can be implemented on both continuous-variable (Markidis 2022; Panichi et al. 2026) and discrete-variable (qubit-based) (Berger et al. 2025; Trahan et al. 2024) architectures, and the discrete-variable versions are sometimes referred to as VQAs for PDEs. In this work, we focus on the discrete-variable architecture.

Precisely, QPINN research (Hegde and Markidis 2024; Berger et al. 2025) employs a quantum model known as the quantum fourier model (QFM) or re-uploading model, originally introduced in Schuld et al. (2021). In this approach, the circuit is made up of alternating encoding and trainable layers. The function induced by the quantum model, also called the *hypothesis function* (the outputs of the hypothesis function are identical to those of the quantum model), can be expressed as a Fourier series. By varying all circuit parameters, these hypothesis functions collectively form the *hypothesis space* of the model. Other QPINN models can also employ hypothesis functions based on Chebyshev polynomials (Kyriienko et al. 2021) or Lagrange polynomials (Hunout et al. 2025). Moreover, QPINN with QFM (Hegde and Markidis 2024; Berger et al. 2025) and other VQAs designed to solve PDEs (Kyriienko et al. 2021; Hunout et al. 2025) often employ a hardware efficient ansatz (HEA) (Kandala et al. 2017) as the

training block. In practice, this introduces hidden constraints and makes it difficult to precisely characterize the hypothesis space from a theoretical perspective. Such limitations highlight a broader issue in VQA: most quantum models offer little theoretical guarantees that their hypothesis space contains an appropriate approximation of the target function (Chang and Cerezo 2025).

To better understand and characterize the hypothesis space, we turn to the quantum signal processing (QSP) framework (Gilyén et al. 2019), where hypothesis functions can be expressed as univariate polynomials. Subsequently, the work in Yu et al. (2024) extended QSP to multivariate settings, and this analysis shows that the quantum circuit resource complexity (depth, number of parameters) grows exponentially with the the degree of the polynomial, making the approach impractical.

To address these limitations, we introduce an efficient QPINN model whose hypothesis space provably contains a *tensor-decomposed* polynomial that accurately approximates the target function, where tensor-decomposed (Kolda and Bader 2009) refers to representations that can be expressed as sums of products of univariate components, with the number of summation terms corresponding to the *tensor rank*. In fact, analytical solutions of many types of PDEs such as the Heat equation, the Laplace equation, the time-dependent Schrödinger equation, and the PDEs for portfolio optimization can be considered as tensor-decomposed structures, meaning that these solutions can be expressed as a sum of products of univariate functions (Logan 2015). By further approximating each univariate function with a univariate polynomial, our proposed model effectively exploits this structural property, thereby reducing the quantum resource complexity for representing PDE solutions, from exponential to polynomial complexity when the tensor rank is not large. The tensor-decomposed structure serves as an inductive bias, making our QPINN particularly suitable for PDEs whose solutions admit such a decomposition. In addition to the quantum model, we also propose a quantum-inspired PINN obtained by a slight modification of the QPINN architecture, whose hypothesis functions can be tensor-decomposed polynomials and can be simulated efficiently on classical computers.

As we discussed before, the hypothesis space of previous QPINNs often consists of Fourier series (or other polynomials) with HEAs, the actual hypothesis space is not the full set of Fourier series of a fixed degree with arbitrary coefficients, making the practical hypothesis space smaller than the ideal functional family. This effectively provides an upper bound on the hypothesis space size. In contrast, we consider the opposite perspective. The hypothesis space of our QPINN contains all tensor-decomposed polynomials of a fixed degree and bounded tensor rank, meaning that all possible coefficient choices are representable. As a result, the minimal distance (i.e., approximation error) between the target function and the hypothesis space can be explicitly estimated when the target function satisfies the required regularity conditions (e.g., continuity). Although the actual hypothesis space can be much larger than the set of tensor-decomposed polynomials, this set provides a lower bound on the hypothesis space size, thereby guaranteeing the existence of an approximation to the PDE solution within the QPINN hypothesis space.

## Related works

Similar ideas to our QPINN and quantum-inspired PINN have also been explored on the classical side. Polynomial hypothesis functions have been studied in PINNs (Tang et al. 2023), while tensor-decomposition techniques have been incorporated into classical neural networks (Cohen et al. 2016; Chrysos et al. 2021). Our approaches share conceptual similarities with the framework proposed in (Vemuri et al. 2025), where the classical PINN learns each separable component of the tensor-decomposed hypothesis functions in order to efficiently approximate PDE solutions.

Some researches have demonstrated that orthogonal polynomials, such as the Chebyshev polynomials, present some theoretical advantages over the standard non-orthogonal polynomials for solving PDEs (Powell 1981; Boyd 2001). One quantum approach has also employed Chebyshev polynomials as hypothesis functions for solving PDEs (Kyriienko et al. 2021).

However, it was experimentally observed in Tang et al. (2023) that standard non-orthogonal polynomials achieve significantly faster training convergence than Chebyshev polynomials, while exhibiting a comparable error after convergence when solving low-dimensional PDEs such as the Burgers equation, Sine-Gordon, and Allen-Cahn equations. Standard non-orthogonal, Chebyshev, and tensor-decomposed polynomial represent particular subclasses within the general family of polynomial hypothesis functions, and it is still not well understood which hypothesis function representation is most appropriate for solving particular classes of PDEs. This observation motivates our investigation into quantum models whose hypothesis spaces are designed to contain polynomial hypothesis functions, as outlined here after:

## Main contribution

- We provide a detailed analysis of the quantum resource cost for implementing univariate, multivariate, and tensor-decomposed polynomials, and compare our results with existing approaches. While the implementation of a general multivariate polynomial requires exponentially large cost complexity, our tensor-decomposed model reduces the cost complexity to polynomial when the corresponding tensor rank is not very large, making the implementation on near-term quantum hardware more feasible.
- We incorporate the tensor-decomposed model to propose a QPINN and a Quantum-inspired PINN. Both models guarantee the existence of a tensor-decomposed polynomial approximating the PDE solution, with the approximation error depending on the PDE and the model parameters. Owing to entanglement, the QPINN has a richer hypothesis space, providing greater expressivity than classical models.
- We conduct numerical simulations on PDEs in portfolio optimization, comparing our QPINN and quantum-inspired PINN with classical PINNs. In particular, we include a classical counterpart PINN that shares the same tensor-decomposed inductive bias as our quantum models, as well as a fully connected PINN with 80 times larger parameters, and demonstrate that our QPINN and Quantum-inspired PINN outperforms classic PINNs experimentally.

This paper is organized as follows. We first introduce the context of the Merton portfolio optimization problem in Section 2.1. The general framework of PINNs and QPINNs is presented in Section 2.2, the polynomials considered in our quantum models are described in Section 2.3. Section 3.1 summarizes the main results on the quantum resource complexity required to implement general and tensor-decomposed polynomials, while Sections 3.2 and 3.3 provide the detailed descriptions. We then present our our QPINN and our Quantum-inspired PINN in Section 3.4. Finally, Section 4 reports the QPINN and PINN experimental simulations, and Section 5 concludes the paper.

# 2 Background

In this section, we present the Merton portfolio optimization problem, how it can be solved within the QPINN framework, and define the corresponding polynomials needed.

## 2.1 Merton Portfolio Optimization

The Merton portfolio optimization problem (Merton 1969) studies how an investor should allocate his wealth with a choice between a risk-free asset (e.g., money deposited in a bank account) with a fixed interest rate $r(t)$ and a risky asset (e.g., a stock) with expected return $\mu(t)$ and volatility $\sigma(t)$. At each time $t$, the investor chooses a fraction $\alpha(t)$ of his wealth $X(t)$ to invest in the risky asset, while the remaining fraction $1-\alpha(t)$ is placed in the risk-free asset. The objective is to maximize the expected utility of the terminal wealth, which corresponds to finding the function

$$v(t,x) = \sup_{\alpha(t)} \mathbb{E}\Big[U\big(X(T)\big) \,\Big|\, X(t) = x\Big], \tag{1}$$

where $U(x)$ is the investor's utility function that encodes risk preferences, $v(t,x)$ is the maximum expected utility that can be achieved by following the optimal strategy from the current time $t$, with wealth $x$, up to the terminal time $T$. In the case of the Merton portfolio optimization model without jumps and with constant parameters, once the value function $v(t,x)$ is known, the optimal investment fraction $\hat{\alpha}(t)$ can be obtained directly, allowing the investor to achieve the best possible growth of wealth.

The dynamics of $X(t)$ follows a stochastic differential equation (SDE), which is often challenging to solve directly. Hence, one typically converts the problem into a PDE problem. In general settings, analytical solutions are unavailable, and we instead obtain the solution to the Merton portfolio problem by numerically solving the corresponding PDE. But in our setting, we can make the following assumptions:

- The expected return $\mu$, volatility $\sigma$, and fixed interest rate $r$ are constants, with $\mu > r$.
- The investor's utility function is $U(x) = \frac{x^\gamma}{\gamma}, x > 0$ with a risk parameter $\gamma \in (0,1)$.

The investor's attitude towards risk determines the curvature of the utility function $U(x)$, and $\gamma \in (0,1)$ indicates risk aversion and implies a concave utility function

(Gollier 2001). Under these assumptions, the function $v(t,x)$ satisfies the Hamilton–Jacobi–Bellman (HJB) PDE on the domain $(t,x) \in [0,T] \times [0,+\infty)$:

$$\begin{cases} \partial_t v(t,x) \partial_x^2 v(t,x) + \partial_x v(t,x) \partial_x^2 v(t,x) rx - \frac{1}{2} \left(\frac{\mu - r}{\sigma}\right)^2 \partial_x v(t,x)^2 = 0, \\ v(T,x) = \frac{x^\gamma}{\gamma}, \\ v(t,1) = \exp(-k(T-t)) \frac{1}{\gamma}, \end{cases} \tag{2}$$

where $k = \frac{1}{2} \frac{\gamma}{\gamma - 1} \left(\frac{\mu - r}{\sigma}\right)^2 - r\gamma$. The analytical solution is given by

$$v(t,x) = \exp(-k(T-t)) \frac{x^\gamma}{\gamma}, \tag{3}$$

and the optimal investment fraction is given by $\hat{\alpha} = \frac{1}{1-\gamma} \frac{\mu - r}{\sigma^2}$. Note that the boundary conditions $v(T,x), v(t,1)$ in Equation (2) are not unique; other valid boundary conditions may also be chosen. See Appendix B for detailed derivation and explanation.

Portfolio optimization problem can alternatively adopt machine learning based approaches that directly learn the optimal fraction through empirical utility maximization, without deriving the HJB equation as shown in a recent study (Kopeliovich and Pokojovy 2024). In this work we focus on the HJB PDE method. The following section introduces the QPINNs framework for addressing general PDEs, with a specific application to the HJB PDE.

## 2.2 Quantum/Classical PINNs framework

A general PDE can be typically expressed in the following form:

$$\begin{aligned} D_x(u;\lambda) &= h(x), \quad \forall x \in \Omega \subset \mathbb{R}^d \\ B_k(u) &= g_k(x), \quad \forall x \in \partial\Omega \subset \mathbb{R}^d, \quad k = 1, 2, \ldots, n_b \end{aligned} \tag{4}$$

where

- $\Omega$ is a domain with boundary $\partial\Omega$.
- $u : \Omega \cup \partial\Omega \to \mathbb{R}$ is the unknown solution.
- $D_x(\cdot;\lambda)$ is a (possibly parameterized) differential operator with parameters $\lambda$.
- $h : \Omega \to \mathbb{R}$ is a given source function.
- $B_k(\cdot)$ is the $k^{\text{th}}$ boundary condition.
- $g_k : \partial\Omega \to \mathbb{R}$ are given boundary functions.
- $n_b$ is the number of given boundary functions.

Given a general PDE defined in Equation (4), for all $k$, let $\left\{x_b^{(i)}, g_k\left(x_b^{(i)}\right)\right\}_{i \in [1, N_b]}$ and $\left\{x_d^{(i)}\right\}_{i=1}^{N_d}$ be the sets of randomly selected boundary points and residual points for training, respectively. These points are usually drawn from an unknown distribution.

Consider a computational model $\mathcal{Q}(\boldsymbol{\theta})$ that is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^m$, typically represented by either a quantum or a classical neural network, where the quantum implementation is referred to as a quantum model. We define the hypothesis space:

$$\mathcal{F}_{\mathcal{Q}} = \left\{ f_{\mathcal{Q}(\boldsymbol{\theta})} : \Omega \cup \partial\Omega \to \mathbb{R} \right\}_{\boldsymbol{\theta} \in \mathbb{R}^m},$$

where $f_{\mathcal{Q}(\boldsymbol{\theta})}$ denotes the function induced by the computational model $\mathcal{Q}(\boldsymbol{\theta})$, referred to as the hypothesis function that approximates the true solution $u$. The hypothesis function $f_{\mathcal{Q}(\boldsymbol{\theta})}$ represents the output of the computational model $\mathcal{Q}(\boldsymbol{\theta})$ as a function of the input domain $\Omega \cup \partial\Omega$ and the trainable parameters $\boldsymbol{\theta}$.

Then we define the loss function as

$$\mathcal{L}_{\text{total}}(\boldsymbol{\theta}) = \sum_{k=1}^{n_d} \mathcal{L}_{k,b}(\boldsymbol{\theta}) + \mathcal{L}_d(\boldsymbol{\theta}), \tag{5}$$

where boundary loss values $\mathcal{L}_{k,b}(\boldsymbol{\theta})$ and PDE residual loss value $\mathcal{L}_d(\boldsymbol{\theta})$ are defined as

$$\begin{aligned} \mathcal{L}_{k,b}(\boldsymbol{\theta}) &= \frac{W_k}{N_b} \sum_{i=1}^{N_b} \left| g_k(x_b^{(i)}) - f_{\mathcal{Q}(\boldsymbol{\theta})}\left(x_b^{(i)}\right) \right|^2 \\ \mathcal{L}_d(\boldsymbol{\theta}) &= \frac{W_d}{N_d} \sum_{i=1}^{N_d} \left| D_x\left(f_{\mathcal{Q}(\boldsymbol{\theta})}\left(x_d^{(i)}\right); \lambda\right) - h\left(x_d^{(i)}\right) \right|^2, \end{aligned} \tag{6}$$

with loss weight parameters $W_k, W_d > 0, \forall k \in [1, n_b]$. Specifically, the loss function for the HJB PDE in Equation (2) is defined as:

$$\mathcal{L}_{\text{total}}(\boldsymbol{\theta}) = \mathcal{L}_{1,b}(\boldsymbol{\theta}) + \mathcal{L}_{2,b}(\boldsymbol{\theta}) + \mathcal{L}_d(\boldsymbol{\theta}), \tag{7}$$

with $W_d, W_1, W_2 > 0$, we have

$$\begin{cases} \mathcal{L}_d(\boldsymbol{\theta}) = \frac{W_d}{N_d} \sum_{i=1}^{N_d} \left( \partial_t v(t,x) \partial_x^2 v(t,x) + \partial_x v(t,x) \partial_x^2 v(t,x) rx - \frac{1}{2} \left(\frac{\mu - r}{\sigma}\right)^2 \partial_x v(t,x)^2 \right)^2, \\ \mathcal{L}_{1,b}(\boldsymbol{\theta}) = \frac{W_1}{N_b} \sum_{i=1}^{N_b} \left( f_{\mathcal{Q}}(T, x; \boldsymbol{\theta}) - \frac{x^\gamma}{\gamma} \right)^2, \\ \mathcal{L}_{2,b}(\boldsymbol{\theta}) = \frac{W_2}{N_b} \sum_{i=1}^{N_b} \left( f_{\mathcal{Q}}(t, 1; \boldsymbol{\theta}) - e^{-k(T-t)} \frac{1}{\gamma} \right)^2 \text{ where } k = -r\gamma + \frac{1}{2} \frac{\gamma}{\gamma - 1} \left(\frac{\mu - r}{\sigma}\right)^2. \end{cases} \tag{8}$$

With this in mind, we define QPINNs as follows:

**Definition 1** (Quantum Physics-Informed Neural Networks) Given a PDE, we consider a quantum model $\mathcal{Q}(\boldsymbol{\theta})$ defined on the PDE domain, whose hypothesis function is denoted by $f_{\mathcal{Q}(\boldsymbol{\theta})}$, and define $\mathcal{F}_{\mathcal{Q}}$ as the corresponding hypothesis space. A Quantum Physics-Informed Neural Network (QPINN) seeks an approximation $f_{\mathcal{Q}(\boldsymbol{\theta}^*)} \in \mathcal{F}_{\mathcal{Q}}$ of the true solution $u$ by solving the optimization problem

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \mathcal{L}_{\text{total}}(\boldsymbol{\theta}), \tag{9}$$

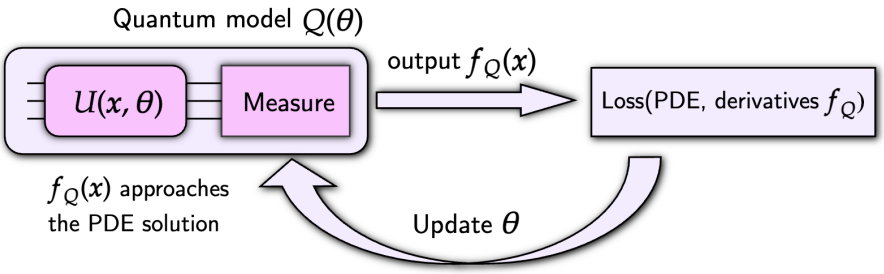


**Fig. 1** Workflow of Quantum Physics-Informed Neural Networks (QPINNs).

where $\mathcal{L}_{\text{total}}(\boldsymbol{\theta})$ is the loss function defined in Equations (5) and (6). We refer to this framework as QPINNs.

Replacing the quantum model with a classical one recovers the conventional PINN framework. In this study, we focus primarily on quantum models. We can define quantum models $\mathcal{Q}(\boldsymbol{\theta})$ on $n$ qubits as

$$\mathcal{Q}(\boldsymbol{\theta}) = \big(U(\boldsymbol{x}, \boldsymbol{\theta}), O\big),$$

where the $2^n$-dimensional unitary $U$ is denoted as a parameterized quantum circuit (PQC), with the vector of trainable parameters $\boldsymbol{\theta} \in \mathbb{R}^m$, the classical data vector $\boldsymbol{x} = (x_1, \ldots, x_D) \in X \subset \mathbb{R}^D$, and $O$ is a Hermitian observable. We have the hypothesis function generated by $\mathcal{Q}(\boldsymbol{\theta})$ as

$$f_{\mathcal{Q}(\boldsymbol{\theta})}(\boldsymbol{x}) = \langle 0|U(\boldsymbol{x}, \boldsymbol{\theta})^\dagger O U(\boldsymbol{x}, \boldsymbol{\theta})|0\rangle.$$

The workflow of QPINNs is shown in Figure 1: we begin by choosing a quantum model $\mathcal{Q}(\boldsymbol{\theta})$ consisting of a PQC $U$ and an observable $O$, which together define the hypothesis function $f_{\mathcal{Q}(\boldsymbol{\theta})}$. Using the parameter shift rule (Mitarai et al. 2018), we obtain estimated derivatives of $f_{\mathcal{Q}(\boldsymbol{\theta})}$. From these derivatives and guided by the PDE we want to solve, we evaluate a loss function defined in Equations (5) and (6), then apply gradient descent or other algorithms to update the circuit parameters $\boldsymbol{\theta}$. In principle, each update brings the model's output closer to the true solution of the PDE.

The Weierstrass approximation theorem guarantees that any smooth function can be approximated by a polynomial, and the approximation of multivariate functions is ensured by its multivariate extension (Nikol'skii 2013). From Yu et al. (2024), we know that with suitable choices of $\mathcal{Q}(\boldsymbol{\theta})$ and under ideal (noise-free and sufficiently large) conditions, any multivariate polynomial can be implemented. This implies that any continuous target function can be approximated to arbitrary accuracy by a quantum model employing polynomial hypothesis functions, or by a model whose hypothesis space contains arbitrary polynomials. However, implementing general polynomials

requires exponentially deep circuits (Yu et al. 2024), and running such circuits on current noisy intermediate-scale quantum (NISQ) hardware is infeasible due to limited coherence times, gate fidelities, and qubit counts (Preskill 2018). Hence, we seek an efficient method to express suitable polynomials within a quantum model. In the following section, we present several classes of polynomials, including the tensor-decomposed polynomials, and describe how they can be implemented using quantum models.

## 2.3 Polynomials variants definition

We introduce two forms of polynomials: the general polynomial and its tensor-decomposed variant, highlighting how the latter enables efficient quantum implementation.

**Definition 2** (Polynomial) A real univariate polynomial with degree $L$, coefficient $c_n \in \mathbb{R}$, index $n \in [L]$ is defined as $p(x) = \sum_n c_n x^n$. A real multivariate polynomial $p(\boldsymbol{x})$ with $D$ variables and degree at most $L$ in each variable is defined as

$$p(\boldsymbol{x}) := \sum_{\boldsymbol{n}} c_{\boldsymbol{n}} \boldsymbol{x}^{\boldsymbol{n}}, \tag{10}$$

where $\boldsymbol{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D, \boldsymbol{n} = (n_1, \ldots, n_D) \in [L]^D, c_{\boldsymbol{n}} \in \mathbb{R}$ and $\boldsymbol{x}^{\boldsymbol{n}} = x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$.

We define polynomials with degree at most $L$ in each variable as $\|\boldsymbol{n}\|_\infty = \max_i n_i \leq L$, polynomials with total degree at most $L$ as $\|\boldsymbol{n}\|_1 = \sum_i n_i \leq L$, and $[N] := \{0, \cdots, N\}$ for $N \in \mathbb{Z}^+$. In addition to this ordinary form, we can apply the *tensor rank decomposition* (Kolda and Bader 2009) to the polynomial coefficients such that

$$c_{\boldsymbol{n}} = c_{n_1,\ldots,n_D} = \sum_{r=1}^{R} \lambda_r c_{r,n_1} c_{r,n_2} \cdots c_{r,n_D}, \tag{11}$$

where $c_{r,n_j} \in \mathbb{R}$, $n_j \in [L]$, $\forall r \in [R], j \in [D]$, and the tensor rank $R \in [1, (L+1)^D]$. By combining Equations (10) and (11), we obtain the following definition:

**Definition 3** (Tensor-Decomposed Polynomial) Let $p(\boldsymbol{x})$ be a real multivariate polynomial with $D$ variables and degree at most $L$ in each variable, it can also be written as

$$p(\boldsymbol{x}) = \sum_{r=1}^{R} \lambda_r \prod_{j=1}^{D} p_{r,j}(x_j), \tag{12}$$

where each $p_{r,j}(x_j) = \sum_{n_j} c_{r,n_j} x_j^{n_j}$ is the univariate polynomial, tensor rank $R \in [1, (L+1)^D]$, $\lambda_r \in \mathbb{R}$, $c_{r,n_j} \in \mathbb{R}$, $n_j \in [L], \forall r \in [R], j \in [D]$. We call such $p(\boldsymbol{x})$ a tensor-decomposed polynomial (TD polynomial).

See details in Appendix A. In practice, a PDE solution can often be well-approximated by a hypothesis of relatively low rank $R$. One may fix $R$ in advance and then train

to obtain a solution within an acceptable error, which corresponds to the notion of canonical polyadic (CP) decomposition.

# 3 Main results

Having defined the polynomial and its tensor-decomposed form, we discuss the quantum circuit implementation of them. We first present a summary table outlining our main theoretical results, and then provide detailed derivations and explanations in the subsequent sections.

## 3.1 Polynomials resources summary

| Polynomial form | Width | Depth | Number of parameters | Reference |
|---|---|---|---|---|
| Univariate polynomial with $\lvert n\rvert \leq L$ | 3 | $O(L)$ | $2L+1$ | Our Proposition 1 |
| Multivariate polynomial with $\lVert\boldsymbol{n}\rVert_1 \leq L$ | $O(D+\log L+L\log D)$ | $O\left(L^2D^L(\log L+L\log D)\right)$ | $O\left(LD^L(L+D)\right)$ | Yu et al. (2024, Theorem 1) |
| Multivariate polynomial with $\lVert\boldsymbol{n}\rVert_\infty \leq L$ | $O(D\log L)$ | $O\left(D^2L^{D+1}\log L\right)$ | $O(DL^{D+1})$ | Our Theorem 2 |
| TD polynomial with $R \in [1,(L+1)^D]$ | $2D+\lceil\log R\rceil+1$ | $O(RDL\log R)$ | $O\left(RDL\right)$ | Our Theorem 3 |

**Table 1** Summary of the quantum model resources required for implementing polynomials presented in Definition 2 and 3.

The table 1 summarizes both known complexity bounds (Yu et al. 2024) and our own results (marked as propositions/theorems in later sections). Specifically, "Width" refers to the number of qubits required (i.e., the quantum circuit register size); "Depth" denotes the circuit depth under the native gate set, which includes single-qubit gates and CNOT gates; "Number of parameters" indicates the total number of trainable circuit parameters. Besides, the quantum model output is always real and its absolute value is bounded by 1 since it is defined through the expectation values of Hermitian observables whose eigenvalues lie in $[-1,1]$. A scaling factor is typically applied to ensure accurate approximation of target functions, see the references in table for details.

It should be noted that the required resource complexity for different quantum computer platforms could be different. For some types of quantum computer, such as those based on the neutral atom platform, multi-controlled rotation gates can be implemented directly and relatively efficiently (Li et al. 2021). In this case, they are considered as "native gates" for neutral atom quantum computers. In contrast, for platforms such as superconducting quantum computers, multi-controlled rotation gates must be decomposed into a large number of simpler "native gates" such as CNOT and single-qubit gates. It is important to note that the cost of implementing the same quantum gate can vary significantly across different quantum computing platforms, therefore we analyze the implementation cost complexity of quantum models from the perspective of different native gate sets. In this work, we consider cases where multi-controlled gates are treated as native gates only when explicitly stated. Otherwise, the native gate set consists of single-qubit gates and CNOT gates (e.g., Table 1).

The quantum model resources needed depend on different polynomial forms according to their algebraic structure (ordinary or tensor-decomposed form) and their norm constraints ($\|\boldsymbol{n}\|_1 \leq L$ or $\|\boldsymbol{n}\|_\infty \leq L$). The choice of $\|\boldsymbol{n}\|_1$ norm constraint leads to combinatorial growth $O(D^L)$, while the $\|\boldsymbol{n}\|_\infty$ norm constraint produces $O(L^D)$ terms, changing the complexity in both quantum circuit depth and number of parameters. Moreover, for polynomials that can exploit tensor rank decomposition to factorize multivariate coefficients into a sum of products of univariate coefficients in Equation (11), we can significantly optimize the complexity with respect to $D$ compared to Yu et al. (2024, Theorem 1), as both the circuit depth and the number of parameters are reduced from exponential to polynomial complexity when the tensor rank is not large. In the following sections 3.2 and 3.3, we present the detailed descriptions of the results summarized in Table 1.

## 3.2 Polynomial implementation

In this subsection, we present how to implement polynomials within quantum models.

### 3.2.1 Univariate case

We first introduce the quantum model for implementing real univariate polynomials, based on the lemmas of QSP (Gilyén et al. 2019). Given an integer $L$ and parameter $\boldsymbol{\theta} \in \mathbb{R}^{L+1}$, with input $x \in [-1, 1]$, define the parameterized unitary

$$U_{\boldsymbol{\theta}}(x) := R_z(\theta_L) \prod_{j=0}^{L-1} R_x(-2\arccos(x)) R_z(\theta_j), \tag{13}$$

that is shown in Figure 2. Here, $R_z(\theta)$ and $R_x(\theta)$ denote single-qubit rotation gates about the Pauli-Z and Pauli-X axes, respectively, each parameterized by the rotation angle $\theta$.

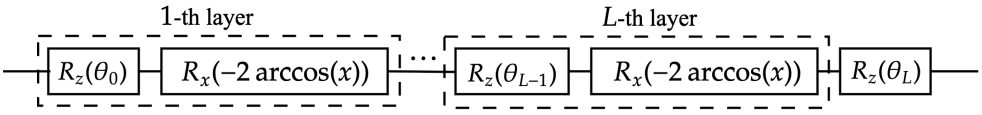


**Fig. 2** Circuit of $U_{\boldsymbol{\theta}}(x)$. Since the circuit has a alternating structure, we consider it to comprise $L$ layers, each consisting of $R_x(-2\arccos(x))$ and $R_z(\theta_j)$, followed by an additional $R_z(\theta_L)$.

We then propose the quantum resource bound analysis for the implementation of arbitrary real univariate polynomial under two different quantum native gate sets:

**Proposition 1** (Quantum circuit for univariate polynomial) *For any real polynomial* $p(x) \in \mathbb{R}[x]$ *that satisfies* $\deg(p(x)) \leq L$ *and* $\forall x \in [-1, 1], |p(x)| \leq \frac{1}{2}$, *there exists a quantum model* $\mathcal{Q}$ *that consists of a PQC* $W_p(\boldsymbol{x})$ *and an observable* $Z^{(0)}$ *such that*

$$f_{\mathcal{Q}}(x) := \langle 0|W_p^\dagger(x) Z^{(0)} W_p(\boldsymbol{x})|0\rangle = p(x),$$

*where $Z^{(0)}$ is the Pauli Z observable on the first qubit. The width of the PQC is at most 3, the number of parameters is at most $2L+1$. The PQC $W_p(\boldsymbol{x})$ can be expressed as at most $4L$ double-controlled rotation gates and 4 Hadamard gates, with depth $4L+2$. Alternatively, it can be expressed using $36L$ single-qubit gates and $32L$ CNOT gates, with depth $60L-5$.*

The circuit diagram of $W_p(\boldsymbol{x})$ is shown in Figure 3.

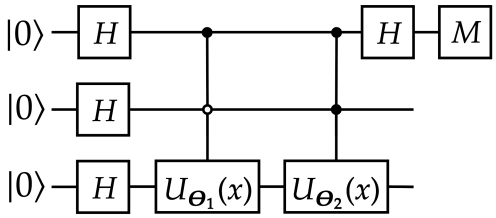


**Fig. 3** The quantum model of Proposition 1 can be considered as the Hadamard test to estimate $p(x) = \langle+|^{\otimes 2}(|0\rangle\langle 0|\otimes U_{\boldsymbol{\theta_1}}(x) + |1\rangle\langle 1|\otimes U_{\boldsymbol{\theta_2}}(x))|+\rangle^{\otimes 2}$ where $U_{\boldsymbol{\theta_1}}(x), U_{\boldsymbol{\theta_2}}(x)$ are defined as Equation 13 and shown in Figure 2. $H$ and $M$ denote the Hadamard gate and measurement operation, respectively; this notation applies throughout this work.

The proof is provided in Appendix C.1.1. As discussed in Section 3.1, the required quantum resource complexity can vary across different quantum computing platforms. The resource analysis with double-controlled rotation gates as native gates corresponds to neutral-atom quantum platforms, whereas the one considering only single-qubit and CNOT gates as native gates corresponds to superconducting platforms.

### 3.2.2 Multivariate case

We consider the implementation of multivariate polynomials. Given an integer $L$ and parameter $\boldsymbol{\theta} \in \mathbb{R}^{L+1}$, with input $x \in [-1, 1]$, define the parameterized unitary

$$U^p(\boldsymbol{x}) = \bigotimes_{i=1}^{D} \left( R_z\left(\theta_L^i\right) \prod_{j=0}^{L-1} R_x(-2\arccos(x)) R_z\left(\theta_j^i\right) \right), \tag{14}$$

that is shown in Figure 4.

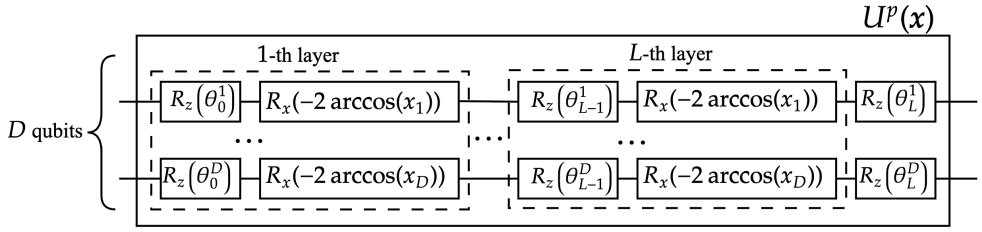


**Fig. 4** The circuit $U^p(\boldsymbol{x})$ retains exactly the same structure as $U_{\boldsymbol{\theta}}(x)$ in Figure 2, except that it acts on multiple qubits via tensor products.

We can then propose the quantum resource complexity analysis for implementing any real multivariate polynomial:

**Theorem 2** (Quantum circuit for multivariate polynomial) *Let* $p(\boldsymbol{x}) = \sum_{\boldsymbol{n}} c_{\boldsymbol{n}} \boldsymbol{x}^{\boldsymbol{n}}$ *be any real multivariate polynomial with* $D$ *variables and degree at most* $L$ *in each variable such that* $\boldsymbol{x} \in [-1,1]^D$, $c_{\boldsymbol{n}} \in \mathbb{R}$, $\boldsymbol{n} \in [L]^D$. *Then there exists a quantum model* $\mathcal{Q}$ *that consists of a PQC* $W_p(\boldsymbol{x})$ *and an observable* $Z^{(0)}$ *with the scaling factor* $\Lambda$ *such that*

$$f_{\mathcal{Q}}(\boldsymbol{x}) := \langle 0|W_p^\dagger(\boldsymbol{x}) Z^{(0)} W_p(\boldsymbol{x})|0\rangle = p(\boldsymbol{x})/\Lambda,$$

*where* $\Lambda = |c_{\boldsymbol{n}}|_\infty (L+1)^D$ *and* $Z^{(0)}$ *is the Pauli* $Z$ *observable on the first qubit. The width of the PQC is* $O(D \log L)$, *the depth is* $O\left(D^2 L^{D+1} \log L\right)$, *and the number of parameters is* $O\left(DL^{D+1}\right)$.

The circuit diagram of $W_p(\boldsymbol{x})$ is shown in Figure 5.

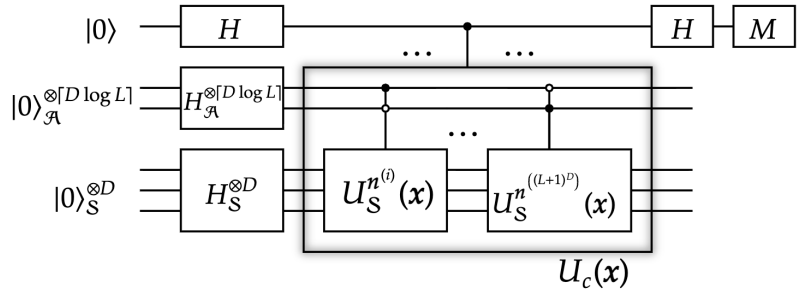


**Fig. 5** The circuit of $W_p(\boldsymbol{x})$ consists of a single-qubit system, a $\lceil D \log L \rceil$-qubits system $\mathcal{A}$, and a $D$-qubits system $\mathcal{S}$. This quantum model can consider as the Hadamard test to estimate $p(\boldsymbol{x}) = \langle +|_{\mathcal{A}}^{\otimes\lceil D\log L\rceil} \langle +|_{\mathcal{S}}^{\otimes D} U_c(\boldsymbol{x}) |+\rangle_{\mathcal{A}}^{\otimes\lceil D\log L\rceil} |+\rangle_{\mathcal{S}}^{\otimes D}$ such that $U_c(\boldsymbol{x}) = \sum_{i=1}^{(L+1)^D} |i\rangle_{\mathcal{A}} \langle i| \otimes U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x})$ where $U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x})$ is defined in Equation 14 and shown in Figure 4.

The proof is provided in Appendix C.1.2. Theorem 2 is closely related to Yu et al. (2024, Theorem 1), as both establish circuit complexity bounds for real multivariate

polynomials. However, the exact complexities differ, and we additionally provide an explicit bound on the scaling factor for general real multivariate polynomials. Despite this, both their circuit depth and number of parameters grow exponentially, making them prohibitively expensive in practice. In the next part, we propose a new theorem within the tensor decomposition framework, showing that under suitable constraints, some circuit resources (width, depth, number of parameters) exhibit only polynomial complexity, rendering the approach far more practical for real-world problems.

## 3.3 Tensor-decomposed polynomial implementation

In this subsection, we present the quantum models designed to implement tensor-decomposed polynomials and analyze their corresponding quantum resource complexity.

**Theorem 3** (Quantum circuit for tensor-decomposed polynomial) *Let $p(\boldsymbol{x})$ be any real multivariate polynomial with $D$ variables and degree at most $L$ in each variable such that*

$$p(\boldsymbol{x}) = \sum_{i=1}^{R} \lambda_i \prod_{j=1}^{D} p_{i,j}\left(x_j\right),$$

*where $R \in [1, (L+1)^D]$, $\sum_{i=1}^{R} |\lambda_i| = \Lambda \in \mathbb{R}$, and each $p_{i,j}(x_j)$ is a univariate polynomial of degree $L$ satisfying $|p_{i,j}(x_j)| \leq 1/2$ for $x_j \in [-1, 1], \forall i \in [R], j \in [D]$. Then, there exists a quantum model $\mathcal{Q}$ that consists of a PQC $W_p(\boldsymbol{x})$ and an observable $Z^{(0)}$ such that*

$$f_{\mathcal{Q}}(\boldsymbol{x}) := \langle 0|W_p^{\dagger}(\boldsymbol{x}) Z^{(0)} W_p(\boldsymbol{x})|0\rangle = p(\boldsymbol{x})/\Lambda,$$

*where $Z^{(0)}$ is the Pauli $Z$ observable on the first qubit. The width of the PQC is at most $2D + \lceil \log R \rceil + 1$, the depth is $O(RDL \log R)$, and the number of parameters is $O(RDL)$.*

The circuit diagram of the $W_p(\boldsymbol{x})$ is shown in Figure 6 and Figure 7.

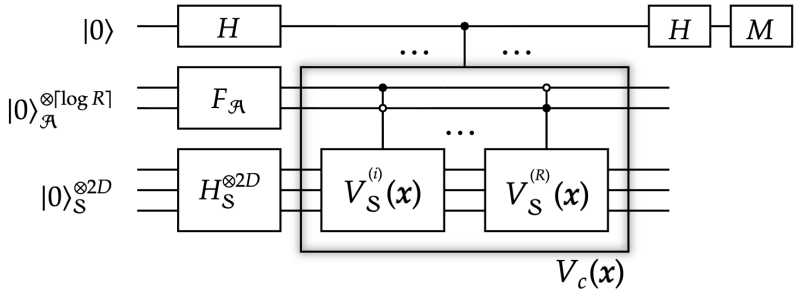


**Fig. 6** The circuit of $W_p(\boldsymbol{x})$ consists of a single-qubit system, a $\lceil \log R \rceil$-qubits system $\mathcal{A}$, and a $2D$-qubits system $\mathcal{S}$. We have $F_{\mathcal{A}} |0\rangle_{\mathcal{A}}^{\otimes \lceil \log R \rceil} = \sum_{i=1}^{R} \sqrt{\frac{|\lambda_i|}{\Lambda}} |i\rangle_{\mathcal{A}}$, and this quantum model can be considered as the Hadamard test circuit to estimate $p(\boldsymbol{x})/\Lambda = \langle v_p| V_c(\boldsymbol{x}) |v_p\rangle$ such that $|v_p\rangle = \sum_{i=1}^{R} \sqrt{\frac{|\lambda_i|}{\Lambda}} |i\rangle_{\mathcal{A}} \otimes |+\rangle_{\mathcal{S}}^{\otimes 2D}$, and $V_c(\boldsymbol{x}) = \sum_{i=1}^{R} |i\rangle_{\mathcal{A}} \langle i| \otimes V_{\mathcal{S}}^{(i)}(\boldsymbol{x})$ where $V_{\mathcal{S}}^{(i)}(\boldsymbol{x})$ is shown in Figure 7.

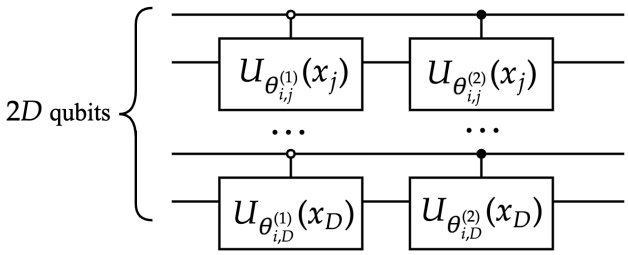


**Fig. 7** Circuit of $V_S^{(i)}(\boldsymbol{x}) = \bigotimes_{j=1}^{D}\left(|0\rangle\langle 0|\otimes U_{\boldsymbol{\theta}_{i,j}^{(1)}}(x_j)+|1\rangle\langle 1|\otimes U_{\boldsymbol{\theta}_{i,j}^{(2)}}(x_j)\right)$ where $U_{\boldsymbol{\theta}_{i,j}^{(1)}}(x_j), U_{\boldsymbol{\theta}_{i,j}^{(2)}}(x_j)$ are defined in Equation 13 and shown in Figure 2.

The proof is provided in Appendix C.2, and we refer to this quantum model as the *tensor-decomposed model*. We can observe that the circuit depth and the number of parameters grow polynomially rather than exponentially when the tensor rank $R$ is not very large. Here, we specifically consider a simple case of Theorem 3 when the tensor rank $R = 1$:

**Corollary 4** (Quantum circuit for rank-1 tensor-decomposed polynomial) Let $p(\boldsymbol{x}) = \prod_{j=1}^{D} p_j\left(x_j\right)$ be any real multivariate polynomial such that $\forall j \in [D], x_j \in [-1, 1]$, each $p_{i,j}(x_j)$ is a univariate polynomial of degree $L$ satisfying $|p_{i,j}(x_j)| \leq 1/2$. Then there exists a quantum model $\mathcal{Q}$ that consists of a PQC $W_p(\boldsymbol{x})$ and an observable $Z^{(0)}$ such that

$$f_{\mathcal{Q}}(\boldsymbol{x}) := \langle 0|W_p^{\dagger}(\boldsymbol{x})Z^{(0)}W_p(\boldsymbol{x})|0\rangle = p(\boldsymbol{x}),$$

where $Z^{(0)}$ is the Pauli $Z$ observable on the first qubit. The width of the PQC is at most $2D + 1$, the depth is at most $4LD + 2$ by allowing double-controlled rotation gates to be implemented natively, and the number of parameters is at most $(2L + 1)D$.

The proof is provided in Appendix C.2.1. Actually, the scaling bound for the univariate polynomial $|p_{i,j}(x_j)| \leq 1/2$ in Theorem 3 and Corollary 4 is a sufficient but not a necessary condition for the existence of the quantum model $\mathcal{Q}$. The scaling bound for both results is derived from Corollary 6 in Appendix, where this condition was originally introduced. In fact, the necessary condition is given in Equation C18.

We apply the model based on Corollary 4 to solve the Merton portfolio optimization problem, see Section 4 for details.

## 3.4 Integrating the tensor-decomposed model into QPINNs

The tensor-decomposed model in Theorem 3 explicitly implements tensor-decomposed polynomials, which often appear in PDE solutions. In this subsection, we first introduce a QPINN based on this tensor-decomposed model with an additional entangling layer. It is also useful to consider the QPINN without an entangling layer, which we refer to as the quantum-inspired PINN, meaning that it can be efficiently simulated classically. The general QPINN framework is defined in Section 2.2.

### 3.4.1 QPINN: expanding the hypothesis space with entanglement

If we directly adapt the tensor-decomposed model within the QPINN framework, the corresponding hypothesis space contains tensor-decomposed polynomials with tensor rank $R$. However, since the quantum resource cost in Theorem 3 grows linearly with $R$, only relatively small values of $R$ are feasible on current quantum hardware, whereas many PDE solutions require substantially higher tensor rank $R$ to achieve acceptable approximation accuracy. Consequently, restricting the model to a fixed tensor rank $R$ limits its expressivity and may lead to unacceptable approximation errors in complex cases.

To obtain a QPINN with better expressivity, we expand each block $V_c(\boldsymbol{x})$ defined in the circuit diagram of Theorem 3, by attaching an entangling unitary

$$V_c(\boldsymbol{x}) \longrightarrow V_c(\boldsymbol{x})\, e^{-iH(\boldsymbol{\lambda})}, \tag{15}$$

where $H(\boldsymbol{\lambda})$ is a $2D$-qubit Hamiltonian with parameters $\boldsymbol{\lambda}$ satisfying

$$e^{-iH(\boldsymbol{\lambda}_0)} = I \tag{16}$$

for a particular parameter choice $\boldsymbol{\lambda}_0$, where $D$ is the number of variables and $I$ is the identity matrix. This guarantees that the original tensor-decomposed hypothesis space is entirely contained within the enlarged hypothesis space with the $e^{-iH(\boldsymbol{\lambda})}$, which means the enlarged hypothesis space is larger than the original one. Because of the circuit architecture shown in Figure 6, the entangling unitary $e^{-iH(\boldsymbol{\lambda})}$ is implemented in the overall circuit as a controlled $e^{-iH(\boldsymbol{\lambda})}$ operation with a single control qubit (typically the first qubit), in the same sense as the control structure used in CNOT gates. We refer to this controlled $e^{-iH(\boldsymbol{\lambda})}$ as the *added entangling layer* in the QPINN. Figure 8 shows its general implementation.

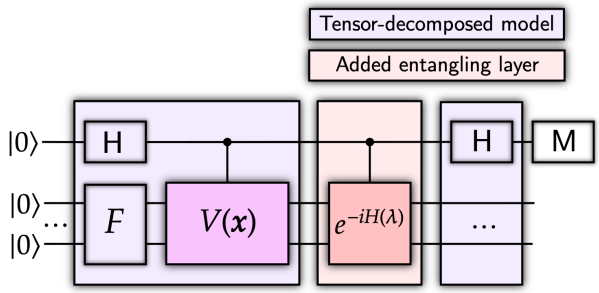


**Fig. 8** When the added entangling layer is the identity, This is equivalent to the Hadamard test and recovers the tensor-decomposed model in Theorem 3. $\langle v|\, V(\boldsymbol{x})\, |v\rangle = p(\boldsymbol{x})/\Lambda$ where $F|0\rangle = |v\rangle$.

The added entangling layer can introduce non-separable multi-qubit correlations, expanding the hypothesis space beyond tensor-decomposed structures and making the overall model not classically simulable. This enlarged hypothesis space provides significantly greater expressivity and allows the QPINN to approximate more complex

PDE solutions. However, increased expressivity may come at the cost of reduced trainability, particularly through the barren plateau (BP) phenomenon, which refers to the exponential vanishing of gradient variance during training (McClean et al. 2018). Appropriate designs of the Hamiltonian $H$ may help balance this trade-off and avoid the BP phenomenon, further exploration of such designs is an interesting direction for future work.

### 3.4.2 Quantum-inspired PINN: a dequantized special case

Although quantum models are usually intended to run on quantum computers, recent studies (Cotler et al. 2021; Cerezo et al. 2025) have shown that many quantum machine learning algorithms can in fact be efficiently simulated on classical hardware, a phenomenon known as *dequantization* (Chia et al. 2020).

When the Hamiltonian $H(\boldsymbol{\lambda})$ defined in Equation (15) satisfies Equation (16) for all parameter choices $\boldsymbol{\lambda}$ (rather than only for a particular parameter choice), the corresponding QPINN model reduces to the tensor-decomposed model proposed in Theorem 3 (without the added entangling layer). Its hypothesis function is a tensor-decomposed product of one-dimensional components, each of which can be obtained from a single-qubit circuit shown in Equation (13) whose output can be written explicitly as a univariate function. Since single-qubit circuits are classically simulable with negligible cost, this tensor-decomposed model can be evaluated efficiently by computing these univariate functions and their tensor-decomposed product directly on a classical computer, without executing the underlying quantum circuit. In this case, it loses its expressivity advantage but becomes classically simulable, and should therefore be considered as a quantum-inspired PINN rather than a pure QPINN.

We emphasize that classical simulability is not necessarily a negative feature. On the contrary, it enables practical deployment on today's hardware. If the quantum-inspired PINN shows clear theoretical or experimental advantages, it may be more practically useful than QPINNs relying on current quantum hardware.

## 4 Experiments

In this section, we compare our QPINN and Quantum-inspired PINN presented in Section 3.4 with two classical PINNs for solving the Merton Portfolio Optimization HJB PDE defined in Equation 2, with the parameters $r = 0.02, T = 1.0, \gamma = 0.95, \mu = 0.0219, \sigma = 0.2$. These parameters define the market environment and investor preferences for solving the HJB PDE (see Section 2.1 for further details).

### 4.1 Configuration

**QPINN and Quantum-inspired PINN.** The QPINN is based on the tensor-decomposed model in Corollary 4 (Theorem 3 when tensor rank $R = 1$) with an added entangling layer. We take the controlled-$e^{-iH(\lambda)}$ as the added entangling layer with the 4-qubit Hamiltonian $H(\lambda) = \frac{\lambda}{2}(I \otimes Z \otimes Z \otimes I)$ where $\lambda \in \mathbb{R}$ is a trainable parameter, as defined in Equations (15) and (16).

The circuit diagram of our QPINN is shown in Figure 9. The tensor-decomposed model follows Corollary 4 with polynomial degree $L = 1$ and number of variables $D = 2$. The added entangling layer controlled-$e^{-iH(\lambda)}$ is implemented by an `IsingZZ` gate on qubits 3 and 4, with identity matrices on qubits 2 and 5, under the control of qubit 1. The QPINN therefore has 6 parameters from the tensor-decomposed model and 1 parameter from the added entangling layer, which means there are 7 trainable parameters in total. In this experiment, we focus on a relatively simple Hamiltonian, while the extension to more general and expressive entangling Hamiltonians is left for future work.

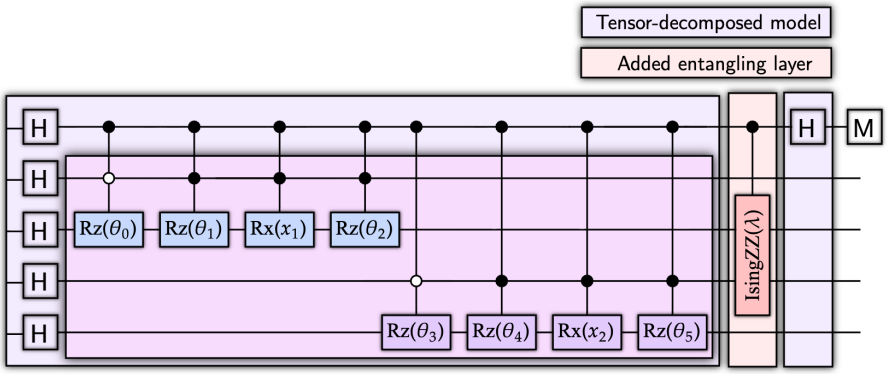


**Fig. 9** Circuit of QPINN for solving HJB PDE in the Merton portfolio optimization. It combines a tensor-decomposed model with an added entangling layer. The `IsingZZ` gate denotes the two-qubit operation $e^{-i\frac{\theta}{2}Z\otimes Z}$, and `Rz`, `Rx` denote single-qubit rotation gates about the Pauli-Z, Pauli-X axes, respectively.

When the parameter $\lambda = 0$, the added entangling layer reduces to an identity matrix. Since the polynomial degree $L = 1$ and the number of variables $D = 2$, the corresponding hypothesis function can be a product of two degree 1 univariate polynomials $p_1(x)p_2(t)$. The product represents a tensor-decomposed structure with tensor rank $R = 1$. A larger $L$ value improves expressivity, allowing the target function to be approximated more precisely, but also increases the required resource costs. In this case, the hypothesis space contains functions of the form $\{p_1(x)p_2(t)\}$. This special case corresponds to the Quantum-inspired PINN by keeping $\lambda = 0$ at all times, as shown in Figure 9 without the added entangling layer.

When $\lambda \in \mathbb{R}$ in general, we can't express the hypothesis function explicitly, but the hypothesis space also necessarily contains functions of the tensor-decomposed structure $\{p_1(x)p_2(t)\}$. Although the added entangling layer enlarges the hypothesis space of the QPINN beyond tensor-decomposed polynomials, the part of the circuit preceding this layer still follows the tensor-decomposed structure. In fact, the actual analytical solution of this HJB PDE is $v(t,x) = \exp(-k(T-t))\frac{x^\gamma}{\gamma}$, which is also a product of two univariate functions, and is similar to the tensor-decomposed structure $p_1(x)p_2(t)$. Hence, we consider this tensor-decomposed structure contains useful

information for training, referred as an inductive bias. The QPINN retains the similar inductive bias as the Quantum-inspired PINN since the added entangling layer weakens it but does not eliminate it.

We emphasize that real-world PDE problems often have no analytic solution, and here we only use it to demonstrate our models. Our proposed frameworks can also be applied to more complex cases where analytical solutions are unavailable.

**PINNs.** We compare our QPINN and Quantum-inspired PINN with their classical counterpart, the *Counterpart PINN*, which shares the same hypothesis function as the Quantum-inspired PINN and thus has the same inductive bias advantage. The trainable parameters of the Counterpart PINN are the coefficients of two univariate polynomials $p_1$ and $p_2$ with degree 2, giving a total of 6 parameters. We also apply a commonly used fully connected PINN, the *FC PINN*, which consists of 5 hidden layers with 10 neurons each and Tanh activations, and contains 481 trainable parameters in total.

**Hyperparameter for all models.** We use the `torch_optimizer.Lamb` (You et al. 2020) with `weight_decay=0`, `betas=(0.0, 0.0)`. LAMB is a stochastic gradient–based optimizer that adjusts the learning rate of each layer according to the ratio between the norm of the weight and the norm of its gradient. This normalization stabilizes training and ensures a fair comparison between models with largely different parameter counts. The learning rate decreases from $10^{-2}$ to $10^{-3}$ using `torch_optimizer.lr_scheduler.CosineAnnealingLR` during the first 150 epochs, remains at $10^{-3}$ for the next 100 epochs, and is finally set to $2 \times 10^{-4}$ for the remaining 750 epochs, for a total of 1000 epochs. For simplicity, in this simulation we use `torch.autograd` rather than the parameter-shift rule to compute derivatives of the hypothesis functions. The loss function for this HJB PDE is given in Equation 7 and 8 with the loss weights $W_d = W_1 = 1, W_2 = 5$, where each loss term is estimated by uniformly sampling 50 collocation points for $(x, t) \in [0.01, 0.99]$. In addition, we multiply the outputs of all models by a scaling factor of 10. All experiments are performed on the Apple M2 Pro CPU.

## 4.2 Plot comparison

The loss value comparison of QPINN, Quantum-inspired PINN, and two PINNs for solving the Merton Portfolio Optimization HJB PDE defined in Equation 2 are shown in Figure 10.

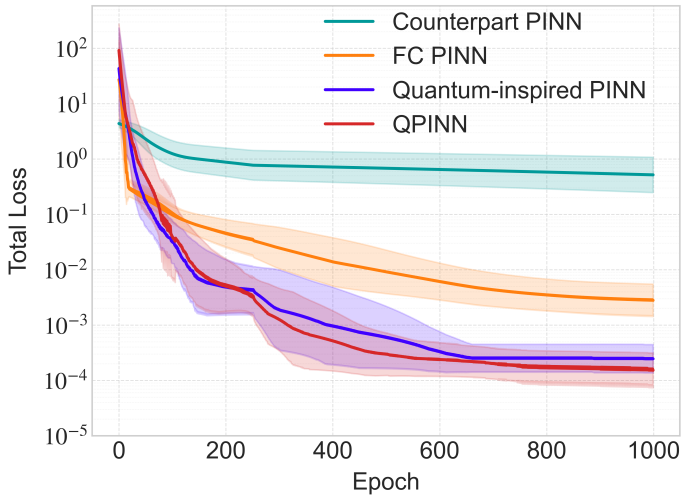


**Fig. 10** Comparison of loss values for solving the HJB PDE in the Merton portfolio optimization, based on 10 independent runs of the QPINN, Quantum-inspired PINN, Counterpart PINN and FC PINN presented in Table **??**, each trained for 1000 epochs. Curves represent the geometric mean loss, and shaded regions denote $\pm 1$ geometric standard deviation.

The QPINN and Quantum-inspired PINN converge both faster and more accurately than the two PINNs. The performance gap between the Quantum-inspired PINN and the Counterpart PINN can reflect the quantum-induced advantage, since both models share the same inductive bias but the Quantum-inspired model achieves better performance. In contrast, the improvement of the QPINN and Quantum-inspired PINN over the FC PINN demonstrates the benefit of the inductive bias, as it leads to better performance even though the FC PINN contains orders of magnitude more parameters. Moreover, although the QPINN and Quantum-inspired PINN converge at comparable rates, the QPINN achieves a lower final loss value. This improvement can be attributed to the larger hypothesis space of the QPINN, which provides greater expressivity.

Figure 10 reflects the trainability disadvantage of classical PINNs. However, the expressivity of the FC PINN and the Counterpart PINN is not as limited as the plot may suggest. Given sufficient training epochs, both models can achieve comparably low loss values, owing to their large number of parameters and the inductive bias, respectively.

To facilitate a more direct comparison, Figure 11 illustrates the 3D surfaces of the analytical solution and the approximations by different models obtained after 1000 training epochs in our 8-th run. The analytical solution represents the maximum expected utility of wealth presented in Equation (3). The QPINN, Quantum-inspired PINN, FC PINN, and Counterpart PINN plots show how each model approximates this analytical benchmark. Visually, the QPINN and Quantum-inspired PINN surface closely match the analytical solution, demonstrating its superior ability to capture the shape and magnitude of the expected utility function compared with classical PINN models.

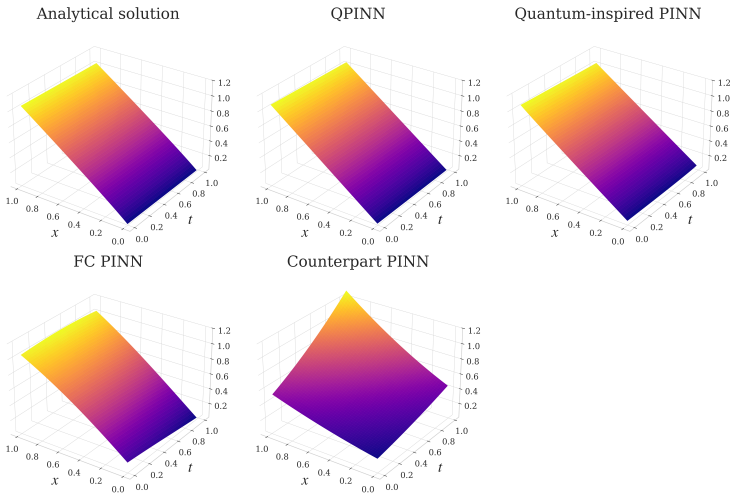


**Fig. 11** 3D comparison of the analytical solution and the approximated solutions of the HJB PDE in the Merton portfolio optimization after 1000 training epochs, based on a single run of the QPINN, Quantum-inspired PINN, Counterpart PINN and FC PINN presented in Table **??**. The analytical solution is $v(t,x) = \exp(-0.019857375(1-t))\frac{x^{0.95}}{0.95}$.

Precisely, we can compare the 2D plot of $v(t = 0.5, x)$ for QPINN, Quantum-inspired PINN, and PINNs, as shown in Figure 12. This cross-sectional view provides a clearer understanding of how each model approximates the analytical solution along the spatial dimension. The function $v(t = 0.5, x)$ represents the expected utility of wealth evaluated at the mid-horizon time $t = 0.5$. As observed, the QPINN approximation closely follows the analytical curve across the entire domain, achieving high accuracy both near the boundaries and within the interior region. In contrast, while the Quantum-inspired PINN performs well in the interior region, its approximation becomes less accurate near the boundaries. In addition, the FC PINN and the Counterpart PINN exhibit even larger deviations from the analytical solution, despite the Counterpart PINN requiring 481 trainable parameters compared with only 6 in the QPINN.

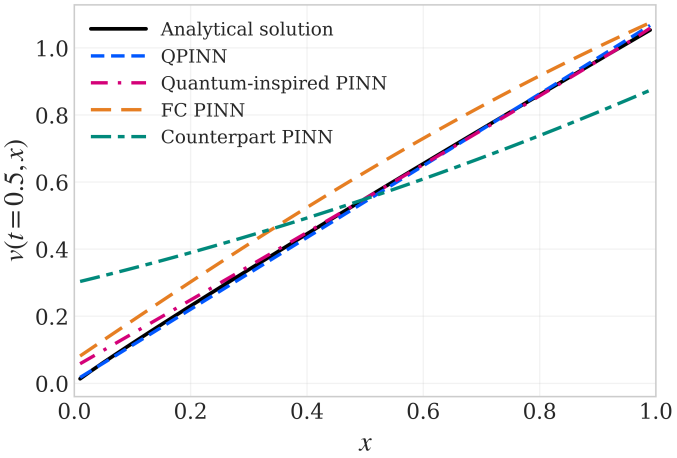


**Fig. 12** 2D comparison of the analytical solution and the approximated solutions of the HJB PDE in the Merton portfolio optimization after 1000 training epochs, based on a single run of the QPINN, Quantum-inspired PINN, Counterpart PINN and FC PINN with time $t = 0.5$.

The experimental observations presented above suggest that the optimization landscape of the quantum-induced models may differ significantly from that of large-scale classical neural networks. Even though the QPINN and the Quantum-inspired PINN use only a few parameters, these models are capable of encoding complex mappings within the Hilbert space, which enables more efficient exploration of the optimization landscape. This partly explains a faster convergence compared with their PINN counterpart, as observed in Figure 10. Importantly, the superior accuracy of the QPINN and the Quantum-inspired PINN compared to the FC PINN demonstrates that the inductive bias of the model architecture, rather than simply the number of parameters, plays a crucial role in learning solutions to structured PDEs. This highlights the potential of designing quantum or quantum-inspired models whose hypothesis functions reflect properties of the target PDEs.

## 4.3 Financial interpretation

We solve the HJB PDE defined in Equation 2 with the parameters $r = 0.02, T = 1.0, \gamma = 0.95, \mu = 0.0219, \sigma = 0.2$. Here, $r$ denotes the risk-free interest rate, $T$ is the investment horizon, and $\mu$ and $\sigma$ represent the expected return and volatility of the risky asset, respectively. The parameter $\gamma$ characterizes the investor's attitude toward risk. In this setting, the analytical solution to the HJB PDE is $v(t,x) = \exp(-0.019857375(1-t))\frac{x^{0.95}}{0.95}$, which represents the maximum expected utility of wealth. The optimal investment fraction in the risky asset is constant and equal to $\hat{\alpha} = 0.95$, meaning that 95% of the portfolio should be allocated to the risky asset and the remaining 5% to the risk-free asset.

The QPINN and PINN models can be used to solve the HJB PDE to approximate its analytical solution $v(t,x)$. Once the hypothesis functions are obtained from the

approximation shown in Figure 11 , the optimal control $\hat{\alpha}(t)$ can be directly derived from the HJB PDE (Equation (B13) and (B14)), this would allow an investor to achieve his maximum expected wealth.

# 5 Conclusion

In this work, we have introduced a family of quantum models capable of implementing different forms of polynomials, including their univariate, multivariate, and tensor-decomposed variants for solving PDEs in the framework of the PINN model. It is noted that our results reduce the required quantum circuit resources from exponential to polynomial complexity for implementing tensor-decomposed polynomials when the tensor rank is not too large. Hence, it can be seen as an extension to the results presented in Yu et al. (2024), although the underlying assumptions and complexity are different (see Table 1). From a broader perspective, our results highlight that properly designed tensor-decomposed models can naturally capture the separable structures of many PDEs, such as the Heat equation, the Laplace equations, the time-dependent Schrödinger equation, and the PDEs in portfolio optimization. We further find that such quantum tensor-decomposed models can be simulated efficiently on classical hardware, which implies that quantum-induced advantages can be realized within PINN architectures implemented entirely on classical computers.

In contrast, the addition of an entangling layer yields a QPINN that is no longer efficiently simulable on classical hardware. While many quantum machine learning approaches offer limited guarantees regarding the existence of suitable solutions within their hypothesis spaces, our QPINN retains the potential expressivity benefits of quantum entanglement while ensuring that the hypothesis space contains at least one suitable solution. Although our work does not provide an explicit error bound for this approximation, such bounds can be inferred from classical results on polynomial approximations of (continuous) target functions. In our experiments, we use a relatively simple Hamiltonian, but we believe that choosing a Hamiltonian that produces stronger entanglement can further improve the expressivity and performance of QPINN.

Our study primarily addresses expressivity, leaving the equally crucial question of theoretical trainability open for future work. Since trainability is widely believed to be an important source of potential quantum neural network advantages (Schreiber et al. 2023), future work should integrate expressivity analysis with a systematic investigation of optimization landscapes and gradient behavior. Moreover, a rigorous quantification of approximation errors between computational model hypothesis functions and PDE solutions should be essential to establish stronger theoretical guaranties.

Overall, our work develops both a quantum resource analysis for implementing tensor-decomposed polynomials and a hypothesis space characterization of the QPINN and the Quantum-inspired PINN. For both the QPINN and the Quantum-inspired PINN, the hypothesis space contains the entire family of tensor-decomposed polynomials, which guarantees the existence of an approximation to the PDE solution. In the case of

the QPINN, the added entangling layer further expands the hypothesis space beyond tensor-decomposed polynomials, providing additional expressivity that the Quantum-inspired PINN does not possess. Moreover, because these architectures are built upon an efficient tensor-decomposed structure, they offer a scalable pathway toward implementing PDE solvers on near-term quantum hardware. Together, our results offer both theoretical insight and preliminary experimental indications toward developing practical quantum models for solving structured PDEs in finance.

**Author Contributions.** L.W. conceived the research idea, developed the theoretical framework, implemented the models, performed the experiments, and wrote the manuscript. A.L., S.S., and Z.T. supervised the research, provided guidance throughout the project, and contributed to the discussion, revision, and improvement of the manuscript. All authors reviewed and approved the final version of the paper.

**Funding.** This research was supported by French government under the France 2030 program, reference ANR-11-IDEX-0003 within the OI H-Code.

**Data Availability.** The source code implementation is available at [`https://github.com/Letao-WANG/QPINNs-for-Portfolio/tree/main`].

## Declarations

**Conflicts of interest** The authors declare no competing interests.

## Appendix A Tensor-decomposed polynomial derivation

We first recall the notion of *tensor rank decomposition* (Kolda and Bader 2009). Given index $n_m \in I_m, \forall m \in [D]$ with $D$ as an integer, let $\mathcal{A} \in \mathbb{F}^{I_1 \times I_2 \times \cdots \times I_D}$ be an $D$-order tensor over a field $\mathbb{F}$ with entries $\mathcal{A}_{n_1,\ldots,n_D}$. A tensor rank decomposition is a representation

$$\mathcal{A} = \sum_{r=1}^{R} \lambda_r \left(\mathbf{a}_r^{(1)} \otimes \mathbf{a}_r^{(2)} \otimes \cdots \otimes \mathbf{a}_r^{(D)}\right), \tag{A1}$$

where coefficient $\lambda_r \in \mathbb{F}$, tensor $\mathbf{a}_r^{(m)} \in \mathbb{F}^{I_m}$, $R$ denoted as the decomposition rank and "$\otimes$" denotes the outer product. For any $D'$-order tensor $\mathbf{a} \in \mathbb{F}^{I_1 \cdots \times I_{D'}}$, we can consider $\mathbf{a}_{n_1,\ldots,n_{D'}} \in \mathbb{F}$ as the $(n_1, \ldots, n_{D'})$ entry of $\mathbf{a}$. By using the entry equation

$$\left(\mathbf{a}_r^{(1)} \otimes \mathbf{a}_r^{(2)} \otimes \cdots \otimes \mathbf{a}_r^{(D)}\right)_{n_1,\ldots,n_D} = \mathbf{a}_{r,n_1}^{(1)} \mathbf{a}_{r,n_2}^{(2)} \cdots \mathbf{a}_{r,n_D}^{(D)} \equiv \mathbf{a}_{r,n_1} \mathbf{a}_{r,n_2} \cdots \mathbf{a}_{r,n_D},$$

we can obtain the entry of tensor $\mathcal{A}$:

$$\mathcal{A}_{n_1,\ldots,n_D} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_{r,n_1} \mathbf{a}_{r,n_2} \cdots \mathbf{a}_{r,n_D}, \tag{A2}$$

where $\mathcal{A}_{n_1,\ldots,n_D} \in \mathbb{F}, \lambda_r \in \mathbb{F}$ and $\mathbf{a}_{r,n_j} \in \mathbb{F}, \forall j \in [D]$. It is worth noting that there is a variant of the tensor rank decomposition, known as the *CP decomposition* (Hitchcock 1927). The key difference is that tensor rank decomposition provides an exact decomposition, whereas CP decomposition yields an approximate one, with the rank specified by the user. From Equation (A2), the multivariate polynomial coefficients $c_{\boldsymbol{n}}$ in Definition 2 can be expressed as

$$c_{\boldsymbol{n}} = c_{n_1,\ldots,n_D} = \sum_{r=1}^{R} \lambda_r c_{r,n_1} \, c_{r,n_2} \, \cdots \, c_{r,n_D}. \tag{A3}$$

Then we define another version of a univariate polynomial with degree $L$ such that

$$p_{r,j}\left(x_j\right) = \sum_{n_j} c_{r,n_j} x_j^{n_j}, \tag{A4}$$

where $c_{r,n_j} \in \mathbb{R}, n_j \in [L], \forall r \in [R], j \in [D]$. Combining Definition 2 for real multivariate polynomials with Equation (A3) (A4) and assuming$\lambda_r \in \mathbb{R}, \forall r \in [R]$, we can have

$$\begin{aligned}\sum_{r=1}^{R} \lambda_r \prod_{j=1}^{D} p_{r,j}(x_j) &= \sum_{r=1}^{R} \lambda_r \prod_{j=1}^{D} \sum_{n_j} c_{r,n_j} x_j^{n_j} = \sum_{r=1}^{R} \lambda_r \sum_{n_1} \sum_{n_2} \cdots \sum_{n_D} \prod_{j=1}^{D} (c_{r,n_j} x_j^{n_j}) \\ &= \sum_{n_1} \sum_{n_2} \cdots \sum_{n_D} \left( \sum_{r=1}^{R} \lambda_r c_{r,n_1} c_{r,n_2} \cdots c_{r,n_D} \right) x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D} = \sum_{\boldsymbol{n}} c_{\boldsymbol{n}} x^{\boldsymbol{n}}.\end{aligned}$$

It should be noted that for any real multivariate polynomial $p(\boldsymbol{x})$, there always exists an integer $R$ such that its coefficients admit Equation (A3). In the worst case, one can represent each monomial of $p(\boldsymbol{x})$ by a univariate polynomials $p_{r,j}(x_j)$, so that $R$ is bounded above by the number of possible multi-indices $\boldsymbol{n} \in [L]^D$. Hence, $1 \leq R \leq (L+1)^D$.

We then give the formal definition: let $p(\boldsymbol{x})$ be a real multivariate polynomial with $D$ variables and degree at most $L$ in each variable in Definition 2. We can always find its coefficients $(c_{\boldsymbol{n}})$ admit a decomposition of Equation (A3) with rank $R$ , then $p(\boldsymbol{x})$ can be written as

$$p(\boldsymbol{x}) = \sum_{r=1}^{R} \lambda_r \prod_{j=1}^{D} p_{r,j}(x_j),$$

where each $p_{r,j}(x_j)$ is the univariate polynomial, $\lambda_r \in \mathbb{R}, c_{r,n_j} \in \mathbb{R}, n_j \in [L], \forall r \in [R], j \in [D]$. We call such $p(\boldsymbol{x})$ a tensor-decomposed polynomial (TD polynomial).

# Appendix B Merton portfolio optimization derivation

In this work we consider only the case without a jump component. In particular, let $S_0 = (S_0(t))_{t\in[0,T]}$ and $S_1 = (S_1(t))_{t\in[0,T]}$ denote the amount of money the investor has in the risk-free asset and risky asset, respectively. The processes evolve according to

$$\begin{cases} dS_0(t) = rS_0(t)dt \\ dS_1(t) = S_1(t)(\mu(t)dt + \sigma(t)dB(t)) \\ S_0(0) = 1, S_1(0) = s \end{cases} \tag{B5}$$

where $B(t)$ is a standard Brownian motion, $\mu(t)$ is the expected return of the asset at time $t$, and $\sigma(t)$ is the volatility of the asset at time $t$. We assume that $\mu(t)$ and $\sigma(t)$ are both stochastic processes.

Let $\pi = (\pi_0(t), \pi_1(t))_{t\in[0,T]}$ denote an investor portfolio where $\pi_0(t)$ and $\pi_1(t)$ represent the proportion of wealth invested in the risk-free asset and in the risky asset, respectively. Therefore, for every $t \in [0, T]$, they satisfy the relation $\pi_0(t) + \pi_1(t) = 1$. The wealth at time $t$ of such a portfolio is

$$X(t) = \pi_0(t)S_0(t) + \pi_1(t)S_1(t), \tag{B6}$$

and represents the total amount of money invested in the market. The portfolio is assumed to be self-financing, that is,

$$dX(t) = \pi_0(t)dS_0(t) + \pi_1(t)dS_1(t). \tag{B7}$$

Combining Equation (B5) (B6) (B7), the investor's wealth process is given by

$$dX(t) = X(t)rdt + \pi_1(t)S_1(t)(\mu(t) - r)dt + \pi_1(t)S_1(t)\sigma(t)dB(t). \tag{B8}$$

We define $X^{t,x,\alpha}(s)$ as the expected wealth according to Equation (B8) starting from $X(s = t) = x$ and evolving until $s = T$, we also have

$$dX^{t,x,\alpha}(s) = X^{t,x,\alpha}(s)rds + \pi_1(s)S_1(s)(\mu(s) - r)ds + \pi_1(s)S_1(s)\sigma(s)dB(s). \tag{B9}$$

Let $\alpha(t)$ denote the fraction of the total wealth invested in stocks at time $t$ such that

$$\alpha(t) = \frac{\pi_1(t)S_1(t)}{X(t)} \iff \pi_1(t)S_1(t) = \alpha(t)X(t). \tag{B10}$$

Using Equation (B10) (B9), one obtains

$$dX^{t,x,\alpha}(s) = X^{t,x,\alpha}(s)[(\alpha(s)(\mu(s) - r) + r)ds + \alpha(s)\sigma(s)dB(s)]. \tag{B11}$$

The investor's objective is to maximize the expected utility over $[0, T]$, which is defined as

$$\mathcal{J}(t, x, \alpha) = \mathbb{E}\left[U\left(X^{t,x,\alpha}(T)\right)\right],$$

where $U$ is the investor's utility function. We introduce the following assumptions on the utility function:

- $U(x)$ is a continuous, non-decreasing, and concave function on $[0, \infty)$ with $U(0) = 0$.
- There exists $\gamma > 0$ and a constant $K > 0$ such that $U(x) \leq K(1 + x)^\gamma$ for all $x \in [0, \infty)$.

The Merton portfolio optimization problem is to find a function $v$ and an optimal control $\hat{\alpha}$ such that

$$v(t, x) = \sup_{\alpha \in \mathcal{A}} \mathcal{J}(t, x, \alpha) = \mathcal{J}(t, x, \hat{\alpha}), \quad (t, x) \in [0, T] \times [0, +\infty), \tag{B12}$$

which is an optimal control problem. In fact, the function $v$ also solve its corresponding Hamilton-Jacobi-Bellman (HJB) equation:

$$\begin{cases} \frac{\partial v}{\partial t}(t, x) + H\left(t, x, \frac{\partial v}{\partial x}(t, x), \frac{\partial^2 v}{\partial x^2}(t, x)\right) = 0, & (t, x) \in [0, T) \times [0, +\infty) \\ v(T, x) = U(x), & x \in [0, +\infty) \end{cases} \tag{B13}$$

where, for $H(t, x, p, q) \in [0, T] \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}$,

$$H(t, x, p, q) = \sup_{\alpha \in A} \left\{ pxr + px(\mu - r)\alpha + \frac{1}{2} x^2 \sigma^2 q \alpha^2 \right\},$$

is the Hamiltonian of the problem. Let $\psi(\alpha) := pxr + px(\mu - r)\alpha + \frac{1}{2} x^2 \sigma^2 q \alpha^2$. Assume $\mu(t)$ and $\sigma(t)$ as constant, $\mu > r$ and $U(x) = \frac{x^\gamma}{\gamma}$ with risk $\gamma$, we have

$$\psi'(\alpha) = px(\mu - r) + q\sigma^2 x^2 \alpha.$$

Let $\hat{\alpha}$ such that $\psi'(\hat{\alpha}) = 0$, which means

$$\hat{\alpha} = -\frac{\mu - r}{\sigma^2} \frac{p}{qx}. \tag{B14}$$

When the investor is risk-averse, i.e., $\gamma \in (0, 1)$, the point $\hat{\alpha}$ corresponds to the maximum of $\psi(\alpha)$, so we obtain

$$H(t, x, p, q) = prx - \frac{1}{2} \frac{p^2}{q} \left( \frac{\mu - r}{\sigma} \right)^2. \tag{B15}$$

The derivation of Equation (B15) is presented in (Di Persio and Fraccarolo 2025). Hence, we have the corresponding HJB PDE from Equation (B13) and (B15) on the

domain $(t, x) \in [0, T] \times [0, +\infty)$:

$$\begin{cases} \partial_t v(t,x) \partial_x^2 v(t,x) + \partial_x v(t,x) \partial_x^2 v(t,x) rx - \frac{1}{2} \left(\frac{\mu - r}{\sigma}\right)^2 \partial_x v(t,x)^2 = 0, \\ v(T,x) = \frac{x^\gamma}{\gamma}, \\ v(t,1) = \exp(-k(T-t)) \frac{1}{\gamma}, \end{cases} \tag{B16}$$

with the analytical solution $v(t,x) = \exp(-k(T-t)) \frac{x^\gamma}{\gamma}$ where $k = \frac{1}{2} \frac{\gamma}{\gamma - 1} \left(\frac{\mu - r}{\sigma}\right)^2 - r\gamma$, which can be interpreted as the maximum expected utility of wealth. The derivation of the analytical solution and the optimal control $\hat{\alpha} = \frac{1}{1-\gamma} \frac{\mu - r}{\sigma^2}$ is also shown in (Di Persio and Fraccarolo 2025). The solution $v(t,x)$ of the HJB PDE also solves Equation (B12). We can then apply QPINNs to solve the HJB PDE, then calculate the optimal control based on Equation (B14), and thus obtain the Merton portfolio problem solution.

Note that the simplifying assumptions made here are just for illustration; in general, solving the associated HJB PDE is a standard approach for addressing optimal control problems, which can then be solved numerically using QPINNs. These two boundary conditions $v(T,x) = \frac{x^\gamma}{\gamma}, v(t,1) = \exp(-k(T-t)) \frac{1}{\gamma}$ in Equation (B16) are not unique, and other admissible choices can equally be considered, for example $v(t,0) = 0, t \in [T]$.

# Appendix C Implementing polynomials derivation

## C.1 Polynomial implementation

In this subsection, we present existing results from the QSP framework and explain how they are used to implement polynomial functions within quantum models, along with an analysis of the corresponding quantum resource complexity.

### C.1.1 Univariate case

We first introduce the quantum model for implementing real univariate polynomials, based on the lemmas of QSP (Gilyén et al. 2019). Define the encoding operator $S(x)$ such that

$$S(x) := R_x(-2 \arccos(x)) = \begin{pmatrix} x & i\sqrt{1-x^2} \\ i\sqrt{1-x^2} & x \end{pmatrix},$$

where $x \in [-1, 1]$ and

$$U_{\boldsymbol{\theta}}(x) := R_z(\theta_L) \prod_{j=0}^{L-1} S(x) R_z(\theta_j). \tag{C17}$$

**Lemma 5** (Gilyén et al. 2019) There exists $\boldsymbol{\theta} \in \mathbb{R}^{L+1}$ such that

$$p(x) = \langle + | U_{\boldsymbol{\theta}}(x) | + \rangle$$

if and only if the real polynomial $p(x) \in \mathbb{R}[x]$ satisfies

1. $\deg(p(x)) \leq L$,

2. $p(x)$ has parity $L \bmod 2$ (i.e. if $L$ is even/odd then $p(x)$ must be an even/odd function),
3. $\forall x \in [-1, 1], |p(x)| \leq 1$.

The circuit diagram of $U_{\boldsymbol{\theta}}(x)$ is shown in Figure 2. We have several different methods to estimate $\langle +|U_{\boldsymbol{\theta}}(x)|+\rangle$ to obtain $p(x)$. In this work, we consider only the Hadamard test method, whose circuit diagram is shown in Figure C1.

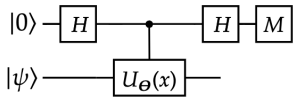


**Fig. C1** Hadamard test to estimate the real part of $\langle\psi|U_{\boldsymbol{\theta}}(x)|\psi\rangle$.

For the Hadamard test method, regardless of the estimated term, it is necessary to measure the first qubit of the circuit $O\left(1/\varepsilon^2\right)$ times in order to estimate the expected value within an error bound $\varepsilon$. In general, $\langle\psi|U|\psi\rangle$ is complex for an arbitrary unitary $U$ and quantum state $|\psi\rangle$. The circuit shown in Figure C1 can only access the real part of $\langle\psi|U|\psi\rangle$, and therefore cannot recover the full complex value. However, when $\langle +|U_\theta(x)|+\rangle$ is real, it allows us to use the circuit in Figure C1 to exactly recover $p(x)$. The same idea applies to other results.

Although Lemma 5 relies on a counter-intuitive parity constraint, this condition can be relaxed. In the following theorem, we strengthen Lemma 5 by removing its parity constraint—at the cost of introducing a $1/2$ normalizing factor, which in practice causes no significant difficulties.

**Corollary 6** There exists $\boldsymbol{\theta}_1 \in \mathbb{R}^L, \boldsymbol{\theta}_2 \in \mathbb{R}^{L+1}$ such that

$$p(x) = \langle +|^{\otimes 2}(|0\rangle\langle 0| \otimes U_{\boldsymbol{\theta_1}}(x) + |1\rangle\langle 1| \otimes U_{\boldsymbol{\theta_2}}(x))|+\rangle^{\otimes 2},$$

where $U_{\boldsymbol{\theta_1}}(x), U_{\boldsymbol{\theta_2}}(x)$ are defined as Equation C17 if the real polynomial $p(x) \in \mathbb{R}[x]$ satisfies

1. $\deg(p(x)) \leq L$,
2. $\forall x \in [-1, 1], |p(x)| \leq \frac{1}{2}$.

*Proof* Let $p(x) = \frac{1}{2}p_{\text{odd}}(x) + \frac{1}{2}p_{\text{even}}(x)$ where $p_{\text{even}}(x) = p(x) + p(-x)$ and $p_{\text{odd}}(x) = p(x) - p(-x)$. We have $\deg(p_{\text{even}}(x)) \leq L, \deg(p_{\text{odd}}(x)) \leq L - 1$ when $L$ is even, and $\deg(p_{\text{even}}(x)) \leq L - 1, \deg(p_{\text{odd}}(x)) \leq L$ when $L$ is odd. We also know that $|p_{\text{odd}}(x)| \leq 1$ and $|p_{\text{even}}(x)| \leq 1$ since $|p(x)| \leq \frac{1}{2}$. Hence from the Lemma 5, we know there exist $\boldsymbol{\theta}_{\text{odd}} \in \mathbb{R}^L, \boldsymbol{\theta}_{\text{even}} \in \mathbb{R}^{L+1}$ ($L$ is even) or $\boldsymbol{\theta}_{\text{odd}} \in \mathbb{R}^{L+1}, \boldsymbol{\theta}_{\text{even}} \in \mathbb{R}^L$ ($L$ is odd) such that

$$p_{\text{odd}}(x) = \langle +|U_{\boldsymbol{\theta}_{\text{odd}}}(x)|+\rangle$$

and

$$p_{\text{even}}(x) = \langle +|U_{\boldsymbol{\theta}_{\text{even}}}(x)|+\rangle$$

then we can build $|0\rangle\langle 0| \otimes U_{\boldsymbol{\theta}_{\text{odd}}} + |1\rangle\langle 1| \otimes U_{\boldsymbol{\theta}_{\text{even}}}$ such that

$$\begin{aligned} &\langle +|^{\otimes 2}(|0\rangle\langle 0| \otimes U_{\boldsymbol{\theta}_{\text{odd}}} + |1\rangle\langle 1| \otimes U_{\boldsymbol{\theta}_{\text{even}}})(x)|+\rangle^{\otimes 2} \\ = \frac{1}{2}\langle +|U_{\boldsymbol{\theta}_{\text{odd}}}(x)|+\rangle + \frac{1}{2}\langle +|U_{\boldsymbol{\theta}_{\text{even}}}(x)|+\rangle &= \frac{1}{2}p_{\text{odd}}(x) + \frac{1}{2}p_{\text{even}}(x) = p(x). \end{aligned}$$

□

In fact, $\forall x \in [-1, 1]$, the condition $|p(x)| \leq \frac{1}{2}$ is stronger than necessary to guarantee the existence of a quantum model. All that is truly required is

$$|p_{\text{odd}}(x)|_\infty, |p_{\text{even}}(x)|_\infty \leq 1, \tag{C18}$$

where $p_{\text{even}}(x) = p(x) + p(-x)$ and $p_{\text{odd}}(x) = p(x) - p(-x)$. However for simplicity, we henceforth replace these two separate bounds by the simple condition $|p(x)| \leq \frac{1}{2}$. Then we can do the Hadamard test to estimate $p(x) = \langle +|^{\otimes 2}(|0\rangle\langle 0| \otimes U_{\boldsymbol{\theta_1}}(x) + |1\rangle\langle 1| \otimes U_{\boldsymbol{\theta_2}}(x))|+\rangle^{\otimes 2}$ and analyze the circuit complexity.

**Proposition 1.** *For any real polynomial $p(x) \in \mathbb{R}[x]$ that satisfies $\deg(p(x)) \leq L$ and $\forall x \in [-1, 1], |p(x)| \leq \frac{1}{2}$, there exists a quantum model $\mathcal{Q}$ that consists of a PQC $W_p(\boldsymbol{x})$ and an observable $Z^{(0)}$ such that*

$$f_{\mathcal{Q}}(x) := \langle 0|W_p^\dagger(x) Z^{(0)} W_p(\boldsymbol{x})|0\rangle = p(x),$$

*where $Z^{(0)}$ is the Pauli $Z$ observable on the first qubit. The width of the PQC is at most 3, the number of parameters is at most $2L + 1$. The PQC $W_p(\boldsymbol{x})$ can be expressed as at most $4L$ double-controlled rotation gates and 4 Hadamard gates, with depth $4L + 2$. Alternatively, it can be expressed using $36L$ single-qubit gates and $32L$ CNOT gates, with depth $60L - 5$.*

The circuit diagram of $W_p(\boldsymbol{x})$ is shown in Figure 3.

*Proof* We use Corollary 6 to build the circuit

$$W_p(\boldsymbol{x}) = (H \otimes H \otimes H)\left(|0\rangle\langle 0| \otimes U_{\boldsymbol{\theta_1}}(x) + |1\rangle\langle 1| \otimes U_{\boldsymbol{\theta_2}}(x)\right)(H \otimes I \otimes I)$$

If we consider quantum computers that can implement multi-qubit controlled rotation gates such as neutral atom platforms, the depths of double controlled rotation gates $U_{\boldsymbol{\theta_1}}, U_{\boldsymbol{\theta_2}}$ are $2(L-1)+1, 2L+1$, respectively. Hence the PQC $W_p(\boldsymbol{x})$ consists of at most $4L$ 2-qubit controlled rotation gates and 4 Hadamard gates, and the circuit depth is $4L+2$. The numbers of parameters of $U_{\boldsymbol{\theta_1}}, U_{\boldsymbol{\theta_2}}$ are $(L-1)+1, L+1$ without counting $x$, respectively. So the total number of parameters is $2L + 1$.

If we consider quantum computers that cannot implement multi-qubit controlled rotation gates natively, from Barenco et al. (1995, Lemma 5.4) we know that for a controlled-$R_p$ gate where $R_p$ is the rotation gate with Pauli matrix $p \in \{z, y\}$, controlled-$R_p$ can be implemented by 2 $R_p$ gates and 2 CNOT gates, and a controlled-$R_x$ gate can be implemented by a controlled-$R_z$ plus 2 Hadamard gates. So a controlled-$R_z$ gate consists of 2 single-qubit gates and 2 CNOT gates with depth 4, and a controlled-$R_x$ gate consists of 4 single-qubit native

gates and 2 CNOT gates with depth 6 because of $R_x(\theta) = HR_z(\theta)H$ where $H$ is Hadamard gates.

Then from Barenco et al. (1995, Lemma 6.1) we also know that a double controlled-$U$ gate can be implemented by 2 controlled-$V$ (where $V^2 = U$), 1 controlled-$V^\dagger$ and 2 CNOT gates, so a double controlled-$R_z$ can be implemented by 6 single-qubit gates and 8 CNOT gates with depth 12, and a double controlled-$R_x$ can be implemented by 12 single-qubit gates and 8 CNOT gates with depth 18. The depth arises from the specific circuit layout described in Barenco et al. (1995, Lemma 6.1).

A double-controlled-$U_{\boldsymbol{\theta_1}}$ can be implemented by $L-1$ double controlled $R_x$ gates and $L$ double controlled $R_z$ gates, which means $12(L-1)+6L$ single-qubit gates and $8(L-1)+8L$ CNOT gates with depth $18(L-1)+12L = 30L-18$. And a double-controlled-$U_{\boldsymbol{\theta_2}}$ can be implemented by $L$ $R_x$ gates and $L+1$ $R_z$ gates, which means $12L+6(L+1)$ single-qubit gates and $8L+8(L+1)$ CNOT gates with depth $18L+12(L+1) = 30L+12$.

Finally, with the extra 4 Hadamard gates and 2 Pauli-X gates (for controlled $|0\rangle$ to $|1\rangle$), we can conclude that the circuit requires $36L$ single-qubit gates and $32L$ CNOT gates with depth $60L-5$. Here, the extra depth due to Pauli-X gates can be ignored. It is worth noting that the bound we provide, derived from Barenco et al. (1995), is only an old upper bound. Improved results may exist, yielding tighter estimates for both gate count and circuit depth. □

In fact, Proposition 1 is similar to the result in Gilyén et al. (2019, Theorem 56), but we don't use their block-encoded method. Moreover, our result provides an extension to this existing work in two perspectives: clear quantum circuit implementation of univariate polynomials and exact quantification of all circuit resources. Actually, we can implement multi-qubit controlled rotation gates on the neutral atom platform quantum computer in an efficient and native way (Li et al. 2021). We observe that building multi-qubit quantum gates directly in a "native" method is more efficient and valuable than constructing multi-qubit gates through a decomposition of two-qubit and single-qubit gates.

Lemma 5 and Corollary 6 give us quantum circuit expressivity analyzes for univariate polynomials, next we will talk about the multivariate polynomial cases.

### C.1.2 Multivariate case

We now turn to discuss how to implement multivariate polynomials, and we define the $L^\infty$-norm of a polynomial with input variable $x \in E$ as $\|p\|_\infty := \sup_{x\in E} |p(x)|$.

**Corollary 7** Given a multivariate polynomial $p(\boldsymbol{x}) = \prod_{j=1}^{D} p_j(x_j)$ in $\boldsymbol{x} = (x_1, \ldots, x_D) \in [-1,1]^D$ such that $\deg(p_j(x_j)) \leq L$, $p_j(x_j)$ has parity $L \bmod 2$, and $\|p_j\|_\infty \leq 1$, there exists a PQC $U^p(\boldsymbol{x})$ such that

$$\langle +|^{\otimes D} U^p(\boldsymbol{x}) |+\rangle^{\otimes D} = p(\boldsymbol{x}).$$

The width of the PQC is at most $D$, the depth is at most $L+1$, and the number of parameters is at most $LD$.

The circuit diagram of $U^p(\boldsymbol{x})$ is shown in Figure 4.

*Proof* From Lemma 5, there exist $D$ single-qubit PQCs $U_{\boldsymbol{\theta}_1}^{p_1}(x_1), U_{\boldsymbol{\theta}_2}^{p_2}(x_2), \ldots, U_{\boldsymbol{\theta}_D}^{p_D}(x_D)$ such that

$$\begin{aligned} \langle +|U_{\boldsymbol{\theta}_1}^{n_1}(x_1)|+\rangle &= p_1(x_1), \\ \langle +|U_{\boldsymbol{\theta}_2}^{n_2}(x_2)|+\rangle &= p_2(x_2), \\ &\cdots \\ \langle +|U_{\boldsymbol{\theta}_D}^{n_D}(x_D)|+\rangle &= p_D(x_D), \end{aligned} \tag{C19}$$

where each real polynomial $p_j(x_j) \in \mathbb{R}[x_j], \forall j \in [D]$ satisfies $\deg(p_j(x_j)) \leq L$, $p_j(x_j)$ has parity $L \bmod 2$, and $\|p_j\|_\infty \leq 1$. Then we can define $p(\boldsymbol{x}) = \prod_j^D p_j(x_j)$ and

$$U^p(\boldsymbol{x}) = \bigotimes_{j=1}^{D} U_{\boldsymbol{\theta}_j}^{p_j}(x_j), \tag{C20}$$

which gives

$$\langle +|^{\otimes D} U^p(\boldsymbol{x})|+\rangle^{\otimes D} = \prod_{j=1}^{D} \langle +|U_{\boldsymbol{\theta}_j}^{p_j}(x_j)|+\rangle = p(\boldsymbol{x}).$$

We can conclude that the depth of $U^p(\boldsymbol{x})$ is at most $L+1$, and the number of parameters is at most $LD$. □

Based on Corollary 7, we can also consider the univariate constrained polynomial $p_j(x_j)$ as a monomial:

**Corollary 8** Given a monomial $c_{\boldsymbol{n}}\boldsymbol{x}^{\boldsymbol{n}} = c_{\boldsymbol{n}} x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$ in $\boldsymbol{x} \in [-1,1]^D$ such that $|c_{\boldsymbol{n}}| \leq 1$ and $\|\boldsymbol{n}\|_\infty \leq L$ for $\boldsymbol{n} = (n_1, \ldots, n_D) \in [L]^D$, there exists a PQC $U^{\boldsymbol{n}}(\boldsymbol{x})$ such that

$$\langle +|^{\otimes D} U^{\boldsymbol{n}}(\boldsymbol{x})|+\rangle^{\otimes D} = c_{\boldsymbol{n}}\boldsymbol{x}^{\boldsymbol{n}}.$$

The width of the PQC is at most $D$, the depth is at most $L+1$, and the number of parameters is at most $LD$.

*Proof* We can regard the univariate constrained polynomial $p_j(x_j)$ in Corollary 7 as a monomial, and the resource required would be the same. □

Corollary 8 is almost the same as Yu et al. (2024, Lemma S5), but we extend the domain of definition of $x$ from $[0,1]^D$ to $[-1,1]^D$, and adjust the condition from $\|\boldsymbol{n}\|_1 \leq L$ to $\|\boldsymbol{n}\|_\infty \leq L$. Since monomials can be expressed explicitly, any real multivariate polynomial can be written as a sum of monomials using the LCU technique. Hence, we have:

**Proposition 9** *For any real multivariate polynomial $p(\boldsymbol{x}) = \sum_{\boldsymbol{n}} c_{\boldsymbol{n}}\boldsymbol{x}^{\boldsymbol{n}}$ with $D$ variables and degree at most $L$ in each variable such that $\|p\|_\infty \leq 1$, $\boldsymbol{x} \in [-1,1]^D$, and the absolute values of all coefficients are smaller than $1/T$ where $T$ is the number of monomials of $p(\boldsymbol{x})$, there exists a quantum model $\mathcal{Q}$ that consists of a PQC $W_p(\boldsymbol{x})$ and an observable $Z^{(0)}$ such that*

$$f_{\mathcal{Q}}(\boldsymbol{x}) := \langle 0|W_p^\dagger(\boldsymbol{x}) Z^{(0)} W_p(\boldsymbol{x})|0\rangle = p(\boldsymbol{x}),$$

*where $Z^{(0)}$ is the Pauli $Z$ observable on the first qubit. The width of the PQC is at most $D + \lceil \log T \rceil + 1$, the depth $O(TDL \log T)$, and the number of parameters is at most $TD(L+1)$.*

The circuit diagram of $W_p(\boldsymbol{x})$ is shown in Figure C2.

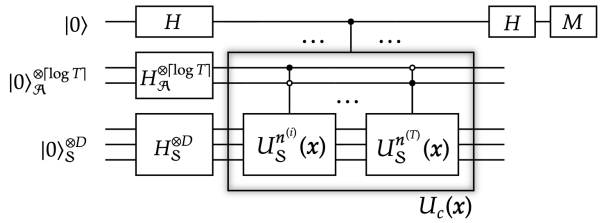


**Fig. C2** The circuit of $W_p(\boldsymbol{x})$ consists of a single-qubit system, a $\lceil \log T \rceil$-qubits system $\mathcal{A}$ where $T$ is the number of monomials of $p(\boldsymbol{x})$, and a $D$-qubits system $\mathcal{S}$. We can consider $W_p(\boldsymbol{x})$ as the Hadamard test to estimate $p(\boldsymbol{x}) = \langle +|_{\mathcal{A}}^{\otimes \lceil \log T \rceil} \langle +|_{\mathcal{S}}^{\otimes D} U_c(\boldsymbol{x}) |+\rangle_{\mathcal{A}}^{\otimes \lceil \log T \rceil} |+\rangle_{\mathcal{S}}^{\otimes D}$ such that $U_c(\boldsymbol{x}) = \sum_{i=1}^{T} |i\rangle_{\mathcal{A}} \langle i| \otimes U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x})$ where $U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x})$ is defined in Equation C20 and shown in Figure 4.

*Proof* Consider the given real multivariate polynomial $p(\boldsymbol{x})$ as

$$p(\boldsymbol{x}) = \sum_{i=1}^{T} c_{\boldsymbol{n}^{(i)}} \boldsymbol{x}^{\boldsymbol{n}^{(i)}} \quad \text{with } |c_{\boldsymbol{n}^{(i)}}| \leq \frac{1}{T}.$$

where $\boldsymbol{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D, \boldsymbol{n} = (n_1, \ldots, n_D) \in [L]^D, c_{\boldsymbol{n}} \in \mathbb{R}$, $\boldsymbol{x}^{\boldsymbol{n}} = x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$, and $T$ is denoted as the number of monomial $c_{\boldsymbol{n}^{(i)}} \boldsymbol{x}^{\boldsymbol{n}^{(i)}}$ and we have $T \leq (L+1)^D$. We want to use a quantum circuit to express $p(\boldsymbol{x})$, let's define a controlled unitary $U_c(\boldsymbol{x})$ on the composite system $\mathcal{A} \otimes \mathcal{S}$ consists of the $\lceil \log T \rceil$-qubits ancilla system $\mathcal{A}$ and the $D$-qubits system $\mathcal{S}$ such that

$$U_c(\boldsymbol{x}) = \sum_{i=1}^{T} |i\rangle_{\mathcal{A}} \langle i| \otimes U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x})$$

and use Corollary 8:

$$\langle +|^{\otimes D} U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x}) |+\rangle^{\otimes D} = C_{\boldsymbol{n}^{(i)}} \boldsymbol{x}^{\boldsymbol{n}^{(i)}}$$

where we choose

$$C_{\boldsymbol{n}^{(i)}} = T\, c_{\boldsymbol{n}^{(i)}}, |C_{\boldsymbol{n}^{(i)}}| \leq 1, \quad \text{since } |c_{\boldsymbol{n}^{(i)}}| \leq \frac{1}{T}$$

and

$$H_{\mathcal{A}}^{\otimes \lceil \log T \rceil} |0\rangle_{\mathcal{A}}^{\otimes \lceil \log T \rceil} = \frac{1}{\sqrt{T}} \sum_{i=1}^{T} |i\rangle_{\mathcal{A}} = |+\rangle_{\mathcal{A}}^{\otimes \lceil \log T \rceil}$$

and we obtain

$$\begin{aligned}
&\langle +|_{\mathcal{A}}^{\otimes \lceil \log T \rceil} \langle +|_{\mathcal{S}}^{\otimes D} U_c(\boldsymbol{x}) |+\rangle_{\mathcal{A}}^{\otimes \lceil \log T \rceil} |+\rangle_{\mathcal{S}}^{\otimes D} \\
&= \langle +|_{\mathcal{A}}^{\otimes \lceil \log T \rceil} \langle +|_{\mathcal{S}}^{\otimes D} \left( \sum_{i=1}^{T} |i\rangle_{\mathcal{A}} \langle i| \otimes U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x}) \right) |+\rangle_{\mathcal{A}}^{\otimes \lceil \log T \rceil} |+\rangle_{\mathcal{S}}^{\otimes D} \\
&= \sum_{i=1}^{T} \langle +|_{\mathcal{A}}^{\otimes \lceil \log T \rceil} |i\rangle_{\mathcal{A}} \langle i|_{\mathcal{A}} |+\rangle_{\mathcal{A}}^{\otimes \lceil \log T \rceil} \langle +|_{\mathcal{S}}^{\otimes D} U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x}) |+\rangle_{\mathcal{S}}^{\otimes D} \\
&= \sum_{i=1}^{T} \left( \tfrac{1}{\sqrt{T}} \right) \left( \tfrac{1}{\sqrt{T}} \right) C_{\boldsymbol{n}^{(i)}} \boldsymbol{x}^{\boldsymbol{n}^{(i)}}
\end{aligned}$$

$$= \sum_{i=1}^{T} \frac{1}{T} C_{\boldsymbol{n}^{(i)}} \boldsymbol{x}^{\boldsymbol{n}^{(i)}} = \sum_{i=1}^{T} c_{\boldsymbol{n}^{(i)}} \boldsymbol{x}^{\boldsymbol{n}^{(i)}} = p(\boldsymbol{x})$$

This explains the reason we set $|c_{\boldsymbol{n}^{(i)}}| \leq \frac{1}{T}$, to satisfy $\left|C_{\boldsymbol{n}^{(i)}}\right| \leq 1$ so that we can use Corollary 8 to express all monomials and the polynomial. Then we still use Hadamard test to estimate

$$p(\boldsymbol{x}) = \langle +|_{\mathcal{A}}^{\otimes \lceil \log T \rceil} \langle +|_{\mathcal{S}}^{\otimes D} U_c(\boldsymbol{x}) |+\rangle_{\mathcal{A}}^{\otimes \lceil \log T \rceil} |+\rangle_{\mathcal{S}}^{\otimes D}$$

The controlled unitary $U_c(\boldsymbol{x}) = \sum_{i=1}^{T} |i\rangle_{\mathcal{A}} \langle i| \otimes U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x})$ can be implemented by $T$ occurrences of $U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x})$ controlled by $\lceil \log T \rceil$-qubits. From Equation C20, each $U_{\mathcal{S}}^{\boldsymbol{n}^{(i)}}(\boldsymbol{x}) = \bigotimes_{j=1}^{D} U_{\boldsymbol{\theta}_j}^{n_j}(x_j)$ can be implemented by $D$ occurrences of $U_{\boldsymbol{\theta}_j}^{n_j}(x_j)$ that defined in Equation C17, which consists of $O(L)$ occurrences of single-qubit rotation gates. Totally, a $U_c(\boldsymbol{x})$ can be implemented by $O(TDL)$ single-qubit rotation gates controlled by $\lceil \log T \rceil$-qubits.

We know that a single-qubit rotation gate controlled by $\lceil \log T \rceil$-qubits could be implemented by a quantum circuit consisting of CNOT gates and single-qubit gates with depth $O(\log T)$ (da Silva and Park 2022). Thus $U_c(\boldsymbol{x})$ could be implemented by a quantum circuit with depth $O(TDL \log T)$ and width $D + \lceil \log T \rceil$.

Then we build Hadamard test circuit $W_p^{\dagger}(\boldsymbol{x})$ to estimate $p(\boldsymbol{x})$ such that

$$W_p(\boldsymbol{x}) = (H \otimes H^{D+\lceil \log T \rceil}) \left(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U_c(\boldsymbol{x})\right) (H \otimes I \otimes I) \tag{C21}$$

The extra controlled-$U_c(\boldsymbol{x})$ and Hadamard gates don't change the depth complexity, but only increase the circuit width by 1. Hence we have the circuit $W_p^{\dagger}(\boldsymbol{x})$ with width $D + \lceil \log T \rceil + 1$, depth $O(TDL \log T)$, and the number of parameters $TD(L+1)$. □

As $\boldsymbol{n} = (n_1, \ldots, n_D) \in [L]^D$, the number of monomials $T \leq (L+1)^D$, we can adjust the Proposition 9 to eliminate the parameter $T$ so that the bound for coefficients absolute values becomes $1/(L+1)^D$. But keeping the coefficients so small obstructs optimization, and the maximum absolute value of circuit output is only 1. To solve this we can simply multiply the final output of the circuit by a factor $\Lambda$ to express more general polynomials while keeping the quantum model ($W_p(\boldsymbol{x})$ and $Z^{(0)}$) unchanged.

**Theorem 2.** *Let $p(\boldsymbol{x}) = \sum_{\boldsymbol{n}} c_{\boldsymbol{n}} \boldsymbol{x}^{\boldsymbol{n}}$ be any real multivariate polynomial with $D$ variables and degree at most $L$ in each variable such that $\boldsymbol{x} \in [-1,1]^D$, $c_{\boldsymbol{n}} \in \mathbb{R}$, $\boldsymbol{n} \in [L]^D$. Then there exists a quantum model $\mathcal{Q}$ that consists of a PQC $W_p(\boldsymbol{x})$ and an observable $Z^{(0)}$ with the scaling factor $\Lambda$ such that*

$$f_{\mathcal{Q}}(\boldsymbol{x}) := \langle 0 | W_p^{\dagger}(\boldsymbol{x}) Z^{(0)} W_p(\boldsymbol{x}) | 0 \rangle = p(\boldsymbol{x})/\Lambda,$$

*where $\Lambda = |c_{\boldsymbol{n}}|_{\infty}(L+1)^D$ and $Z^{(0)}$ is the Pauli $Z$ observable on the first qubit. The width of the PQC is $O(D \log L)$, the depth is $O\left(D^2 L^{D+1} \log L\right)$, and the number of parameters is $O\left(DL^{D+1}\right)$.*

The circuit diagram of $W_p(\boldsymbol{x})$ is shown in Figure 5.

*Proof* Following Proposition 9, replace $T$ by $(L+1)^D$, we then obtain the width is at most $D(\log(L+1)+1)+2$, the depth is $O\left((L+1)^{D+1} D^2 \log(L+1)\right)$, and the number of parameters is at most $(L+1)^D D((L+1)+1) + D$. Using the complexity notation, we have the

width is $O(D \log L)$, the depth is $O(D^2 L^{D+1} \log L)$, and the number of parameters is at most $O(DL^{D+1})$.

Then we need to satisfy the bound for coefficients absolute values, and the property that the quantum circuit measurements can only output values within 1 absolute value. We can divide the polynomial by a scaling factor $\Lambda$ to satisfy the two constraints together. Precisely, for any $p(\boldsymbol{x}) = \sum_{\boldsymbol{n}} c_{\boldsymbol{n}} \boldsymbol{x}^{\boldsymbol{n}}$, the maximum coefficients absolute values of $p(\boldsymbol{x})/\Lambda$ is $|c_{\boldsymbol{n}}|_\infty/\Lambda$ such that

$$|c_{\boldsymbol{n}}|_\infty/\Lambda \leq \frac{1}{(L+1)^D},$$

the smallest scaling factor is

$$\Lambda = |c_{\boldsymbol{n}}|_\infty (L+1)^D.$$

Once the bound condition for coefficients absolute values is satisfied, we also have $|p(\boldsymbol{x})/\Lambda|_\infty \leq 1$.

□

Theorem 2 is similar to the results of Yu et al. (2024, Theorem 1): For any multivariate polynomial $p(\boldsymbol{x}) = \sum_{\boldsymbol{n}} c_{\boldsymbol{n}} \boldsymbol{x}^{\boldsymbol{n}}$ with $D$ variables and $\|\boldsymbol{n}\|_1 \leq L$ such that $|p(\boldsymbol{x})| \leq 1$ for $\boldsymbol{x} \in [0,1]^D$, there exists a PQC $W_p(\boldsymbol{x})$ such that $\langle 0|W_p^\dagger(\boldsymbol{x}) Z^{(0)} W_p(\boldsymbol{x})|0\rangle = p(\boldsymbol{x})$, with PQC width $O(D + \log L + L \log D)$, the depth is $O\left(L^2 D^L (\log L + L \log D)\right)$, and the number of parameters is $O\left(LD^L(L+D)\right)$.

In fact, they considered a stricter condition $\|\boldsymbol{n}\|_1 \leq L$ instead of our $\|\boldsymbol{n}\|_\infty \leq L$ and use the bound $T \leq (L+1)D^L$ instead of our $T = (L+1)^D$, domain $\boldsymbol{x} \in [0,1]^D$ instead of our $\boldsymbol{x} \in [-1,1]^D$. Moreover, they should also need a scaling factor to ensure that the PQC is found for all monomials of eligible polynomials. Compared to their results, our condition not only precise the scaling factor bound, but also has fewer restrictions on degree and domain of input. Besides, our complexity result is clearly better when $L$ is larger than $D$, which is also the more common case in certain complex financial problems.

## C.2 Tensor-decomposed polynomial implementation

**Theorem 3.** *Let $p(\boldsymbol{x})$ be any real multivariate polynomial with $D$ variables and degree at most $L$ in each variable such that*

$$p(\boldsymbol{x}) = \sum_{i=1}^{R} \lambda_i \prod_{j=1}^{D} p_{i,j}(x_j),$$

*where $R \in [1, (L+1)^D]$, $\sum_{i=1}^{R} |\lambda_i| = \Lambda \in \mathbb{R}$, and each $p_{i,j}(x_j)$ is a univariate polynomial of degree $L$ satisfying $|p_{i,j}(x_j)| \leq 1/2$ for $x_j \in [-1,1], \forall i \in [R], j \in [D]$. Then there exists a quantum model $\mathcal{Q}$ that consists of a PQC $W_p(\boldsymbol{x})$ and an observable $Z^{(0)}$ such that*

$$f_{\mathcal{Q}}(\boldsymbol{x}) := \langle 0|W_p^\dagger(\boldsymbol{x}) Z^{(0)} W_p(\boldsymbol{x})|0\rangle = p(\boldsymbol{x})/\Lambda,$$

*where $Z^{(0)}$ is the Pauli $Z$ observable on the first qubit. The width of the PQC is at most $2D + \lceil \log R \rceil + 1$, the depth is $O(RDL \log R)$, and the number of parameters is $O(RDL)$.*

The circuit diagram of the PQC $W_p(\boldsymbol{x}) = (H \otimes F_{\mathcal{A}} \otimes H_{\mathcal{S}}^{\otimes 2D})\,(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes V_c(\boldsymbol{x}))\,(H \otimes I_{\mathcal{A}} \otimes I_{\mathcal{S}})$, with $V_c(\boldsymbol{x}) = \sum_{i=1}^{R} |i\rangle_{\mathcal{A}} \langle i| \otimes V_{\mathcal{S}}^{(i)}(\boldsymbol{x})$, and $V_{\mathcal{S}}^{(i)}(\boldsymbol{x})$ are shown in Figure 6 and Figure 7, respectively.

*Proof* For each term $i \in [R]$ and each variable $x_j$, apply Corollary 6 to the univariate polynomial $p_{i,j}(x_j)$. Since $\deg(p_{i,j}) \le L$ and $|p_{i,j}(x_j)| \le \frac{1}{2}$, there exist angles $\boldsymbol{\theta}_{i,j}^{(1)} \in \mathbb{R}^{L}$ and $\boldsymbol{\theta}_{i,j}^{(2)} \in \mathbb{R}^{L+1}$ such that

$$p_{i,j}(x_j) = \langle +|^{\otimes 2}\Big(|0\rangle\langle 0| \otimes U_{\boldsymbol{\theta}_{i,j}^{(1)}}(x_j) + |1\rangle\langle 1| \otimes U_{\boldsymbol{\theta}_{i,j}^{(2)}}(x_j)\Big)|+\rangle^{\otimes 2}.$$

Define a $2D$-qubit unitary for each $i$ by

$$V^{(i)}(\boldsymbol{x}) = \bigotimes_{j=1}^{D}\Big(|0\rangle\langle 0| \otimes U_{\boldsymbol{\theta}_{i,j}^{(1)}}(x_j) + |1\rangle\langle 1| \otimes U_{\boldsymbol{\theta}_{i,j}^{(2)}}(x_j)\Big),$$

and consider $V^{(i)}$ in the $2D$ qubit system $\mathcal{S}$ such that,

$$\langle +|_{\mathcal{S}}^{\otimes 2D} V_{\mathcal{S}}^{(i)}(\boldsymbol{x})\,|+\rangle_{\mathcal{S}}^{\otimes 2D} = \prod_{j=1}^{D} p_{i,j}(x_j). \tag{C22}$$

Define $F_{\mathcal{A}}$ on the ancilla register $\mathcal{A}$ of $\lceil \log R \rceil$ qubits such that

$$F_{\mathcal{A}}\,|0\rangle_{\mathcal{A}}^{\otimes \lceil \log R \rceil} = \sum_{i=1}^{R} \sqrt{w_i}\,|i\rangle_{\mathcal{A}} \quad \text{where } w_i = \frac{|\lambda_i|}{\sum_{k=1}^{R}|\lambda_k|}$$

with $\sum_{i=1}^{R} w_i = 1$, and define the controlled unitary $V_c(\boldsymbol{x})$ such that

$$V_c(\boldsymbol{x}) = \sum_{i=1}^{R} |i\rangle\langle i|_{\mathcal{A}} \otimes V_{\mathcal{S}}^{(i)}(\boldsymbol{x}).$$

Define $|v_p\rangle = \sum_{i=1}^{R} \sqrt{w_i}\,|i\rangle_{\mathcal{A}} \otimes |+\rangle_{\mathcal{S}}^{\otimes 2D}$, we have

$$\begin{aligned}
&\langle v_p|\,V_c(\boldsymbol{x})\,|v_p\rangle \\
&= \Big(\sum_{i'=1}^{R} \sqrt{w_{i'}}\,\langle i'|_{\mathcal{A}} \otimes \langle +|_{\mathcal{S}}^{\otimes 2D}\Big)\Big(\sum_{i=1}^{R} |i\rangle_{\mathcal{A}}\langle i| \otimes V_{\mathcal{S}}^{(i)}(\boldsymbol{x})\Big)\Big(\sum_{k=1}^{R} \sqrt{w_k}\,|k\rangle_{\mathcal{A}} \otimes |+\rangle_{\mathcal{S}}^{\otimes 2D}\Big) \\
&= \sum_{i',i,k} \sqrt{w_{i'}w_k}\,\langle i'|i\rangle_{\mathcal{A}}\langle i|k\rangle_{\mathcal{A}}\,\langle +|V_{\mathcal{S}}^{(i)}(\boldsymbol{x})|+\rangle_{\mathcal{S}}^{\otimes 2D} = \sum_{i=1}^{R} w_i\,\langle +|_{\mathcal{S}}^{\otimes 2D} V_{\mathcal{S}}^{(i)}(\boldsymbol{x})\,|+\rangle_{\mathcal{S}}^{\otimes 2D} \\
&= \sum_{i=1}^{R} w_i \prod_{j=1}^{D} p_{i,j}(x_j) \\
&= \frac{1}{\sum_{k=1}^{R}|\lambda_k|}\sum_{i=1}^{R}|\lambda_i| \prod_{j=1}^{D} p_{i,j}(x_j) \\
&\equiv \frac{1}{\sum_{k=1}^{R}|\lambda_k|}\sum_{i=1}^{R}\lambda_i \prod_{j=1}^{D} p_{i,j}(x_j) \\
&= p(\boldsymbol{x})/\Lambda.
\end{aligned}$$

In the second-to-last row, we can assume $\sum_{i=1}^{R} |\lambda_i| \prod_{j=1}^{D} p_{i,j}(x_j) \equiv \sum_{i=1}^{R} \lambda_i \prod_{j=1}^{D} p_{i,j}(x_j)$ without loss of generality, since the sign of $\lambda_i$ can always be absorbed into one of the univariate polynomials $p_{i,j}\left(x_j\right)$. Then we can do the Hadamard test to estimate $\langle v_p| V_c(\boldsymbol{x}) |v_p\rangle = p(\boldsymbol{x})/\Lambda$.

Similar ideas with Proposition 9, $V_c(\boldsymbol{x})$ could be implemented by a quantum circuit with depth $O(RDL \log R)$, and width $2D + \lceil \log R \rceil$. For the Hadamard test part, from (Möttönen et al. 2005), we know $F$ can be implemented by a quantum circuit of CNOT gates and single-qubit rotation gates size $O(R)$, so the depth and width of circuit is still the same. The extra controlled-$V_c(\boldsymbol{x})$ gates introduced by Hadamard test also doesn't change the complexity. Note that the number of parameters in the PQC equals the number of parameters in $V_c(\boldsymbol{x})$ plus the number of parameters in $F$, i.e. $O(RDL)$ and $O(R)$, which is still $O(RDL)$ totally. □

### C.2.1 Rank-1 tensor-decomposed polynomial implementation

In this work, we specifically consider a simple case when $R = 1$, i.e. rank-1 multivariate polynomial $p(\boldsymbol{x})$:

**Corollary 4.** *Let $p(\boldsymbol{x}) = \prod_{j=1}^{D} p_j\left(x_j\right)$ be any real multivariate polynomial such that $\forall j \in [D], x_j \in [-1, 1]$, each $p_{i,j}(x_j)$ is a univariate polynomial of degree $L$ satisfying $|p_{i,j}(x_j)| \leq 1/2$. Then there exists a quantum model $\mathcal{Q}$ that consists of a PQC $W_p(\boldsymbol{x})$ and an observable $Z^{(0)}$ such that*

$$f_{\mathcal{Q}}(\boldsymbol{x}) := \langle 0|W_p^{\dagger}(\boldsymbol{x}) Z^{(0)} W_p(\boldsymbol{x})|0\rangle = p(\boldsymbol{x}),$$

*where $Z^{(0)}$ is the Pauli $Z$ observable on the first qubit. The width of the PQC is at most $2D+1$, the depth is at most $4LD+2$ by allowing double-controlled rotation gates to be implemented natively, and the number of parameters is at most $(2L+1)D$.*

*Proof* For each variable, the depth of the double-controlled rotation gates is at most $2L+1+2(L-1)+1 = 4L$ and the number of parameters is at most $2L+1$. Thus, with $D$ variables and including the additional Hadamard layers, the total circuit depth is $4LD+2$, whereas the total number of parameters is at most $(2L+1)D$. □

## References


Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. Physical Review A **52**(5), 3457–3467 (1995) https://doi.org/10.1103/physreva.52.3457

Berry, D.W., Childs, A.M., Ostrander, A., Wang, G.: Quantum algorithm for linear differential equations with exponentially improved dependence on precision. Communications in Mathematical Physics **356**(3), 1057–1081 (2017) https://doi.org/10.1007/s00220-017-3002-y

Berger, S., Hosters, N., Möller, M.: Trainable embedding quantum physics informed neural networks for solving nonlinear pdes. Scientific Reports **15**(1), 18823 (2025) https://doi.org/10.1038/s41598-025-02959-z

Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. In: Samuel J. Lomonaco, J., Brandt, H.E. (eds.) Quantum Computation and Information. Contemporary Mathematics, vol. 305, pp. 53–74. American Mathematical Society, Providence, RI (2002). https://doi.org/10.1090/conm/305/05215

Benedetti, M., Lloyd, E., Sack, S., Fiorentini, M.: Parameterized quantum circuits as machine learning models. Quantum Science and Technology **4**(4), 043001 (2019) https://doi.org/10.1088/2058-9565/ab4eb5

Boyd, J.P.: Chebyshev and Fourier Spectral Methods, 2nd edn. Dover Publications, Mineola, New York (2001)

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum machine learning. Nature **549**(7671), 195–202 (2017) https://doi.org/10.1038/nature23474

Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S.C., Endo, S., Fujii, K., McClean, J.R., Mitarai, K., Yuan, X., Cincio, L., Coles, P.J.: Variational quantum algorithms. Nature Reviews Physics **3**(9), 625–644 (2021) https://doi.org/10.1038/s42254-021-00348-9

Chang, S.Y., Cerezo, M.: A Primer on Quantum Machine Learning (2025). https://arxiv.org/abs/2511.15969

Chia, N.-H., Gilyén, A., Li, T., Lin, H.-H., Tang, E., Wang, C.: Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing. STOC 2020, pp. 387–400. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3357713.3384314 . https://doi.org/10.1145/3357713.3384314

Cotler, J., Huang, H.-Y., McClean, J.R.: Revisiting dequantization and quantum advantage in learning tasks (2021). https://arxiv.org/abs/2112.00811

Cerezo, M., Larocca, M., García-Martín, D., Diaz, N.L., Braccia, P., Fontana, E., Rudolph, M.S., Bermejo, P., Ijaz, A., Thanasilp, S., Anschuetz, E.R., Holmes, Z.: Does provable absence of barren plateaus imply classical simulability? Nature Communications **16**(1) (2025) https://doi.org/10.1038/s41467-025-63099-6

Chrysos, G.G., Moschoglou, S., Bouritsas, G., Deng, J., Panagakis, Y., Zafeiriou, S.P.: Deep polynomial neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1–1 (2021) https://doi.org/10.1109/tpami.2021.3058891

Chen, Z., Shaviner, G.G., Chandravamsi, H., Pisnoy, S., Frankel, S.H., Pereg, U.: Quantum physics-informed neural networks for maxwell's equations: circuit design, "black hole" barren plateaus mitigation, and gpu acceleration. Quantum Machine Intelligence **8**(1), 21 (2026) https://doi.org/10.1007/s42484-026-00365-w

Cohen, N., Sharir, O., Shashua, A.: On the expressive power of deep learning: A tensor analysis. In: Feldman, V., Rakhlin, A., Shamir, O. (eds.) Proceedings of the 29th Conference on Learning Theory (COLT 2016). Proceedings of Machine Learning Research, vol. 49, pp. 698–728. PMLR, New York, NY, USA (2016). https://proceedings.mlr.press/v49/cohen16.html

Di Persio, L., Fraccarolo, N.: Exploring optimisation strategies under jump-diffusion dynamics. Mathematics **13**(3) (2025) https://doi.org/10.3390/math13030535

Silva, A.J., Park, D.K.: Linear-depth quantum circuits for multiqubit controlled gates. Physical Review A **106**(4) (2022) https://doi.org/10.1103/physreva.106.042602

Farea, A., Khan, S., Serdar Celebi, M.: Qcpinn: quantum-classical physics-informed neural networks for solving pdes. Machine Learning: Science and Technology **6**(4), 045053 (2025) https://doi.org/10.1088/2632-2153/ae1c91

Gaitan, F.: Finding flows of a navier–stokes fluid through quantum computing. npj Quantum Information **6**, 61 (2020) https://doi.org/10.1038/s41534-020-00291-0

Giovannetti, V., Lloyd, S., Maccone, L.: Quantum random access memory. Physical Review Letters **100**(16) (2008) https://doi.org/10.1103/physrevlett.100.160501

Gollier, C.: The Economics of Risk and Time, pp. 43–54. MIT Press, Cambridge, MA (2001). https://doi.org/10.7551/mitpress/2622.001.0001

Gilyén, A., Su, Y., Low, G.H., Wiebe, N.: 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, pp. 193–204. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3313276.3316366 . https://doi.org/10.1145/3313276.3316366

Gonzalo, V., Wahl, M., Zagst, R.: Dynamic portfolio optimization using information from a crisis indicator. Mathematics **13**, 2664 (2025) https://doi.org/10.3390/math13162664

Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Physical Review Letters **103**(15) (2009) https://doi.org/10.1103/physrevlett.103.150502

Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. Journal of Mathematics and Physics **6**, 164–189 (1927)

Hunout, J., Laizet, S., Iannucci, L.: Variational quantum algorithm based on lagrange polynomial encoding to solve differential equations. Physical Review A **111**(6) (2025) https://doi.org/10.1103/physreva.111.062404

Hegde, P.R., Markidis, S.: Quantum physics informed neural networks. In: Workshop Proceedings of the 53rd International Conference on Parallel Processing. ICPP Workshops '24, pp. 114–115. Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3677333.3678272

Kacewicz, B.: Almost optimal solution of initial-value problems by randomized and quantum algorithms. J. Complex. **22**(5), 676–690 (2006)

Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review **51**(3), 455–500 (2009) https://doi.org/10.1137/07070111X

Kandala, A., Mezzacapo, A., Temme, K., Takita, M., Brink, M., Chow, J.M., Gambetta, J.M.: Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature **549**(7671), 242–246 (2017) https://doi.org/10.1038/nature23879

Kopeliovich, Y., Pokojovy, M.: Portfolio optimization with feedback strategies based on artificial neural networks. Finance Research Letters **69**, 106185 (2024) https://doi.org/10.1016/j.frl.2024.106185

Kyriienko, O., Paine, A.E., Elfving, V.E.: Solving nonlinear differential equations with differentiable quantum circuits. Phys. Rev. A **103**, 052416 (2021) https://doi.org/10.1103/PhysRevA.103.052416

Kilianová, S., Trnovska, M.: Robust portfolio optimization via solution to the hamilton–jacobi–bellman equation. International Journal of Computer Mathematics **93**, 1–10 (2014) https://doi.org/10.1080/00207160.2013.871542

Li, M., Guo, F.-Q., Jin, Z., Yan, L.-L., Liang, E.-J., Su, S.-L.: Multiple-qubit controlled unitary quantum gate for rydberg atoms using shortcut to adiabaticity and optimized geometric quantum operations. Phys. Rev. A **103**, 062607 (2021) https://doi.org/10.1103/PhysRevA.103.062607

Lantigua, S., Giraldi, G., Portugal, R.: Classical-quantum hybrid architecture for physics-informed neural networks. Phys. Rev. A, (2026) https://doi.org/10.1103/fdd3-qz1s

Logan, J.D.: Applied Partial Differential Equations, 3rd edn. Undergraduate Texts in Mathematics. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-12493-3

Luo, K., Zhao, J., Wang, Y., Li, J., Wen, J., Liang, J., Soekmadji, H., Liao, S.: Physics-informed neural networks for pde problems: a comprehensive review. Artificial Intelligence Review **58**(323) (2025) https://doi.org/10.1007/s10462-025-11322-7

Markidis, S.: On physics-informed neural networks for quantum computers. Frontiers in Applied Mathematics and Statistics **8** (2022)

McClean, J.R., Boixo, S., Smelyanskiy, V.N., Babbush, R., Neven, H.: Barren plateaus in quantum neural network training landscapes. Nature Communications **9**(1) (2018) https://doi.org/10.1038/s41467-018-07090-4

Merton, R.: Lifetime portfolio selection under uncertainty: The continuous-time case. The Review of Economics and Statistics **51**, 247–57 (1969) https://doi.org/10.2307/1926560

Matteo, O.D., Gheorghiu, V., Mosca, M.: Fault-tolerant resource estimation of quantum random-access memories. IEEE Transactions on Quantum Engineering **1**, 1–13 (2020) https://doi.org/10.1109/tqe.2020.2965803

Mitarai, K., Negoro, M., Kitagawa, M., Fujii, K.: Quantum circuit learning. Physical Review A **98**(3) (2018) https://doi.org/10.1103/physreva.98.032309

Möttönen, M., Vartiainen, J.J., Bergholm, V., Salomaa, M.M.: Transformation of quantum states using uniformly controlled rotations. Quantum Info. Comput. **5**(6), 467–473 (2005)

Nikol'skii, S.M.: Analysis III: Spaces of Differentiable Functions. Encyclopaedia of Mathematical Sciences, vol. 91. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36247-3

Oz, F., San, O., Kara, K.: An efficient quantum partial differential equation solver with chebyshev points. Scientific Reports **13**, 7767 (2023) https://doi.org/10.1038/s41598-023-34966-3

Oz, F., Vuppala, R.K.S.S., Kara, K., Gaitan, F.: Solving burgers' equation with quantum computing. Quantum Information Processing **21**(1), 30 (2022) https://doi.org/10.1007/s11128-021-03391-8

Panichi, G., Corli, S., Prati, E.: Quantum physics-informed neural networks for multivariable partial differential equations. Phys. Rev. Appl. **25**, 014001 (2026) https://doi.org/10.1103/6nh4-yh2y

Powell, M.J.D.: Approximation Theory and Methods. Cambridge University Press, New York, USA (1981)

Preskill, J.: Quantum computing in the nisq era and beyond. Quantum **2**, 79 (2018) https://doi.org/10.22331/q-2018-08-06-79

Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics **378**, 686–707

(2019) https://doi.org/10.1016/j.jcp.2018.10.045

Schreiber, F.J., Eisert, J., Meyer, J.J.: Classical surrogates for quantum learning models. Phys. Rev. Lett. **131**, 100803 (2023) https://doi.org/10.1103/PhysRevLett.131.100803

Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing **26**(5), 1484–1509 (1997) https://doi.org/10.1137/s0097539795293172

Schuld, M., Sweke, R., Meyer, J.J.: Effect of data encoding on the expressive power of variational quantum-machine-learning models. Phys. Rev. A **103**, 032430 (2021) https://doi.org/10.1103/PhysRevA.103.032430

Tang, S., Feng, X., Wu, W., Xu, H.: Physics-informed neural networks combined with polynomial interpolation to solve nonlinear partial differential equations. Computers & Mathematics with Applications **132**, 48–62 (2023) https://doi.org/10.1016/j.camwa.2022.12.008

Trahan, C., Loveland, M., Dent, S.: Quantum physics-informed neural networks. Entropy **26**(8) (2024) https://doi.org/10.3390/e26080649

Vemuri, S.K., Büchner, T., Niebling, J., Denzler, J.: Functional tensor decompositions for physics-informed neural networks. In: Antonacopoulos, A., Chaudhuri, S., Chellappa, R., Liu, C.-L., Bhattacharya, S., Pal, U. (eds.) Pattern Recognition, pp. 32–46. Springer, Cham (2025)

Wang, X., Hu, L.: A new method to solve the hamilton-jacobi-bellman equation for a stochastic portfolio optimization model with boundary memory. Journal of Industrial and Management Optimization **18**(6), 3831–3845 (2022) https://doi.org/10.3934/jimo.2021137

Yu, Z., Chen, Q., Jiao, Y., Li, Y., Lu, X., Wang, X., Yang, J.Z.: Non-asymptotic approximation error bounds of parameterized quantum circuits. In: The Thirty-eighth Annual Conference on Neural Information Processing Systems (2024). https://openreview.net/forum?id=XCkII8nCt3

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., Hsieh, C.-J.: Large batch optimization for deep learning: Training bert in 76 minutes. In: International Conference on Learning Representations (ICLR) 2020 (2020). https://arxiv.org/abs/1904.00962