Neural Quantum Spectral Operator Learning for Solving Partial Differential Equations

Chanyoung Kim[1] **Myeonghwan Seong**[2] **Yujin Kim**[2]

Daniel K. Park[2,3,4†] **Youngjoon Hong**[5,†]

[1]Interdisciplinary Program in Artificial Intelligence, Seoul National University
[2]Department of Statistics and Data Science, Yonsei University
[3]Department of Applied Statistics, Yonsei University
[4]Department of Quantum Information, Yonsei University
[5]Department of Mathematical Sciences, Seoul National University

chanyoung.kim@snu.ac.kr
{europa0306, k.yujin2228, dkd.park}@yonsei.ac.kr
hongyj@snu.ac.kr

Abstract

Partial differential equations (PDEs) are central to modeling physical and engineering systems, but repeatedly solving parametric PDEs remains computationally expensive. Operator learning enables fast surrogate inference, yet typically requires large input–output paired datasets generated by costly high-fidelity PDE solvers. Unsupervised operator learning frameworks alleviate data dependency but remain hindered by computational bottlenecks. To address this limitation, we propose the Neural Variational Quantum Linear Solver (NVQLS), the first hybrid quantum–classical operator learning framework leveraging the Legendre–Galerkin weak formulation. We first resolve the sign ambiguity in VQLS energy minimization, preventing erroneous solution representations. Furthermore, we introduce a neural embedding, a novel encoding scheme that maps varying forcings and PDE coefficients into parameterized quantum circuit representations. These structural innovations improve per-iteration computational cost and memory scaling under suitable assumptions, while achieving superior accuracy compared to classical baselines. Validations on 1D, 2D and 3D parametric PDEs under diverse boundary conditions demonstrate NVQLS's capability to simultaneously process varying inputs, offering a scalable unsupervised approach to quantum-enhanced operator learning.

1 Introduction

Solving partial differential equations (PDEs) lies at the heart of scientific computing, underpinning advances in physics [37, 11] and engineering [50, 49, 21] as well as downstream tasks such as inverse problems [24, 8] and uncertainty quantification [51, 12]. Operator learning [30, 22, 13] aims to approximate the PDE operator, which maps PDE inputs (e.g., forcing, coefficients, boundary conditions) to corresponding PDE solutions. Models such as FNO [25] and DeepONet [30] enable fast inference once trained, but typically require large collections of precomputed solutions. By incorporating PDE information directly into the

†Corresponding authors: Daniel K. Park (dkd.park@yonsei.ac.kr), Youngjoon Hong (hongyj@snu.ac.kr)

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

objective function, unsupervised operator learning [45, 26, 9] bypasses this reliance but still remains computationally bottlenecked as the system size grows. For instance, sampling-based frameworks that utilize automatic differentiation (AD), such as PI-DON [45], suffer from diminished sampling efficiency and slower convergence as the spatial dimension d increases [46, 31, 17, 47, 19]. Meanwhile, although weak-formulation approaches operating in coefficient space (e.g., SCLON [9]) provide high accuracy by leveraging enriched numerical schemes, they struggle with the exponentially growing system size N^d, where N is the number of basis functions per dimension.

Quantum computing offers a distinct computational paradigm in scientific computing, providing a potential route to mitigating classical computational bottlenecks. For example, the Harrow–Hassidim–Lloyd (HHL) algorithm [14] provides a theoretical framework for solving linear systems with runtime polynomial in $\log N$ under strong assumptions. Variational Quantum Linear Solvers (VQLS) [5] were later proposed as heuristic alternatives that avoid costly subroutines by instead employing parameterized quantum circuits and classical optimization. With standard PDE discretization schemes, these quantum linear solvers have emerged as powerful tools for addressing various PDE problems [32, 2, 28, 42, 7]. Parallel to these numerical solvers, learning-based paradigms [33, 23], including Quantum PINNs [41, 38] and QFNO [18] leverage variational quantum circuits as generic function approximators for modeling PDE solutions. However, VQLS and PINN-type solvers are inherently instance-specific, requiring re-optimization for every new PDE condition. While existing quantum operator frameworks, such as Quantum DeepONet [48], can generalize across varying inputs, they predominantly depend on supervised, data-driven training that requires massive datasets of precomputed solutions. These constraints limit the scalability of quantum PDE solvers in high-dimensional and high-resolution scenarios.

To overcome these limitations, we propose Neural Variational Quantum Linear Solver (NVQLS), a hybrid quantum–classical framework for unsupervised operator learning. To the best of our knowledge, this is the first quantum spectral operator learning framework to incorporate the Legendre-Galerkin weak formulation into its loss construction. Our main contributions are summarized as follows:

- We introduce NVQLS, which integrates the Legendre–Galerkin weak formulation with a VQLS-inspired objective to learn solution operators across PDE instances without requiring precomputed solution labels.
- We develop a phase-aware overlap objective that resolves the sign ambiguity of standard VQLS. This is critical for PDE applications, where sign-flipped quantum state representatives can lead to physically and numerically incorrect classical solution fields.
- We introduce a neural embedding scheme that maps varying PDE inputs, including forcing functions and coefficients, to quantum circuit parameters. This enables multi-instance operator learning with shallow variational circuits, instead of retraining a separate quantum model for each PDE instance.
- We provide a complexity analysis showing improved per-iteration computational cost and classical working memory under the stated assumptions over classical unsupervised neural-operator baselines, and validate NVQLS on diverse 1D, 2D and 3D PDEs, including jointly varying forcing–coefficient inputs and a wave-equation benchmark where NVQLS outperforms representative unsupervised classical baselines.

2 Preliminaries and Related Works

2.1 Spectral Operator Learning

We begin by considering a general partial differential equation defined on a bounded d-dimensional domain $\Omega \subset \mathbb{R}^d$, which is characterized by a diffusion coefficient ϵ, a coefficient function $a(\mathrm{x})$, and an external forcing term $f(\mathrm{x})$:

$$\begin{aligned} -\epsilon \operatorname{div}(a\nabla u) + \mathcal{F}(u) &= f, \quad \text{in } \Omega \\ \mathcal{B}(u) &= 0, \quad \text{on } \partial\Omega \end{aligned} \tag{1}$$

where $\mathcal{F}$ denotes a spatial derivative operator and $\mathcal{B}$ represents a boundary operator. This formulation applies generally to second-order elliptic PDEs, including important models such as reaction–diffusion, Helmholtz and convection–diffusion equations. Within the Legendre–Galerkin framework, the governing PDE yields its weak formulation, characterized by the bilinear form $B(\cdot,\cdot)$ and the linear functional $l(\cdot)$:

$$B(u,\phi) := \epsilon \int_\Omega a(\mathrm{x})(\nabla u \cdot \nabla \phi)(\mathrm{x})\, d\mathrm{x} + \int_\Omega [\mathcal{F}(u)(\mathrm{x})]\, \phi(\mathrm{x})\, dx = \int_\Omega f(\mathrm{x})\, \phi(\mathrm{x})\, d\mathrm{x} =: l(\phi). \quad (2)$$

In the domain $I = [-1, 1] \subset \mathbb{R}$, we approximate the PDE solution $u(x)$ by

$$u(x) \approx \hat{u}(x) = \sum_{k=0}^{N-1} \hat{\alpha}_k \phi_k(x), \quad (3)$$

where the basis functions $\phi_k = L_k + a_k L_{k+1} + b_k L_{k+2}$ are defined by a compact linear combination of Legendre polynomials $\{L_k\}$. The basis coefficients a_k and b_k are chosen to strongly enforce the exact boundary conditions, including Dirichlet, Neumann, and mixed types. This basis representation effectively reduces the weak form (2) to a discrete form

$$A\alpha = F, \quad \text{where} \quad A_{kj} = B(\phi_j, \phi_k), \quad F_k := l(\phi_k), \quad (4)$$

where the spectral matrix A and the forward transform F are computed from the explicit calculation of the bilinear and linear forms. More generally, since the d-dimensional basis is defined by the tensor product of 1D basis functions, the spectral coefficient vector α lies in $\mathbb{R}^{N^d}$ and the matrix A is in $\mathbb{R}^{N^d \times N^d}$. A detailed explanation of the weak form and matrix structures is provided in Appendix B.

Inspired by the weak formulation, the Spectral Coefficient Learning via Operator Network (SCLON) [9] learns a coefficient operator that maps a forcing function to spectral coefficients of the corresponding PDE solution. To this end, SCLON trains a neural network to map PDE instances (e.g., forcing functions, initial conditions, or PDE coefficients) to their corresponding spectral coefficients $\hat{\alpha}$ by minimizing an unsupervised loss function based on PDE residuals in the coefficient space. However, as the dimensions of both input features and spectral coefficients scale as $\mathcal{O}(N^d)$, the network size suffers from the curse of dimensionality. This exponential growth with d leads to prohibitive memory bottlenecks and computational costs, particularly for high-dimensional PDEs or high-accuracy simulations requiring a large number of basis modes.

2.2 VQLS

As a heuristic alternative to the HHL algorithm, VQLS [5] was proposed to solve linear systems with parameterized quantum circuits and classical optimization. Given a linear system $Ax = b$ of dimension $N = 2^n$, VQLS aims to produce an n-qubit quantum state $|x\rangle$ such that $A|x\rangle$ is proportional to the target state $|b\rangle$. On the Hilbert space of n qubits, the solution is represented by a parameterized quantum circuit $|x\rangle = V(\theta)|0\rangle$, while the target state is assumed to be prepared as $|b\rangle = U|0\rangle$ for a fixed unitary operator U. The matrix A is expressed as a linear combination of unitaries,

$$A = \sum_l^L c_l A_l, \quad (5)$$

where $A_l \in \mathbb{C}^{2^n \times 2^n}$ are typically Pauli operators, or more generally unitary basis elements, and c_l are complex coefficients. The standard VQLS objective maximizes the fidelity between the normalized state $A|x\rangle / \sqrt{\langle x| A^\dagger A |x\rangle}$ and the target state $|b\rangle$, which can be written as

$$\mathcal{L}_{\mathrm{VQLS}}(\theta) = 1 - \frac{\sum_{l,l'}^L c_{l'}^* c_l \langle 0|V(\theta)^\dagger A_{l'}^\dagger U|0\rangle \langle 0|U^\dagger A_l V(\theta)|0\rangle}{\sum_{l,l'}^L c_{l'}^* c_l \langle 0|V^\dagger(\theta) A_{l'}^\dagger A_l V(\theta)|0\rangle}. \quad (6)$$

Equivalently, this objective can be viewed as minimizing the expectation value of the effective Hamiltonian $H \equiv A^\dagger(I - |b\rangle\langle b|)A$. This energy-based formulation is useful for

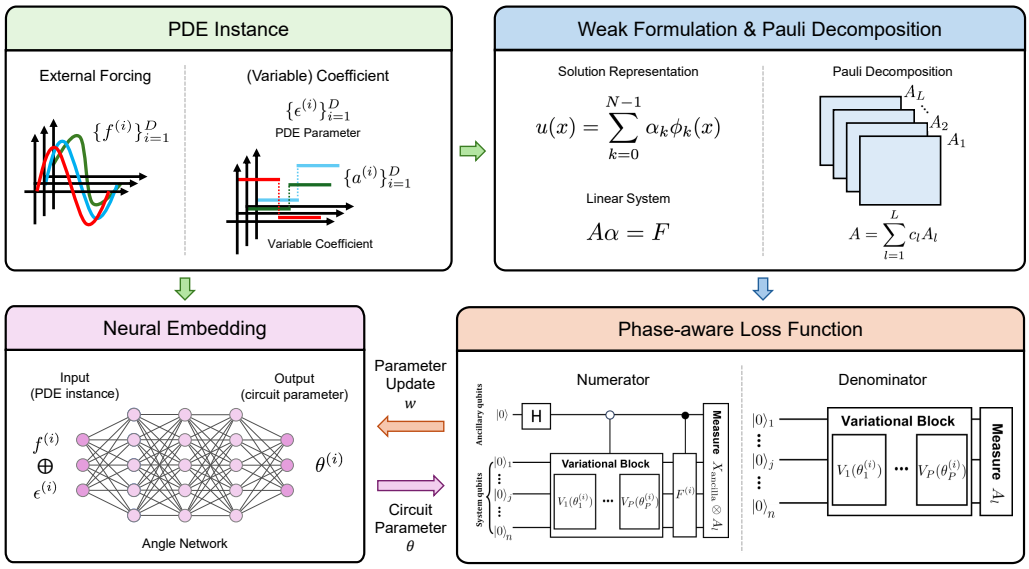


Figure 1: Training procedure of the NVQLS framework. An angle network encodes a batch of PDE instances into quantum circuit parameters θ, while a loss function based on quantum state overlap is utilized to minimize weak-form residuals. Each component of the loss function is evaluated through quantum subroutines with the measurement grouping.

variational optimization, but it also makes the objective insensitive to global sign or phase information of the solution state. This becomes problematic in PDE settings, where the sign of the solution carries physical and numerical meaning. Because of the double-sum structure in Eq. (6), standard VQLS requires $\mathcal{O}(L^2)$ expectation evaluations, often implemented through controlled-unitary operations, which can lead to substantial circuit-depth overhead.

2.3 Related Works

Previous applications of VQLS for solving PDEs have largely been confined to problems yielding highly structured matrices [28, 29, 42, 2]. Other works have focused on improving the practicality of VQLS and broadening its scope of applications [44, 36, 34, 35, 40, 16, 10]. Yet, most VQLS-based approaches are typically restricted to instance-specific training, requiring retraining for every new problem setup. In parallel, quantum approaches for operator learning, such as QFNO [18] and Quantum DeepONet [48], use variational quantum circuits as function approximators for PDE solutions. However, these methods follow a supervised learning paradigm and require large collections of precomputed solution data. In contrast, our work develops a hybrid quantum-classical framework for unsupervised operator learning, where PDE structure is incorporated into the training objective, predicting spectral solution coefficients across problem instances without labeled solutions.

3 Methodology

In this section, we introduce the Neural Variational Quantum Linear Solver (NVQLS), a hybrid variational quantum algorithm designed to approximate the spectral operator that maps a batch of forcing functions $\{f^{(i)}\}_{i=1}^{D}$ to their corresponding solution coefficients. The training objective of NVQLS is to minimize weak-form residuals in the spectral coefficient space. Figure 1 provides an overview of the NVQLS training procedure.

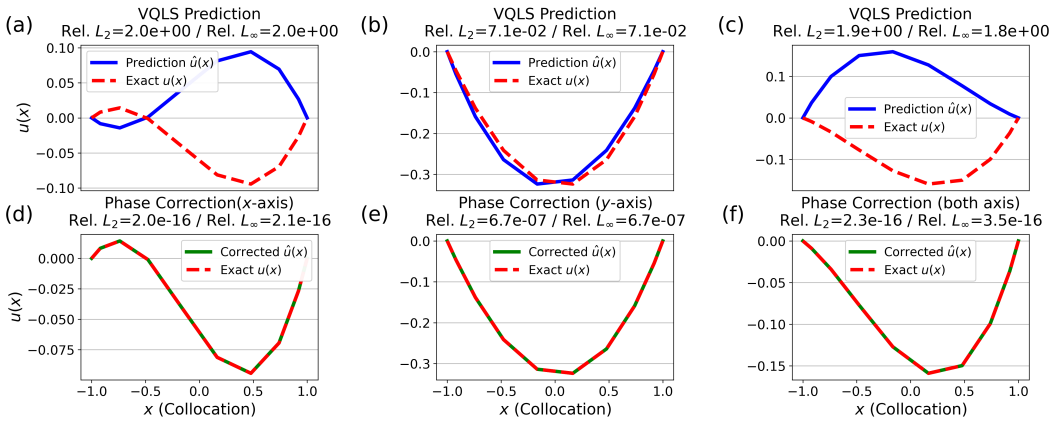


Figure 2: Illustration of the reflection error inherent in standard VQLS and the effect of phase correction. Rows from top to bottom display (a–c) the initial VQLS predictions exhibiting reflection errors, and (d–f) the corresponding solutions after applying the phase correction. Columns from left to right correspond to reflections along the x-axis, the y-axis, and both axes, respectively.

3.1 Phase-Aware Overlap Cost Function

The standard VQLS objective determines the solution state only up to a global phase. Consequently, when applied to real-valued PDE solution coefficients, it can lead to sign or reflection errors in the predicted solution, as illustrated in Fig. 2. To resolve this ambiguity, we introduce a *phase-aware overlap cost* that directly uses the real part of the quantum state overlap to recover the correct solution sign:

$$\mathcal{L}_{\mathrm{NVQLS}}(w) = \frac{1}{D}\sum_{i=1}^{D}\left(1 - \frac{\mathrm{Re}\left(\sum_{l=1}^{L} c_l \left\langle F^{(i)}\right| A_l \left|\hat{\alpha}(f^{(i)};w)\right\rangle\right)}{\sqrt{\sum_{l=1}^{R} d_l \left\langle \hat{\alpha}(f^{(i)};w)\right| A_l \left|\hat{\alpha}(f^{(i)};w)\right\rangle}}\right), \tag{7}$$

with $A = \sum_{l=1}^{L} c_l A_l$ and $A^\dagger A = \sum_{l=1}^{R} d_l A_l$ where $A_l \in \{I, X, Y, Z\}^{\otimes n}$ is an n-qubit Pauli operator. Here, the forward transform of each forcing function is represented by unitaries $|F^{(i)}\rangle = U^{(i)}|0\rangle$. This formulation avoids the double-sum structure of the standard VQLS numerator by using a linear overlap between the transformed forcing vector and the predicted residual vector. Moreover, the Pauli operators in Eq. (7) appear as observables measured at the end of the circuit, rather than as controlled unitaries inside Hadamard-test-type circuits. This structure enables commuting Pauli strings to be grouped and measured together, reducing the number of circuit evaluations required for loss estimation. A detailed derivation of Eq. (7) is provided in Appendix D.

3.2 Neural Embedding for Multiple Instance Learning

To enable generalization across multiple PDE instances, we introduce a *neural embedding* scheme that maps instance-dependent PDE inputs to quantum circuit parameters. Specifically, a classical neural network g with trainable weights w takes each forcing function $f^{(i)}$ as input and outputs the corresponding circuit parameters $\theta^{(i)} = g(f^{(i)}; w)$. The predicted solution coefficients are then encoded in the quantum state

$$|\hat{\alpha}\rangle = V(\theta^{(i)})|0\rangle = V\left(g(f^{(i)};w)\right)|0\rangle. \tag{8}$$

To validate the proposed hybrid structure, Fig. 3 presents an ablation study comparing our quantum-classical neural embedding against two purely quantum baselines: a sequential embedding formulated as $V(\theta, f) = V(\theta)U(f)$ and a data re-uploading scheme, which interleaves data encoding layers throughout the variational circuit to increase expressivity.

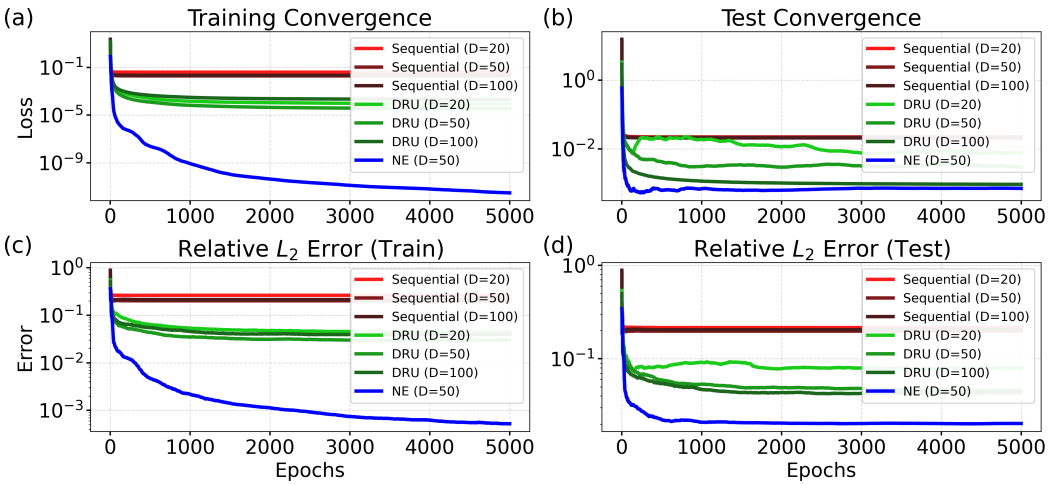


Figure 3: Ablation study on the effect of Neural Embedding (NE). Plots (a–d) show convergence histories for training/test losses and relative L_2 errors. We compare NVQLS with NE (7,780 parameters, depth 10) using $D = 50$ against two quantum baselines (1,200 parameters, depth 100) evaluated across $D \in \{20, 50, 100\}$. Here, D denotes the number of training instances. The results highlight that neural embedding enables effective multi-instance learning with a significantly shallower circuit.

Both baselines use 100 variational layers with approximately 1,200 parameters, but exhibit training difficulties as the number of instances increases ($D = 20, 50,$ and 100). In contrast, NVQLS with neural embedding generalizes to 200 test samples from a training set of only $D = 50$, using 10 layers with 7,780 parameters. As further detailed in Appendix G, neural embedding achieves superior convergence and generalization even with only 2 variational layers and 1,304 parameters.

Furthermore, by adopting this hybrid architecture, the neural network projects the input forcing function onto a compressed parameter space of dimension $\mathcal{O}(n^2) = \mathcal{O}((\log N^d)^2)$. This reduced output dimension lowers the computational cost of both forward and backward passes through the classical network.

3.3 Conditional Per-Iteration Computational Cost and Classical Memory Estimates

We estimate the per-iteration computational cost and peak classical working memory of NVQLS under a heuristic cost model. Quantum computational cost is measured by circuit depth times total shots, and classical computational cost by scalar arithmetic operations.

The estimates rely on three scaling assumptions, detailed together with their scope and limitations in Appendix C. First, we assume that an ansatz whose depth grows linearly with n and that uses $\mathcal{O}(n^2)$ gate parameters is sufficiently expressive for the solution families considered, as empirically supported in Appendix K. Second, we assume that the forward-transformed forcing vectors can be prepared with the shallow embedding scheme described in Appendix H, which yields measurement circuits of depth $\mathcal{O}(\log_2 N^d)$. Finally, based on observed counts for the studied operators, we assume $\mathcal{O}(N^d \log_2 N^d)$ grouped measurement settings per loss evaluation. As shown in Appendix I, Pauli truncation can reduce this count to $\mathcal{O}(\log_2 N^d)$ while keeping the operator-approximation error within the prescribed tolerance. Remark 3.1 summarizes the work and memory estimates, including the further reduction of quantum computational cost under Pauli truncation, and Table 1 compares NVQLS with VQLS and SCLON/PI-DON.

Remark 3.1 (Per-iteration computational cost and classical memory). Let D denote the number of PDE instances used in one shared parameter update. Under the heuristic cost model and scaling assumptions above, the per-iteration computational costs of NVQLS with

Table 1: Conditional computational cost per step for D PDE instances and peak classical working memory for fixed D. For NVQLS and SCLON/PI-DON, the per-step cost covers one parameter update using a batch of D instances; for VQLS, it covers D sequential parameter updates, one for each instance. *Under the stated Pauli-truncation assumptions, the estimated computational cost of NVQLS reduces to $\mathcal{O}\big(D(\log_2 N^d)^4/\epsilon^2 + DN^d(\log_2 N^d)^2\big)$; the memory bound is unchanged.

Model	Per-step computational cost estimate	Classical working memory
NVQLS	$\mathcal{O}\big(DN^d(\log_2 N^d)^4/\epsilon^2 + DN^d(\log_2 N^d)^2\big)^*$	$\mathcal{O}\big(N^d(\log_2 N^d)^2\big)$
VQLS	$\mathcal{O}\big(DN^{2d}(\log_2 N^d)^5/\epsilon^2\big)$	$\mathcal{O}\big((\log_2 N^d)^2\big)$
SCLON / PI-DON	$\mathcal{O}\big(DN^{2d}\big)$	$\mathcal{O}\big(N^{2d}\big)$

parameter-shift gradients satisfy

$$C_{\text{model}} \in O\bigg(\frac{DN^d(\log_2 N^d)^4}{\epsilon^2} + DN^d(\log_2 N^d)^2\bigg), \tag{9}$$

where $0 < \epsilon \leq 1$ is the root-mean-square error tolerance for each measurement-setting estimator. Under Pauli truncation, the per-iteration cost bound reduces to $\mathcal{O}\big(D(\log_2 N^d)^4/\epsilon^2 + DN^d(\log_2 N^d)^2\big)$, which simplifies to $\mathcal{O}\big(DN^d(\log_2 N^d)^2\big)$ for fixed ϵ. Furthermore, for fixed D, the number M_{work} of fixed-precision floating-point variables required in classical working memory satisfies $M_{\text{work}} \in O\big(N^d(\log_2 N^d)^2\big)$. This estimate assumes quantum-hardware execution and excludes inactive training data, read-only operator tables, and additional measurement-processing workspace.

4 Experimental Results

4.1 Spectral Operator Learning with External Forcing Inputs

In this section, we present numerical experiments on learning operators that map forcing functions to the corresponding solutions for various PDEs. The input forcing functions are generated under shallow embedding and general settings, as detailed in Appendix H. Detailed training setups and network architectures are deferred to Appendix E, while problem formulations are provided in Appendix J.

Steady-state Elliptic PDEs. We first evaluated NVQLS on steady-state elliptic PDE benchmarks, which consist of the reaction–diffusion, Helmholtz, and convection–diffusion equations in one and two dimensions, posed under both homogeneous Dirichlet and Neumann boundary conditions. As reported in Table J.1, the mean relative L_2 and L_∞ errors across all 1D and 2D benchmarks remain at low orders of magnitude, ranging from 10^{-4} to 10^{-2}, demonstrating the robust generalization capability of the NVQLS framework across diverse physical systems. We further extended NVQLS to 3D steady-state elliptic PDEs, where it attains relative L_2 and L_∞ errors on the order of 10^{-2} (Table J.2). For the 1D benchmarks, we also compared NVQLS against SCLON, the classical counterpart in spectral coefficient learning. Figure 4 shows the best-performing NVQLS predictions, namely the test samples with the lowest relative L_2 error among 200 test samples, together with the corresponding SCLON predictions. As summarized in Table 2, NVQLS attains lower relative L_2 and L_∞ errors than SCLON across all three 1D benchmarks.

Comparison with Classical Operator Learning. We first compare NVQLS with PI-DON [45], a representative unsupervised operator learning framework that shares the label-free training setting of NVQLS. Here, we consider the time-dependent wave equation with homogeneous Dirichlet boundary conditions, defined in Eq. (43), and provide the training and test error curves of both models in Fig. J.2. Figure 5 presents the solution profiles of both models for the NVQLS test sample with the largest mean absolute error (MAE) among 200 test samples, together with snapshots at $t \in \{0.4, 0.8, 1.2, 1.6\}$ with $t_{\max} = 2$. Even in this worst case, NVQLS attains lower errors than PI-DON, with errors on the order of 10^{-3}, and, as shown in Table J.1, its mean relative L_2 error over all test samples is 2.4×10^{-3}. This result highlights the capability of NVQLS to resolve time-dependent wave dynamics more accurately than the classical PI-DON baseline.

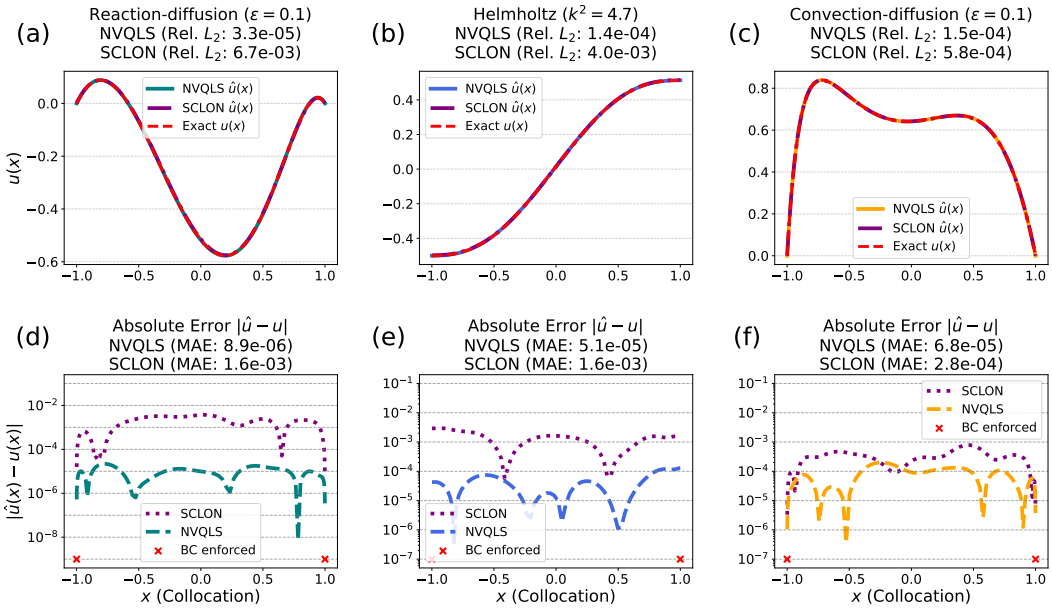


Figure 4: Numerical results for 1D steady-state elliptic PDEs. Top row: (a–c) Predicted solution $\hat{u}$ versus exact solution u. Bottom row: (d–f) Absolute error $|\hat{u} - u|$. Columns from left to right correspond to the reaction–diffusion ($\epsilon = 0.1$, Dirichlet BC), Helmholtz ($k^2 = 4.7$, Neumann BC), and convection–diffusion ($\epsilon = 0.1$, Dirichlet BC) equations, respectively.

Table 2: Comparison of relative L_2 and L_∞ errors between NVQLS and SCLON on one-dimensional steady-state elliptic PDEs, namely the reaction–diffusion (RD) and convection–diffusion (CD) equations under homogeneous Dirichlet boundary conditions, and the Helmholtz (Helm) equation under homogeneous Neumann boundary conditions. Each entry is the mean error over 200 test samples.

	RD ($\epsilon = 0.1$)		Helm ($k^2 = 4.7$)		CD ($\epsilon = 0.1$)	
Model	Rel. L_2	Rel. L_∞	Rel. L_2	Rel. L_∞	Rel. L_2	Rel. L_∞
NVQLS (Ours)	$\mathbf{1.9 \times 10^{-4}}$	$\mathbf{2.0 \times 10^{-4}}$	$\mathbf{2.5 \times 10^{-3}}$	$\mathbf{2.5 \times 10^{-3}}$	$\mathbf{1.0 \times 10^{-3}}$	$\mathbf{1.1 \times 10^{-3}}$
SCLON	4.8×10^{-3}	5.5×10^{-3}	7.3×10^{-3}	7.6×10^{-3}	1.3×10^{-3}	1.4×10^{-3}

To compare NVQLS with supervised operator learning, which relies on a different training paradigm, we examine its data efficiency when labeled numerical solutions are scarce. We adopt the Fourier neural operator (FNO) [25] as a representative supervised framework and train it with $D \in \{50, 100, 200\}$ input-solution pairs, whereas NVQLS is trained without solution labels. For this comparison, both models are trained for 5,000 epochs, and we report the minimum test error attained by each model during training on the same 200 test samples. As shown in Table 3, with its test errors increasing markedly as the number of labeled training samples decreases, FNO requires 100 labeled samples to reach a relative L_2 error comparable to that of NVQLS, which uses no labeled data, and clearly outperforms NVQLS only when trained on 200 samples. However, for the relative L_∞ error, NVQLS attains a lower error (1.40×10^{-2}) than FNO in all settings. These results indicate that NVQLS achieves competitive accuracy without precomputed solution labels, supporting its utility when generating high-fidelity training solutions is costly.

4.2 Extension to Two-dimensional Parametric PDEs for Joint Coefficient–Forcing Operator Learning

We now extend spectral operator learning to two-dimensional PDE problems, where the operator maps jointly varying coefficient and forcing function inputs to their corresponding

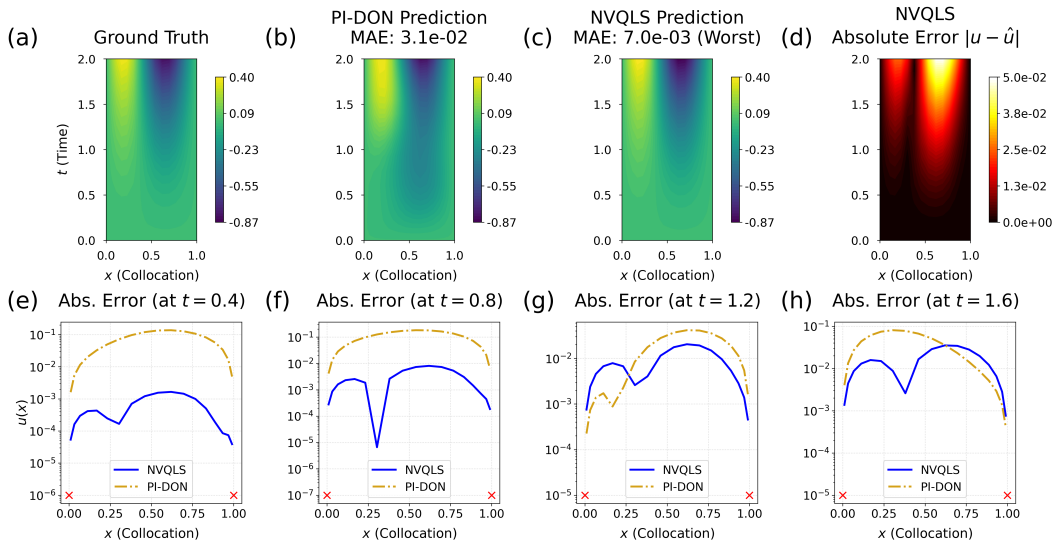


Figure 5: Representative solution profiles for the wave equation. Row 1: Spatio-temporal profiles of (a) the ground truth, (b) the PI-DON prediction, (c) the NVQLS prediction, and (d) the absolute error of the NVQLS prediction. Row 2: (e–h) Snapshots comparing the absolute errors of each model at $t = 0.4, 0.8, 1.2$, and 1.6. Notably, across 200 test samples, even the worst-case prediction of NVQLS demonstrates higher accuracy than the PI-DON prediction.

Table 3: Comparison of NVQLS and supervised FNO on the wave equation with the homogeneous Dirichlet boundary condition under limited labeled-data settings. Here, D denotes the number of paired input–solution training data. All models are trained for 5,000 epochs, and the minimum test errors on 200 unseen samples attained during training are reported.

Metric	NVQLS (Ours, $D = 0$)	FNO ($D = 50$)	FNO ($D = 100$)	FNO ($D = 200$)
Rel. L_2 error	2.26×10^{-2}	3.54×10^{-2}	2.14×10^{-2}	$\mathbf{1.30 \times 10^{-2}}$
Rel. L_∞ error	$\mathbf{1.40 \times 10^{-2}}$	3.98×10^{-2}	2.30×10^{-2}	1.42×10^{-2}

solutions. For training, the input instances are tuples of a forcing function $f^{(i)}$ and PDE coefficients $k^{(i)}$, represented as $\{(k^{(i)}, f^{(i)})\}_{i=1}^{D}$. Each tuple is mapped to quantum circuit parameters $\theta^{(i)} = g(k^{(i)}, f^{(i)}; w)$ via the angle network g. We demonstrate this using the two-dimensional Helmholtz equation, characterized by a squared wavenumber k^2:

$$\begin{aligned} \Delta u(x,y) + k^2 u(x,y) &= f(x,y), \quad \text{in } \Omega \\ \mathcal{B}(u)(x,y) &= 0, \quad \text{on } \partial\Omega \end{aligned} \tag{10}$$

Since the corresponding spectral method matrix is determined by the parameter k and involves tensor products of fixed one-dimensional matrices, we separate the constant and parameter-dependent terms of the matrix (i.e., $B + k^2 C$) in optimization, so that the Pauli decompositions of a fixed set of matrices are computed once rather than for every instance of k. The same separation applies to PDEs with discontinuous variable coefficients, for which example applications are provided in Appendix J.5.

Figure 6 presents three representative test instances of operator learning with joint parameter and forcing inputs for the 2D Helmholtz equation under homogeneous Dirichlet boundary conditions. Each input instance consists of a squared wavenumber k_i^2 sampled from a uniform distribution and a forcing function f_i given by a linear combination of trigonometric functions. Across 200 test samples, NVQLS attains mean relative L_2 and L_∞ errors of

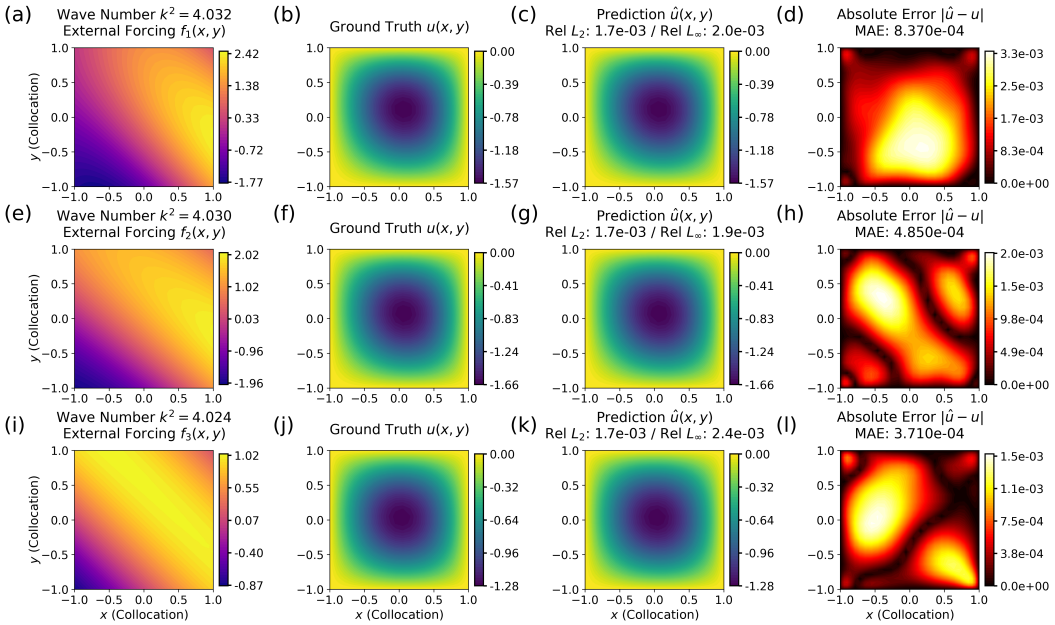


Figure 6: Numerical examples of operator learning with joint parameter and forcing inputs for the two-dimensional Helmholtz equation under the homogeneous Dirichlet boundary conditions. Row 1 (Case 1): (a) Input pairs f_1 and $k^2 = 4.032$ for the angle network, (b) ground truth u_1, (c) NVQLS prediction $\hat{u}_1$, and (d) absolute error $|\hat{u}_1 - u_1|$ on interior nodal points. Row 2 and Row 3 show the corresponding results for Case 2 (f_2 and $k^2 = 4.030$) and Case 3 (f_3 and $k^2 = 4.024$), respectively.

3.0×10^{-3} and 3.2×10^{-3}, respectively (Table J.1), demonstrating its ability to approximate the solution operator across jointly varying parameters and forcing functions.

5 Conclusion

In this work, we proposed a hybrid quantum–classical framework for unsupervised spectral operator learning, designed to solve partial differential equations efficiently. To achieve this, we integrated variational quantum circuits with an angle network architecture for batch training and introduced a phase-aware overlap cost function to resolve the sign ambiguity inherent in VQLS. Our theoretical estimates and empirical analyses suggest that, under the stated assumptions, NVQLS improves scalability in terms of both measurement cost and neural network computational complexity. Furthermore, the model exhibits broad generalization across the tested PDE instances and boundary conditions without retraining. For the PDE families studied here, these results establish NVQLS as a hybrid quantum-classical framework applicable to diverse PDE problems, offering a methodological step toward unsupervised quantum-enhanced operator learning.

Limitations. Despite the inherent scalability of the proposed method, our current experimental scope is limited by the computational cost constraints of classical simulation. Our experiments therefore focus on validating the NVQLS framework and its improved scaling under the stated assumptions for the PDE families studied here, rather than demonstrating a practical quantum advantage. Hardware-inspired simulations further show that gate and readout noise degrade training accuracy, with improved performance at reduced noise rates (Appendix M). We expect the full potential of this framework to be realized as quantum hardware technology matures beyond the current limitations.

Acknowledgments and Disclosure of Funding

This research was supported by the National Research Foundation of Korea (NRF) grant RS-2025-02309510; the Basic Science Research Program funded through NRF by the Ministry of Education (Grant No. 2340012301); the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Korea government (MSIT) (RS-2023-00219980); the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [NO.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)]; the Center for AI and Natural Sciences at Korea Institute for Advanced Study (KIAS); and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (Grant No. RS-2024-00464556).

The authors declare no competing interests.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019.

[2] Mazen Ali and Matthias Kabel. Performance study of variational quantum algorithms for solving the poisson equation on a quantum computer. *Physical Review Applied*, 20(1):014054, 2023.

[3] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2020.

[4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.

[5] Carlos Bravo-Prieto, Ryan LaRose, Marco Cerezo, Yigit Subasi, Lukasz Cincio, and Patrick J Coles. Variational quantum linear solver. *Quantum*, 7:1188, 2023.

[6] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. The Prindle, Weber and Schmidt Series in Mathematics. PWS-Kent Publishing Company, Boston, fourth edition, 1989.

[7] Andrew M. Childs and Jin-Peng Liu. Quantum spectral methods for differential equations. *Communications in Mathematical Physics*, 375(2):1427–1457, 2020.

[8] Sung Woong Cho and Hwijae Son. Physics-informed deep inverse operator networks for solving pde inverse problems, 2025.

[9] Junho Choi, Taehyun Yun, Namjung Kim, and Youngjoon Hong. Spectral operator learning for parametric pdes without data reliance. *Computer Methods in Applied Mechanics and Engineering*, 420:116678, 2024.

[10] Abeynaya Gnanasekaran and Amit Surana. Efficient variational quantum linear solver for structured sparse matrices. In *2024 IEEE International Conference on Quantum Computing and Engineering* (*QCE*), volume 1, pages 199–210. IEEE, 2024.

[11] Linyan Gu, Shanlin Qin, Lei Xu, and Ronglian Chen. Physics-informed neural networks with domain decomposition for the incompressible navier–stokes equations. *Physics of Fluids*, 36(2):021914, 02 2024.

[12] Ling Guo, Hao Wu, Wenwen Zhou, Yan Wang, and Tao Zhou. Ib-uq: Information bottleneck based uncertainty quantification for neural function regression and neural operator learning, 2023.

[13] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc.

[14] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[16] Aruto Hosaka, Koichi Yanagisawa, Shota Koshikawa, Isamu Kudo, Xiafukaiti Alifu, and Tsuyoshi Yoshida. Preconditioning for a variational quantum linear solver. *arXiv preprint arXiv:2312.15657*, 2023.

[17] Zheyuan Hu, Khemraj Shukla, George Em Karniadakis, and Kenji Kawaguchi. Tackling the curse of dimensionality with physics-informed neural networks. *Neural Networks*, 176:106369, 2024.

[18] Nishant Jain, Jonas Landman, Natansh Mathur, and Iordanis Kerenidis. Quantum fourier networks for solving parametric pdes. *Quantum Science and Technology*, 9(3):035026, 2024.

[19] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations, 2019.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

[21] Seid Koric, Asha Viswantah, Diab W. Abueidda, Nahil A. Sobh, and Kamran Khan. Deep learning operator network for plastic deformation with variable loads and material properties. *Engineering with Computers*, 40(2):917–929, 2024.

[22] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, 24(1), January 2023.

[23] Oleksandr Kyriienko, Annie E. Paine, and Vincent E. Elfving. Solving nonlinear differential equations with differentiable quantum circuits. *Phys. Rev. A*, 103:052416, May 2021.

[24] Myeong-Su Lee, Taehyun Yun, Youngjoon Hong, and Namjung Kim. Spectral coefficient learning via operator networks for inverse problems of parametric partial differential equations. *Engineering Applications of Artificial Intelligence*, 166:113665, 2026.

[25] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[26] Zongyi Li, Hongkai Zheng, Nikola B. Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

[27] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[28] Hai-Ling Liu, Yu-Sen Wu, Lin-Chun Wan, Shi-Jie Pan, Su-Juan Qin, Fei Gao, and Qiao-Yan Wen. Variational quantum algorithm for the poisson equation. *Physical Review A*, 104(2):022418, 2021.

[29] YY Liu, Zhen Chen, Chang Shu, Siou Chye Chew, Boo Cheong Khoo, Xiang Zhao, and YD Cui. Application of a variational hybrid quantum-classical algorithm to heat conduction equation and analysis of time complexity. *Physics of Fluids*, 34(11), 2022.

[30] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[31] Sidharth S. Menon and Ameya D. Jagtap. Anant-net: Breaking the curse of dimensionality with scalable and interpretable neural surrogate for high-dimensional pdes. *Computer Methods in Applied Mechanics and Engineering*, 447:118403, 2025.

[32] Mauro E. S. Morales, Lirandë Pira, Philipp Schleich, Kelvin Koor, Pedro C. S. Costa, Dong An, Alán Aspuru-Guzik, Lin Lin, Patrick Rebentrost, and Dominic W. Berry. Quantum linear system solvers: A survey of algorithms and applications, 2025.

[33] Lukas Mouton, Florentin Reiter, Ying Chen, and Patrick Rebentrost. Deep-learning-based quantum algorithms for solving nonlinear partial differential equations. *Phys. Rev. A*, 110:022612, Aug 2024.

[34] Hrushikesh Patil, Yulun Wang, and Predrag S Krstić. Variational quantum linear solver with a dynamic ansatz. *Physical Review A*, 105(1):012423, 2022.

[35] Aidan Pellow-Jarman, Ilya Sinayskiy, Anban Pillay, and Francesco Petruccione. A comparison of various classical optimizers for a variational quantum linear solver. *Quantum Information Processing*, 20(6):202, 2021.

[36] Aidan Pellow-Jarman, Ilya Sinayskiy, Anban Pillay, and Francesco Petruccione. Near term algorithms for linear systems of equations. *Quantum Information Processing*, 22(6):258, 2023.

[37] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[38] Alexandr Sedykh, Maninadh Podapaka, Asel Sagingalieva, Karan Pinto, Markus Pflitsch, and Alexey Melnikov. Hybrid quantum physics-informed neural networks for simulating computational fluid dynamics in complex shapes. *Machine Learning: Science and Technology*, 5(2):025045, may 2024.

[39] Jie Shen, Tao Tang, and Li-Lian Wang. *Spectral Methods: Algorithms, Analysis and Applications*, volume 41 of *Springer Series in Computational Mathematics*. Springer Berlin, Heidelberg, 1 edition, 2011.

[40] Amit Surana and Abeynaya Gnanasekaran. Variational quantum framework for partial differential equation constrained optimization. *ACM Transactions on Quantum Computing*, 2024.

[41] Corey Trahan, Mark Loveland, and Samuel Dent. Quantum physics-informed neural networks. *Entropy*, 26(8), 2024.

[42] Corey Jason Trahan, Mark Loveland, Noah Davis, and Elizabeth Ellison. A variational quantum linear solver application to discrete finite-element methods. *Entropy*, 25(4), 2023.

[43] Lloyd N. Trefethen. *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics, 2000.

[44] Gloria Turati, Alessia Marruzzo, Maurizio Ferrari Dacrema, and Paolo Cremonesi. An empirical analysis on the effectiveness of the variational quantum linear solver. *arXiv preprint arXiv:2409.06339*, 2024.

[45] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40):eabi8605, 2021.

[46] Tao Wang, Bo Zhao, Sicun Gao, and Rose Yu. Understanding the difficulty of solving cauchy problems with pinns. In Alessandro Abate, Mark Cannon, Kostas Margellos, and Antonis Papachristodoulou, editors, *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*, volume 242 of *Proceedings of Machine Learning Research*, pages 453–465. PMLR, 15–17 Jul 2024.

[47] Yifan Wang, Zhongshuo Lin, Yangfei Liao, Haochen Liu, and Hehu Xie. Solving high-dimensional partial differential equations using tensor neural network and a posteriori error estimators. *J. Sci. Comput.*, 101(3), November 2024.

[48] Pengpeng Xiao, Muqing Zheng, Anran Jiao, Xiu Yang, and Lu Lu. Quantum deeponet: Neural operators accelerated by quantum computing. *Quantum*, 9:1761, 2025.

[49] Enrui Zhang, Ming Dao, George Em Karniadakis, and Subra Suresh. Analyses of internal structures and defects in materials using physics-informed neural networks. *Science Advances*, 8(7):eabk0644, 2022.

[50] Guangya Zhang, Tie Xu, Jinli Xu, and Hu Wang. A graph-structured, physics-informed deeponet neural network for complex structural analysis. *Machine Learning and Knowledge Extraction*, 7(4), 2025.

[51] Zongren Zou, Xuhui Meng, Apostolos F. Psaros, and George E. Karniadakis. Neuraluq: A comprehensive library for uncertainty quantification in neural differential equations and operators. *SIAM Review*, 66(1):161–190, 2024.

A Nomenclature

Table A.1: Notations and Descriptions

Notation	Description
n	Number of qubits
N	Number of basis functions
d	Dimension of PDE
$K = N^d = 2^n$	System Size
L	Number of Pauli matrices for Pauli-decomposition
x_n	n-th nodal points on a spatial domain
L_k	Legendre polynomial of degree k
ϕ_k	k-th basis function
$\mathcal{F}$	Differential operator
$\mathcal{B}$	Boundary operator
$u(\cdot)$	Solution of PDE
α_k	k-th Spectral coefficient of the solution $u(\cdot)$
$\hat{u}(\cdot)$	Approximated solution
$\hat{\alpha}_k$	k-th Spectral coefficient of the approximated solution $\hat{u}(\cdot)$
$\hat{\alpha}$	Spectral coefficient vector of the prediction $\hat{u}(\cdot)$
$f(\cdot)$ or $f^{(i)}(\cdot)$	(i-th) forcing function
$\tilde{f}_k$ or $\tilde{f}_k^{(i)}$	k-th forward transformation of the forcing function f or $f^{(i)}$
S	Stiffness matrix
M	Mass matrix
R	Convection matrix
A	Spectral matrix corresponding to PDE problem
c_l	Coefficient of Pauli string of A
d_l	Coefficient of Pauli string of $A^\dagger A$
F or $F^{(i)}$	Forward transform of forcing vector f or $f^{(i)}$
D	Size of data instances
$V(\cdot)$	Parameterized quantum circuit
$\mathcal{L}$	Cost function
θ	Parameters for quantum circuit (output of angle network)
w	Angle network parameters
$w^\star$	Optimal parameters for angle network
g	Angle network
γ	Numerator in our loss $\mathcal{L}_{\text{NVQLS}}$
β	Radicand in the denominator of $\mathcal{L}_{\text{NVQLS}}$
l_V	Number of layers of the parameterized quantum circuit $V(\cdot)$
$\mathcal{C}$	Estimated computational cost
$\mathcal{M}$	Estimated classical working memory
ϵ	Root-mean-square error tolerance per measurement-setting estimator

B Derivation of Weak Formulation

In this section, we present the weak formulation in detail. For a comprehensive overview of the spectral methods employed here, we refer the reader to the standard reference [39]. We begin by examining second-order elliptic partial differential equations on a bounded domain $\Omega \subset \mathbb{R}^n$ subject to boundary conditions, given a PDE parameter $\epsilon > 0$ and an external forcing function f:

$$\begin{aligned} -\epsilon\Delta u + \mathcal{F}(u, \nabla u) &= f, \quad x \in \Omega \subset \mathbb{R}^n, \\ \mathcal{B}(u, \nabla u) &= 0, \qquad x \in \partial\Omega, \end{aligned} \tag{11}$$

where $\mathcal{F}$ denotes a differential operator and $\mathcal{B}$ represents a boundary operator. For clarity, we first consider the one-dimensional reaction–diffusion case where $\mathcal{F}(u) = u$ and $\Omega = (-1, 1)$.

For $k = 0, 1, \ldots, N-1$, the Legendre–Galerkin weak formulation is given by

$$\int_\Omega -\epsilon \Delta u(x)\, \phi_k(x)\, dx + \int_\Omega u(x)\, \phi_k(x)\, dx = \int_\Omega f(x)\, \phi_k(x)\, dx, \tag{12}$$

where the basis functions are compact combinations of Legendre polynomials $\{L_k\}$:

$$\phi_k = L_k + a_k L_{k+1} + b_k L_{k+2}. \tag{13}$$

Here, the exact boundary conditions, including Dirichlet, Neumann, and mixed types, are mathematically enforced by selecting the coefficients a_k and b_k. By representing the solution u as a finite linear combination of the basis functions $\{\phi_k\}_{k=0}^{N-1}$, the weak formulation yields

$$-\epsilon \sum_{k=0}^{N-1} \alpha_k \left(\int_\Omega \phi_j''(x) \phi_k(x)\, dx \right) + \sum_{k=0}^{N-1} \alpha_k \left(\int_\Omega \phi_j(x) \phi_k(x)\, dx \right) = \int_\Omega f(x)\, \phi_j(x)\, dx, \tag{14}$$

for $j = 0, 1, \ldots, N-1$. This produces a linear system $(\epsilon S + M)\alpha = F$ corresponding to the one-dimensional reaction–diffusion equation, where S denotes the stiffness matrix and M represents the mass matrix. Their entries are specified as follows, for $k, j = 0, \ldots, N-1$:

$$S_{kj} = -\int_\Omega \phi_j''(x) \phi_k(x)\, dx, \quad M_{kj} = \int_\Omega \phi_j(x) \phi_k(x)\, dx, \tag{15}$$

with the forward transformed vector given by

$$F = \begin{bmatrix} \tilde{f}_0 & \tilde{f}_1 & \cdots & \tilde{f}_{N-1} \end{bmatrix}^T, \quad \tilde{f}_k = \int_\Omega f(x) \phi_k(x)\, dx. \tag{16}$$

By solving this linear system, one obtains the spectral coefficients

$$\alpha = \begin{bmatrix} \alpha_0 & \alpha_1 & \cdots & \alpha_{N-1} \end{bmatrix}^T. \tag{17}$$

The stiffness matrix S is diagonal and the mass matrix M is symmetric pentadiagonal, whose entries are given by

$$S_{kj} := -\int_I \phi_j'' \phi_k w\, dx = \begin{cases} -(4k+6) b_k, & j = k, \\ 0, & \text{otherwise}, \end{cases} \tag{18}$$

and

$$M_{jk} = M_{kj} := \int_I \phi_j \phi_k w\, dx = \begin{cases} \frac{2}{2k+1} + a_k^2 \frac{2}{2k+3} + b_k^2 \frac{2}{2k+5}, & j = k, \\ a_k \frac{2}{2k+3} + a_{k+1} b_k \frac{2}{2k+5}, & j = k+1, \\ b_k \frac{2}{2k+5}, & j = k+2, \\ 0, & \text{otherwise}. \end{cases} \tag{19}$$

Similarly, for the one-dimensional convection–diffusion equation with Dirichlet boundary conditions, we define the convection matrix R to represent the first spatial derivative u_x:

$$R_{kj} = -R_{jk} = \int_I \phi_j' \phi_k w\, dx = \begin{cases} -2 & k = j+1, \\ 2 & k = j-1, \\ 0 & \text{otherwise}. \end{cases} \tag{20}$$

For the two-dimensional settings, we define the two-dimensional basis as the tensor product of one-dimensional basis functions:

$$\{\phi_k(x) \phi_j(y) \,:\, k, j = 0, \ldots, N-1\}, \tag{21}$$

where $\phi_\cdot(\cdot)$ denotes the one-dimensional basis functions, each formed as a compact combination of Legendre polynomials. Consequently, the predicted solution $\hat{u}(x, y)$ on the two-dimensional domain is expressed as a linear combination of these tensor-product basis functions, with coefficients $\hat{\alpha}_{kj}$:

$$\hat{u}(x, y) = \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} \hat{\alpha}_{kj} \phi_k(x) \phi_j(y). \tag{22}$$

The weak formulation in two dimensions can be expressed as

$$\begin{aligned} &-\epsilon \iint_{\Omega} \Delta u(x,y)\, \phi_k(x)\phi_j(y)\, dx\, dy + \iint_{\Omega} u(x,y)\, \phi_k(x)\phi_j(y)\, dx\, dy \\ &= \iint_{\Omega} f(x,y)\, \phi_k(x)\phi_j(y)\, dx\, dy, \end{aligned} \tag{23}$$

or equivalently,

$$A\alpha = F, \quad \text{where} \quad A = \epsilon(S \otimes M + M \otimes S) + M \otimes M, \tag{24}$$

where α and F are vectors formed by

$$F = (\tilde{f}_{0,0}, \tilde{f}_{1,0}, \ldots, \tilde{f}_{N-1,0}; \tilde{f}_{0,1}, \ldots, \tilde{f}_{N-1,1}; \tilde{f}_{0,N-1}, \ldots, \tilde{f}_{N-1,N-1})^T, \tag{25}$$

and $\otimes$ represents the operation $A \otimes B = (a_{ij}B)_{i,j=0,1,\ldots,N-1}$ (i.e., the Kronecker product).

C Detailed Complexity Analysis

C.1 Cost model and structural assumptions

We restate the assumptions underlying the computational-cost and classical working-memory estimates in Remark 3.1. The computational-cost model quantifies the quantum contribution by circuit depth multiplied by the total number of measurement shots, and the classical contribution by the number of scalar arithmetic operations. Classical working memory is estimated in terms of fixed-precision scalars for fixed D, assuming quantum-hardware execution.

Ansatz and angle network. As the problem dimension N^d grows, we restrict the variational circuit to a shallow ansatz whose depth scales as $\mathcal{O}(\log_2 N^d)$. We assume that this ansatz is sufficiently expressive for the solution families considered. Furthermore, we assume a constant-depth fully connected angle network with input dimension $q \in \mathcal{O}(N^d)$, hidden layers of width $\mathcal{O}(n_\theta)$, and output dimension n_θ, where n_θ denotes the number of circuit parameters. Appendix K provides structural and empirical support for the expressivity of the shallow ansatz and the effectiveness of training with the angle network within the tested regimes.

Efficient state preparation. Under the assumed efficient f-embedding scheme, the state-preparation circuit depth also scales as $\mathcal{O}(\log_2 N^d)$, matching the asymptotic depth of the variational ansatz. Appendix H provides empirical support that the structured forcing functions considered in this work are compatible with shallow state-preparation circuits.

Unitary decomposition scaling. Under commutativity-based grouping, the Pauli decomposition of the linear system yields G distinct measurement settings for one loss evaluation. Based on the empirical observations for the structured PDE operators considered here (Figures C.1 and C.2), we assume $G \in \mathcal{O}(N^d \log_2 N^d)$. For the truncated representation, we additionally assume that the prescribed operator-approximation tolerance can be met using $G_{\text{trunc}} \in \mathcal{O}(\log_2 N^d)$ settings. Appendix I provides empirical support for this favorable measurement scaling and its reduction under accuracy-controlled Pauli truncation.

The scope and limitations of these scaling assumptions are discussed in Section C.4.

C.2 Derivation of computational cost and classical working memory

Computational cost. Grounded in these assumptions, we first count one batch loss/ gradient evaluation and an optimizer update. The number of training iterations depends on initialization and optimization strategies, so the analysis is restricted to per-iteration computational cost.

1. The variational circuit employs a shallow ansatz whose depth is linear in the number of qubits n. For $n = \lceil \log_2 N^d \rceil$ qubits, the circuit contains $\mathcal{O}(n)$ layers with $\mathcal{O}(n)$ parameterized rotation gates per layer. Hence, the total number of circuit parameters is
$$n_\theta \in \mathcal{O}(n^2) = \mathcal{O}\big((\log_2 N^d)^2\big)\,.$$

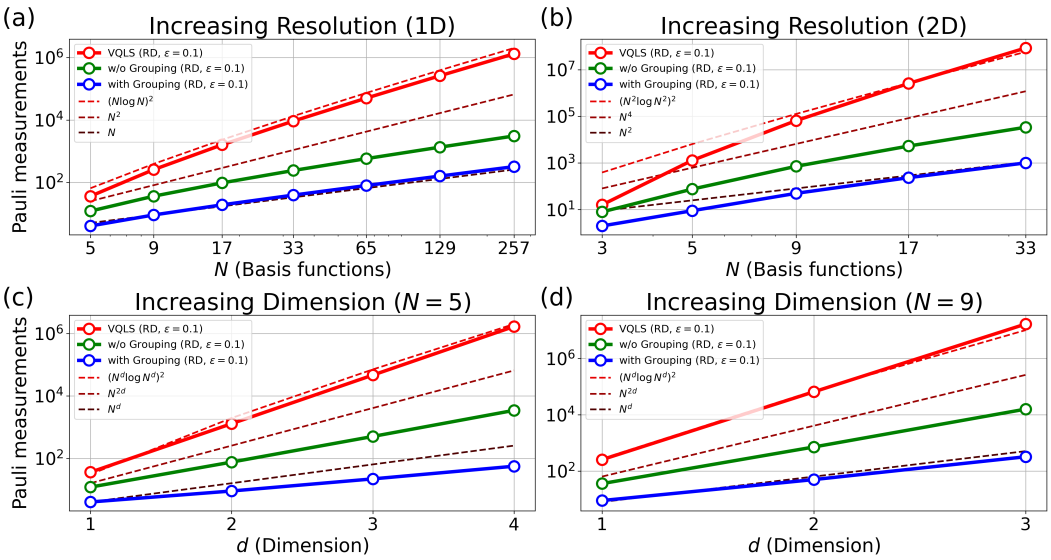


Figure C.1: Empirical analysis of the required Pauli measurements. Comparison between NVQLS with measurement grouping (blue), without grouping (green), and the standard VQLS baseline (red). Scaling with respect to (a)-(b) the number of basis functions N for 1D and 2D cases, and (c)-(d) the dimension d for $N = 5$ and $N = 9$.

2. A single circuit execution includes state preparation and the variational circuit, each of depth $\mathcal{O}(n)$ under the assumptions in Section C.1. With the complete-circuit depth assumption stated in the main text, the total circuit depth is

$$\ell_{\text{circuit}} \in \mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(\log_2 N^d).$$

3. Let Y_g denote an unbiased single-shot estimator of the specified statistic for circuit configuration g. We assume that its variance is uniformly bounded by a constant $C > 0$ across settings, problem instances, and circuit parameters, independently of N^d. For S independent measurement shots, the sample mean satisfies

$$\mathbb{E}\left[\left|\frac{1}{S}\sum_{s=1}^{S} Y_g^{(s)} - \mathbb{E}[Y_g]\right|^2\right] = \frac{\mathrm{Var}(Y_g)}{S} \leq \frac{C}{S}.$$

Therefore, for $0 < \epsilon \leq 1$,

$$S_\epsilon := \max\left\{1, \left\lceil \frac{C}{\epsilon^2} \right\rceil\right\}, \quad S_\epsilon \in O(\epsilon^{-2})$$

independent shots suffice to bound the marginal root-mean-square error by ϵ for each specified measurement estimator.

4. To compute gradients with respect to the circuit parameters θ, the parameter-shift rule (PSR) requires $\mathcal{O}(1)$ shifted evaluations per parameter for the primitive numerator/denominator expectations, yielding $\mathcal{O}(n_\theta)$ evaluations. Their derivatives are combined using the classical chain rule. Since each shifted circuit is evaluated across all G measurement settings, the number of setting evaluations per problem instance is

$$N_{\text{circ}}^{(\text{PSR})} \in \mathcal{O}(G n_\theta) \subseteq \mathcal{O}\big(N^d (\log_2 N^d)^3\big).$$

This count is taken before multiplying by the number of shots. For a batch of D instances, the quantum computational cost is

$$\mathcal{C}_{\text{quantum}} \in \mathcal{O}(DGn_\theta \ell_{\text{circuit}} S_\epsilon) \subseteq \mathcal{O}\left(\frac{DN^d (\log_2 N^d)^4}{\epsilon^2}\right).$$

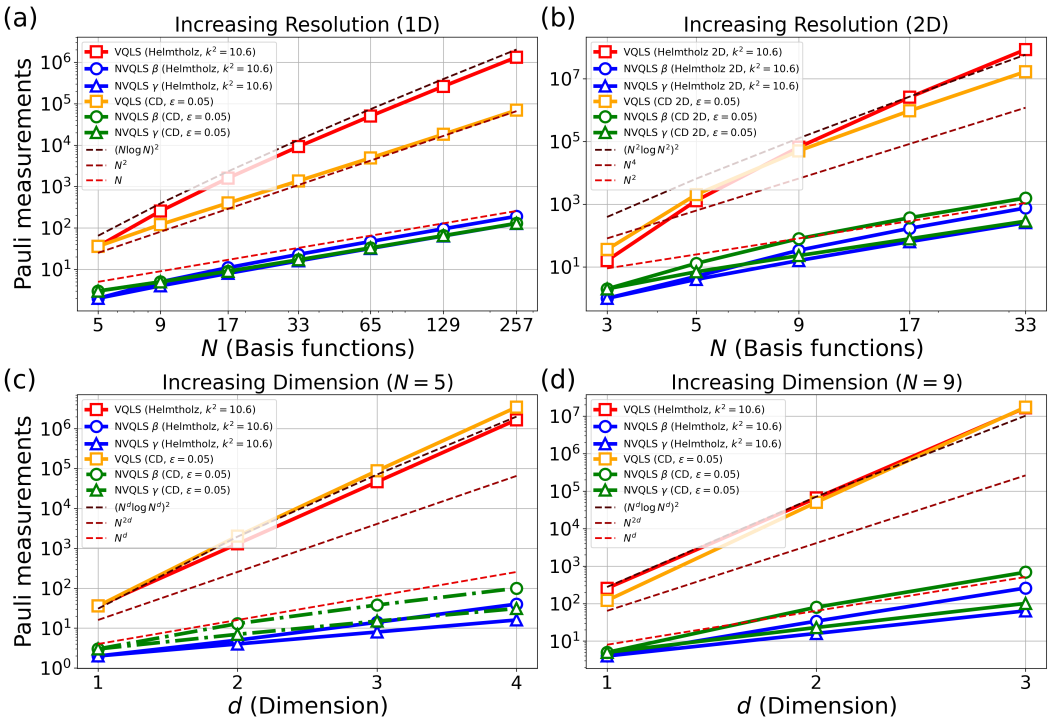


Figure C.2: Empirical analysis of the number of required Pauli measurements in NVQLS framework with grouping of commuting measurement operators compared with VQLS of the Helmholtz equation ($k^2 = 10.6$) and convection diffusion equation ($\epsilon = 0.05$). Top: number of Pauli terms of A for increasing resolution (as a function of the number of basis functions N): (a) one-dimensional case, (b) two-dimensional case. Bottom: number of Pauli terms of A for increasing dimension of PDEs (as a function of the dimension d of PDEs): (c) $N = 5$, (d) $N = 9$.

5. To assess the classical computational cost, recall that the angle network is a constant-depth fully connected network with input dimension $q \in \mathcal{O}(N^d)$, hidden layers of width $\mathcal{O}(n_\theta)$, and output dimension n_θ. The total number of trainable parameters is
$$W \in \mathcal{O}\big(qn_\theta + n_\theta^2\big) \subseteq \mathcal{O}(N^d n^2 + n^4) = \mathcal{O}\big(N^d (\log_2 N^d)^2\big)\,.$$
The forward and backward passes over a batch of size D, combined with the optimizer update, require
$$\mathcal{C}_{\text{classical}} \in \mathcal{O}(DW + W) \subseteq \mathcal{O}\big(DN^d (\log_2 N^d)^2\big)$$
scalar arithmetic operations, using vector–Jacobian products without storing the full network Jacobian.

6. Consequently, the total per-iteration computational cost is obtained by summing the quantum and classical contributions:
$$\begin{aligned}\mathcal{C}_{\text{model}} &= \mathcal{C}_{\text{quantum}} + \mathcal{C}_{\text{classical}} \\ &\in \mathcal{O}(DGn_\theta \ell_{\text{circuit}} S_\epsilon + DW) \\ &\subseteq \mathcal{O}\left(\frac{DN^d (\log_2 N^d)^4}{\epsilon^2} + DN^d (\log_2 N^d)^2\right).\end{aligned} \tag{26}$$

7. Finally, replacing G with the truncated bound $G_{\text{trunc}} \in \mathcal{O}(\log_2 N^d)$ reduces the quantum computational cost to
$$\mathcal{C}_{\text{quantum}}^{(\text{trunc})} \in \mathcal{O}\left(\frac{D(\log_2 N^d)^4}{\epsilon^2}\right).$$

Combining this with the unchanged classical angle-network cost gives the total per-iteration computational cost:

$$\begin{aligned}\mathcal{C}_{\text{model}}^{(\text{trunc})} &= \mathcal{C}_{\text{quantum}}^{(\text{trunc})} + \mathcal{C}_{\text{classical}} \\ &\in \mathcal{O}\left(\frac{D(\log_2 N^d)^4}{\epsilon^2} + DN^d(\log_2 N^d)^2\right).\end{aligned}$$

For fixed D and ϵ, the classical term asymptotically dominates the quantum term in this bound. Thus,

$$\mathcal{C}_{\text{model}}^{(\text{trunc})} \in \mathcal{O}\big(N^d(\log_2 N^d)^2\big).$$

Working memory. For a fixed batch size D, we count classical storage under the conventions in Section C.1.

1. The angle-network parameters, their gradients, and the optimizer states collectively require $\mathcal{O}(W)$ scalars.
2. The active inputs and stored forward activations occupy $\mathcal{O}(q + n_\theta)$ scalars, while the quantum circuit parameters and their corresponding gradient vectors require $\mathcal{O}(n^2)$ scalars.
3. Thus, the total classical working memory required during training is

$$\mathcal{M}_{\text{work}} \in \mathcal{O}(W + q + n_\theta) \subseteq \mathcal{O}\big(N^d(\log_2 N^d)^2\big). \tag{27}$$

Pauli truncation leaves this working-memory bound unchanged under the same storage convention.

C.3 Computational cost and memory estimates for reference baselines

We use the cost model and working-memory conventions of Sections C.1 and C.2. The working-memory exclusions stated in the main text apply to all models.

VQLS. Let L_A denote the number of Pauli terms before grouping. For the operators considered here, $L_A \in \mathcal{O}(N^d \log_2 N^d)$. Direct pairwise evaluation of the global VQLS cost introduced in the main text requires

$$G_{\text{ref}} \in \mathcal{O}(L_A^2) \subseteq \mathcal{O}\big(N^{2d}(\log_2 N^d)^2\big)$$

measurement settings per loss evaluation. For D instances, we count circuit evaluations and classical parameter updates using the bounds on n_θ, ℓ_{circuit}, and S_ϵ from Section C.2. The resulting per-iteration computational cost is

$$\begin{aligned}\mathcal{C}_{\text{VQLS}} &\in \mathcal{O}(Dn_\theta G_{\text{ref}} \ell_{\text{circuit}} S_\epsilon + Dn_\theta) \\ &\subseteq \mathcal{O}\left(\frac{DN^{2d}(\log_2 N^d)^5}{\epsilon^2}\right).\end{aligned}$$

We optimize the D independent instances one at a time, reusing the training workspace after each instance. We accumulate measurement estimates as they are obtained. Precompiled circuits prepare the forcing states, so no N^d-dimensional classical forcing or state vector is retained in active memory. For the active instance, we store the circuit parameters, their gradients, and the optimizer states. Thus, the peak classical working memory is

$$\mathcal{M}_{\text{VQLS}} \in \mathcal{O}(n_\theta) \subseteq \mathcal{O}\big((\log_2 N^d)^2\big).$$

Classical baselines (SCLON and PI-DON). We consider constant-depth dense networks for both baselines. SCLON evaluates the residual by weak-form integration, whereas PI-DON uses pointwise sampling. The dominant fully connected layer in SCLON maps an input vector of dimension proportional to N^d to a vector of N^d spectral coefficients. With the same number of shared nodal points, this layer and the dominant matrix multiplication in PI-DON each involve $\mathcal{O}(N^{2d})$ trainable weights and require $\mathcal{O}(N^{2d})$ arithmetic operations per instance.

Thus, the total number of trainable parameters in each network is $W_{\mathrm{cl}} \in \mathcal{O}(N^{2d})$, and the forward and backward passes require $\mathcal{O}(W_{\mathrm{cl}})$ operations per instance. For a batch of D instances, these passes and one optimizer update require

$$\mathcal{C}_{\mathrm{cl}} \in \mathcal{O}(DW_{\mathrm{cl}} + W_{\mathrm{cl}}) \subseteq \mathcal{O}\big(DN^{2d}\big)$$

scalar arithmetic operations per iteration.

The network parameters, their gradients, and the optimizer states collectively require $\mathcal{O}(W_{\mathrm{cl}})$ fixed-precision scalars. In SCLON, the stored activations occupy $\mathcal{O}(N^d)$ scalars per layer. Since the network depth is constant, storing these activations requires less memory than storing the dense-layer weights. In PI-DON, the dense-layer weights and their associated training states dominate storage during pointwise residual evaluation on the shared nodal points. Thus, the peak classical working memory for either baseline is

$$\mathcal{M}_{\mathrm{cl}} \in \mathcal{O}(W_{\mathrm{cl}}) \subseteq \mathcal{O}\big(N^{2d}\big)\,.$$

These estimates yield the baseline entries in Table 1 and are conditional on the specified reference architectures and evaluation schemes.

C.4 Scope and limitations of the scaling assumptions

The three scaling assumptions above are motivated by the PDE families, operator structures, and architectures considered in this work, and the resulting complexity estimates should be interpreted within this scope. For the variational ansatz, more complex solution families may require a deeper circuit or a larger parameterization, increasing the corresponding terms in the computational cost estimate. Under the remaining assumptions, a polylogarithmic increase in ansatz depth still gives $\mathcal{O}(N^d \operatorname{polylog}(N^d))$ per-iteration computational cost.

For state preparation, the shallow f-embedding assumption is supported by the structured forcing families examined in Appendix H. Less structured forcing inputs may require deeper preparation circuits, in which case the quantum evaluation cost increases with the preparation cost. The stated scaling therefore relies on efficient preparation of the input states.

For the measurement scaling, the empirical counts underlying the estimate are reported in Appendix I. With less favorable sparsity or commuting structure, Pauli grouping may provide a smaller reduction, while truncation may require more retained terms to maintain the desired accuracy. Accordingly, the measurement cost can grow beyond the stated scaling; for a general operator, the Pauli representation may contain up to $\mathcal{O}(N^{2d})$ nonzero coefficients.

D Derivation of Phase-Aware Loss Function

To solve the linear system $A\alpha = F$, our variational quantum framework prepares a parameterized state $|\hat{\alpha}(w)\rangle$. The action of the operator A on this state yields an unnormalized state $A\,|\hat{\alpha}(w)\rangle$. We define the corresponding normalized solution state as:

$$|\psi\rangle = \frac{A\,|\hat{\alpha}(w)\rangle}{\sqrt{\langle\hat{\alpha}(w)|\,A^\dagger A\,|\hat{\alpha}(w)\rangle}}$$

The primary objective is to find the optimal parameters w such that the prepared state $|\psi\rangle$ exactly aligns with the normalized target state $|F\rangle$, which implies $|\psi\rangle = |F\rangle$. In terms of their inner product (overlap), this target alignment is achieved when:

$$\langle F|\psi\rangle = 1$$

Traditional overlap-based loss functions often rely on maximizing the squared magnitude $|\langle F|\psi\rangle|^2$. However, this introduces a critical phase ambiguity, as any state $|\psi\rangle = e^{i\theta}\,|F\rangle$ would still minimize the loss, leading to incorrect solution representations. To resolve this phase ambiguity without introducing the significant measurement overhead associated with computing complex phases, we propose a phase-aware loss function based solely on the real

part of the overlap. Since both $|\psi\rangle$ and $|F\rangle$ are normalized states, the Cauchy-Schwarz inequality restricts their complex overlap to the unit disk:

$$|\langle F|\psi\rangle| \leq 1$$

Expressing the overlap in terms of its real and imaginary components, we have:

$$|\mathrm{Re}\langle F|\psi\rangle|^2 + |\mathrm{Im}\langle F|\psi\rangle|^2 \leq 1$$

From this bounded geometry, it mathematically follows that maximizing the real part strictly constrains the imaginary part. Specifically, if we enforce $\mathrm{Re}\langle F|\psi\rangle = 1$, the inequality mandates that $(\mathrm{Im}\langle F|\psi\rangle)^2 \leq 0$, which is only physically satisfied when $\mathrm{Im}\langle F|\psi\rangle = 0$.

$$\mathrm{Re}\langle F|\psi\rangle = 1 \quad \Longleftrightarrow \quad \langle F|\psi\rangle = 1 \quad (\text{with } \mathrm{Im}\langle F|\psi\rangle = 0)$$

Therefore, explicitly evaluating the imaginary component is mathematically redundant for reaching the exact global minimum. We can define our loss function by simply penalizing the deviation of the real part of the overlap from 1:

$$\mathcal{L}_{\mathrm{NVQLS}}(f; w) = 1 - \mathrm{Re}\langle F|\psi\rangle$$

Substituting the definition of $|\psi\rangle$ into the loss function yields:

$$\mathcal{L}_{\mathrm{NVQLS}}(f; w) = 1 - \frac{\mathrm{Re}(\langle F| A |\hat{\alpha}(w)\rangle)}{\sqrt{\langle\hat{\alpha}(w)| A^\dagger A |\hat{\alpha}(w)\rangle}}$$

Finally, substituting the unitary decompositions of A and $A^\dagger A$, defined as $A = \sum_{l=1}^{L} c_l A_l$ and $A^\dagger A = \sum_{l=1}^{R} d_l A_l$, and averaging over a dataset of size D with target states $|F^{(i)}\rangle$, we arrive at the computable form of the cost function in Eq. (7). We also used an unnormalized cost function $\hat{\mathcal{L}}_{\mathrm{NVQLS}}$ for training stability defined as

$$\hat{\mathcal{L}}_{\mathrm{NVQLS}}(w) = \frac{1}{D}\sum_{i=1}^{D}\left(\mathrm{Re}\left(\sum_{l=1}^{L} c_l \langle F^{(i)}|A_l|\hat{\alpha}\rangle\right) - \sqrt{\sum_{l=1}^{R} d_l \langle\hat{\alpha}|A_l|\hat{\alpha}\rangle}\right)^2 .$$

Here, $\hat{\alpha} = \hat{\alpha}(f^{(i)}; w)$. The rescaled loss is algebraically equivalent to the original cost function because both formulations share the exact same global minimum. By adopting this quadratic form, we eliminate numerical instabilities caused by gradient divergence, leading to significantly more robust and reliable convergence during training. In practice, this loss function often provides better convergence, whereas the fractional form in Eq. (7) can sometimes become unstable during training.

E Training Details

In this section, we provide the details of the model architecture used in our experiments. All simulation training runs are performed using PennyLane [3], supported by the JAX framework [4]. Classical simulations were run on the hardware listed in Table E.1, and the total execution times are reported in Table E.2.

E.1 Quantum Circuit Ansatz

Our model employed the ansatz $V(\theta)$ to prepare $|\alpha\rangle = V(\theta)|0\rangle$ and utilize the classical neural network g to represent θ as $\theta = g(w; F)$ with its parameters w for given forcing vector f. We used two types of ansätze $V(\theta)$: the strongly entangling circuit and the hardware-efficient RY ansatz. Figure E.1 describes each ansatz structure. The strongly entangling layer consists of rotation gates R and CNOT gate. The rotation gate R can represent a general rotation by using three parameters ϕ, θ, ω through two types of single qubit rotation gate R_Y and R_Z, defined as $R \equiv R(\phi, \theta, \omega) = R_Y(\phi)R_Z(\theta)R_Y(\omega)$. This ansatz contains strong entanglements where each qubit is entangled with two other qubits with CNOT gates in each layer, employing the different patterns of entanglements per layer. Due to its characteristics, this possess high representation capacity which can be quantified with high expressibility and high entangling capacity. The number of parameters for this

Table E.1: Specifications of GPU and CPU hardware used for computation

Specification	**Value**
CPU Model Name	Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz
CPU(s)	56
Thread(s) per core	2
Core(s) per socket	28
Socket(s)	1
NUMA node(s)	1
CPU MHz (Max)	3500.00
L1d & L1i cache	1.3 MiB / 896 KiB
L2 cache	35 MiB
L3 cache	42 MiB
RAM	251 GiB
GPU Model name	NVIDIA GeForce RTX 3090
CUDA version	12.4
GPU(s)	4
GPU Architecture	NVIDIA Ampere
Dedicated Memory Size (per GPU)	24 GB
Peak FP32 Performance*	35.6 TFLOPs
Peak Memory Bandwidth*	936 GB/s

* Values based on the official NVIDIA specification sheet for the RTX 3090.

Table E.2: Total execution time for the NVQLS framework across different PDE settings. Each experiment was conducted on a single NVIDIA RTX 3090 GPU. Here, $N_x/N_y/n$ denotes the numbers of basis functions per direction and qubits; for the wave equation, $N_x/N_t/n$ is reported instead, with N_t temporal basis functions.

PDE	$N_x/N_y/n$	**Epochs**	**Total Time (h)**
1D RD	16/-/4	30000	2.0
1D Helmholtz	16/-/4	30000	1.9
1D CD	16/-/4	30000	1.0
1D Wave	8/2/4	50000	2.4
2D RD	8/8/6	200000	202.2
2D Helmholtz	8/8/6	200000	208.1
2D CD	8/8/6	200000	101.1
Joint Helmholtz	8/8/6	100000	24.2

ansatz is $(3 \times n \times l_V)$ which is proportional to the number of qubits n and the number of the ansatz layer l_V.

The Hardware efficient RY ansatz is composed of Hadamard gates H, single qubit rotation gates on y-axis R_Y, and CNOT gates. The Hadamard gates are applied first on each qubit, and rotation gates $R_Y \equiv R_Y(\theta)$ are parameterized with θ. In each layer, the nearest neighbour qubits are entangled by CNOT as well as the first and the last qubits. This ansatz is hardware efficient. Moreover, it maps the initial state $|0\rangle$ to real-valued vector $|\alpha\rangle \in \mathbb{R}^{2^n}$ so we don't need to consider the imaginary part when extracting the vector components from $|\alpha\rangle$. The number of parameters for this ansatz is $(n \times l_V)$, also proportional to both n and l_V.

Table E.3 lists the ansatz configuration used for each benchmark.

E.2 Classical Angle Network

As the angle networks for one-dimensional PDEs, we employ fully connected (FC) neural networks with the Gaussian error linear unit (GeLU) [15] activation functions, except for the

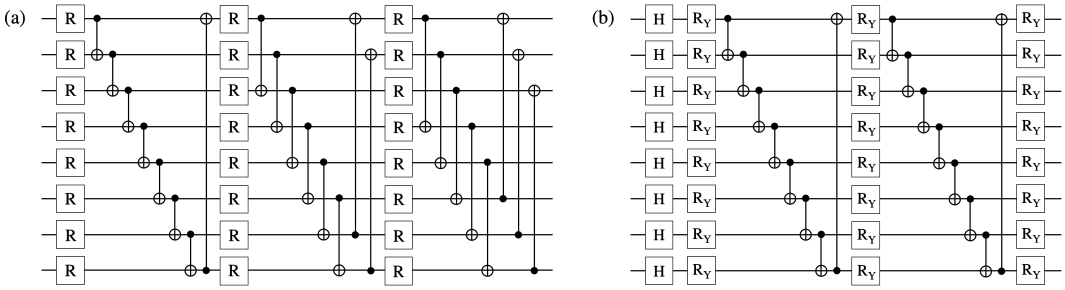


Figure E.1: Ansatz used in our study. (a) Strongly Entangling Layer, (b) Hardware efficient RY ansatz.

Table E.3: Details of the quantum circuit ansatz structure for each PDE benchmark in the main experiment. Here, n denotes the number of qubits and l_V represents the number of layers in the ansatz.

PDE	**Ansatz Structure**	n	l_V
1D RD	StronglyEntanglingLayers	4	13
1D Helmholtz	StronglyEntanglingLayers	4	7
1D CD	StronglyEntanglingLayers	4	7
1D Wave	StronglyEntanglingLayers	4	7
2D RD	StronglyEntanglingLayers	6	7
2D Helmholtz	StronglyEntanglingLayers	6	9
2D CD	StronglyEntanglingLayers	6	7
Joint Helmholtz	StronglyEntanglingLayers	6	9

wave equation, which uses a hybrid network of convolutional neural network (CNN) and FC layers with the Rectified Linear Unit (ReLU) [1] activation functions. For two-dimensional PDEs, including the operator learning with joint inputs, we use a hybrid network of three CNN layers followed by three FC layers with ReLU activation functions. To ensure stable and fast convergence, the L-BFGS optimizer [27] is utilized for the cost function. For the convection–diffusion equation with preconditioner, the adaptive moment estimation (Adam) [20] optimizer is utilized. For hyper-parameter optimization, we adopted a heuristic, progressive approach, starting from a minimal architecture and increasing width or depth only when it improved generalization on validation PDE instances. The architectural details of the angle networks are summarized in Table E.4.

F Performance Metrics

We used numerical solutions based on a spectral method as the ground truth, and both were evaluated on the same grid when computing the errors. Classical spectral convergence results [39, 43, 6] ensure that the approximation error decays extremely fast so that, with a suitable number of basis functions, the numerical error becomes effectively close to the limits of machine precision, typically around $10^{-12} \sim 10^{-14}$.

In our experiments, for the qubit numbers ($n = 4, 5$), the corresponding spectral basis sizes ($N = 16, 32$) are sufficient to reach this accuracy regime, and the numerical solution used for error computation therefore achieves high-accuracy. Any interpolation of the true solution is used only for visualization in the plots and does not affect the underlying error calculations.

To measure the prediction errors, we employ three metrics: batch-wise mean absolute error (MAE), batch-wise relative L_2 error, and batch-wise relative L_∞ error. Given a collection $\{\hat{u}^{(i)}\}_{i=1}^{D}$ of predicted solutions and the corresponding true collection $\{u^{(i)}\}_{i=1}^{D}$, the batch-

Table E.4: Architectural details of the angle network for the primary PDE benchmarks. For 1D cases, the network consists of fully-connected (FC) layers, whereas a hybrid architecture combining convolutional (CNN) and FC layers is employed for 2D problems.

PDE	NN	LAYER	# of params	ACTIVATION
1D RD	FC	4	96,732	GeLU
1D Helmholtz	FC	4	92,052	GeLU
1D CD	FC	4	92,052	GeLU
1D Wave	CNN+FC	3+4	55,612	ReLU
RD 2D	CNN+FC	3+3	182,886	ReLU
Helmholtz 2D	CNN+FC	3+3	89,226	ReLU
CD 2D	CNN+FC	3+3	84,582	ReLU
Joint Helmholtz	CNN+FC	3+3	89,482	ReLU

wise mean absolute error (MAE) is defined by

$$\text{MAE} = \frac{1}{D(N+1)} \sum_{i=1}^{D} \sum_{j=0}^{N} \left| \hat{u}^{(i)}(x_j) - u^{(i)}(x_j) \right|, \tag{28}$$

where D is the number of data instances, N corresponds to the number of basis functions (or spatial points), and $\hat{u}^{(i)}(x_j)$ and $u^{(i)}(x_j)$ are the predicted and true solutions at the collocation point x_j, respectively. The batch-wise relative L_2 error is given by

$$\text{Rel.}L_2 = \frac{1}{D} \sum_{i=1}^{D} \frac{\|\hat{u}^{(i)} - u^{(i)}\|_2}{\|u^{(i)}\|_2} = \frac{1}{D} \sum_{i=1}^{D} \frac{\sqrt{\int |\hat{u}^{(i)} - u^{(i)}|^2 dx}}{\sqrt{\int |u^{(i)}|^2 dx}}, \tag{29}$$

where the integrals are computed using the Legendre–Gauss–Lobatto (LGL) quadrature. Finally, the batch-wise relative L_∞ error is defined by

$$\text{Rel.}L_\infty = \frac{1}{D} \sum_{i=1}^{D} \frac{\|\hat{u}^{(i)} - u^{(i)}\|_\infty}{\|u^{(i)}\|_\infty} = \frac{1}{D} \sum_{i=1}^{D} \frac{\max_{0 \le j \le N} |\hat{u}^{(i)}(x_j) - u^{(i)}(x_j)|}{\max_{0 \le j \le N} |u^{(i)}(x_j)|}. \tag{30}$$

G Ablation Study of Neural Embedding

To evaluate the generalization capability of Neural Embedding (NE), we conducted an ablation study comparing our method against baseline approaches. The first baseline is defined as an extension of VQLS with a sequential data embedding scheme, formulated as:

$$V(\theta, f) = V(\theta)U(f), \tag{31}$$

where the forcing function information is encoded into the quantum circuit using repetitive angle embedding. We also compared our approach with Data Re-uploading (DRU), a recent quantum embedding technique. DRU enhances circuit expressivity by interleaving data encoding $U(f)$ and trainable layers $V(\theta)$, represented as:

$$V(f, \theta) = U(f, \theta)V_{\text{ansatz}}(\theta), \quad \text{where} \quad U(f, \theta) = \prod_{i=1}^{l_{\text{DRU}}} [V_i(\theta_i)U_i(f)]$$

For a rigorous comparison across varying dataset sizes ($D \in \{20, 50, 100\}$), we utilized $l_{\text{DRU}} = 50$ re-uploading layers and an additional 50 Strongly Entangling Layers as the variational ansatz. Both models were optimized using the L-BFGS optimizer to ensure rapid convergence, minimizing the phase-aware overlap cost function (Eq. 7).

In the first experiment, we evaluated the optimal configuration for each method without constraining the number of trainable parameters. Consequently, NVQLS with NE utilized

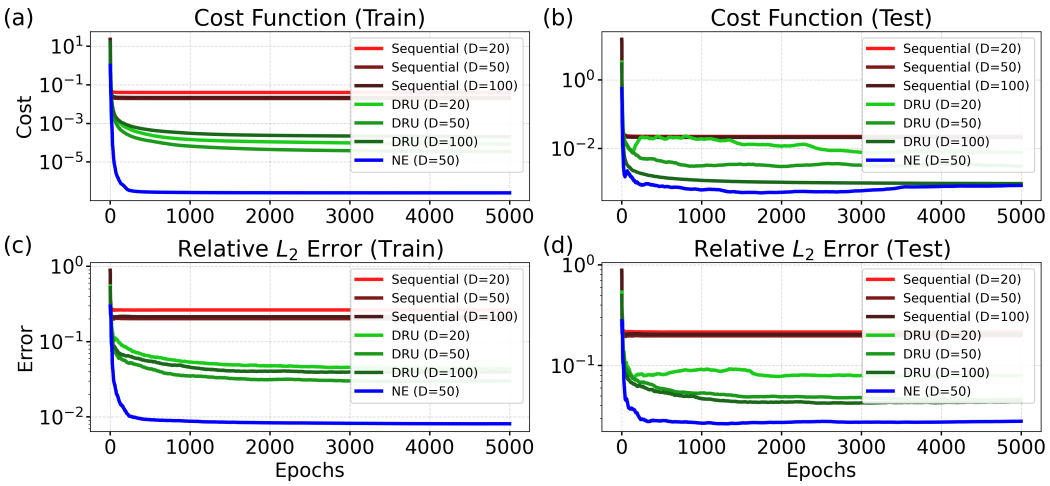


Figure G.1: Efficiency analysis of neural embedding under comparable parameter budgets. The plots illustrate histories over training epochs for (a) training cost, (b) testing cost, (c) relative L_2 training error, and (d) relative L_2 testing error. We compare NVQLS (1,304 parameters, depth 2) against the quantum baselines (1,200 parameters, depth 100). The results demonstrate that neural embedding provides enhanced generalization and performance even with a significantly shallower circuit.

7,780 parameters with an ansatz depth of 10, while the baseline (without NE) employed 1,200 parameters with a depth of 100. As illustrated in Fig. 3, the optimization difficulty for both quantum baselines escalates significantly as the number of instances increases ($D = 20, 50,$ and 100). In contrast, NVQLS with NE effectively handles $D = 50$ training instances and successfully generalizes across 200 test samples, achieving a substantially lower relative error compared to the baselines. These results underscore that simply extending VQLS without a neural network component is insufficient for effective multi-instance learning.

Furthermore, we evaluated the utility of NE by comparing generalization performance under a comparable parameter setting. In this configuration, NVQLS with NE only utilized 1,304 trainable parameters with a drastically reduced circuit depth of 2, whereas the baseline maintained its previous configuration (1,200 parameters/ 100 depth). Figure G.1 presents the corresponding loss curves and error histories. The results demonstrate that NVQLS with NE outperforms the baseline with a comparable parameter budget and a substantially shallower circuit. This underscores the utility of our approach, as it achieves superior performance with significantly reduced model complexity.

H Efficient Circuit Implementation

This appendix provides empirical support for the state preparation assumption used in the complexity analysis. We do not claim that arbitrary amplitude embeddings are shallow or that state preparation is solved in general. Instead, we test whether the structured forcing functions are compatible with shallow circuit preparation. This would allow the input states used by NVQLS to avoid the generic worst-case cost of quantum state preparation.

Shallow Embedding. To examine this point, we consider a shallow real-valued state-preparation family based on single-qubit R_y rotations. We define an angle vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ and prepare the product state

$$|\psi(\boldsymbol{\theta})\rangle = \bigotimes_{i=1}^{n} R_y(\theta_i)|0\rangle \quad \text{with} \quad R_y(\theta) = \exp\left(-\tfrac{i}{2}\theta\, Y\right), \tag{32}$$

Table H.1: Generation details for the input PDE instances in the main experiment. Here, $N_x/N_y/n$ denotes the numbers of basis functions per direction and qubits; for the wave equation, $N_x/N_t/n$ is reported instead, with N_t temporal basis functions.

PDE	**Type**	$N_x/N_y/n$	**Forcing Distribution**	**PDE Coefficient Distribution**
1D RD	Shallow	16/-/4	$\theta \sim U[0, 2\pi)$	0.1
1D Helmholtz	Shallow	16/-/4	$\theta \sim U[0, 2\pi)$	4.7
1D CD	Shallow	16/-/4	$\theta \sim U[0, 2\pi)$	0.1
1D Wave	Specified in Eq. (44)	8/2/4	$\omega \sim U[1, 2)$	–
2D RD	General	8/8/6	$h_i, m_i \sim U[0, 1)$	0.1
2D Helmholtz	General	8/8/6	$h_i, m_i \sim U[0, 1)$	8.9
2D CD	General	8/8/6	$h_i, m_i \sim U[0, 1)$	0.1
Joint Helmholtz	General	8/8/6	$h_i, m_i \sim U[0, 1)$	$U[4, 4.05)$

where Y denotes the Pauli-Y operator. Let $\boldsymbol{a}(\boldsymbol{\theta}) \in \mathbb{R}^K$ denote the real amplitudes of $|\psi(\boldsymbol{\theta})\rangle$ in the computational basis, where $K = 2^n$ is the dimension of the quantum state. We set

$$F_k \;=\; a_k(\boldsymbol{\theta})\,, \qquad k = 0, \ldots, K-1, \tag{33}$$

so that $F \in \mathbb{R}^K$. The dataset is generated classically by sampling $\boldsymbol{\theta}$, computing F, and storing the resulting pairs $(\boldsymbol{\theta}, F)$.

Data Generation for General Setting. To evaluate the generalization performance, we generate a diverse set of input forcing functions using random linear combinations of trigonometric functions. For 1D cases, the forcing functions are defined as:

$$f^{(i)}(x) = h_1^{(i)} \sin(m_1^{(i)} x) + h_2^{(i)} \cos(m_2^{(i)} x), \tag{34}$$

where the parameters $h_j^{(i)}$ and $m_j^{(i)}$ are independently sampled from uniform distributions. These functions, along with their corresponding predicted solutions, are evaluated at nodal points. Similarly, for two-dimensional PDEs, the forcing functions are constructed as:

$$f^{(i)}(x, y) = h_1^{(i)} \sin(m_1^{(i)}(x + y)) + h_2^{(i)} \cos(m_2^{(i)}(x + y)), \quad i = 1, 2, \ldots, D. \tag{35}$$

Table H.1 provides further details regarding the data instance generation, including the specific ranges for the parameters and the PDE-specific coefficients such as the diffusion parameter ϵ and wavenumber k.

Forcing Function Classes and State-Preparation Depth. To characterize the forcing functions compatible with shallow state preparation, we evaluate the trigonometric family in Eq. (34) across increasing frequency ranges. As shown in Table H.2, the corresponding Galerkin forcing states in the range $m_1, m_2 \in [0, 1]$ used in our main experiments are accurately approximated by the single-layer R_y construction in Eq. (32). These results show that the trigonometric forcing states can be accurately prepared with shallow circuits, with additional rotation layers and entangling gates supporting higher-frequency inputs.

Table H.2: State-preparation accuracy across forcing-frequency ranges. Each row averages 800 cases over $K = 2^n \in \{16, 32, 64, 128, 256\}$. Here, e_F is the Euclidean distance between normalized target and prepared forcing states, and e_u is the relative L_2 difference between their Galerkin solutions for the one-dimensional Helmholtz equation with $k^2 = 4.7$ and homogeneous Dirichlet boundary conditions.

Frequency interval	R_y layers	State error e_F	Solution error e_u
$[0, 1]$	1	3.17×10^{-3}	6.52×10^{-4}
$[1, 2]$	2	2.07×10^{-3}	2.60×10^{-4}
$[2, 4]$	3	1.36×10^{-4}	9.96×10^{-6}
$[4, 8]$	4	1.49×10^{-3}	4.68×10^{-4}

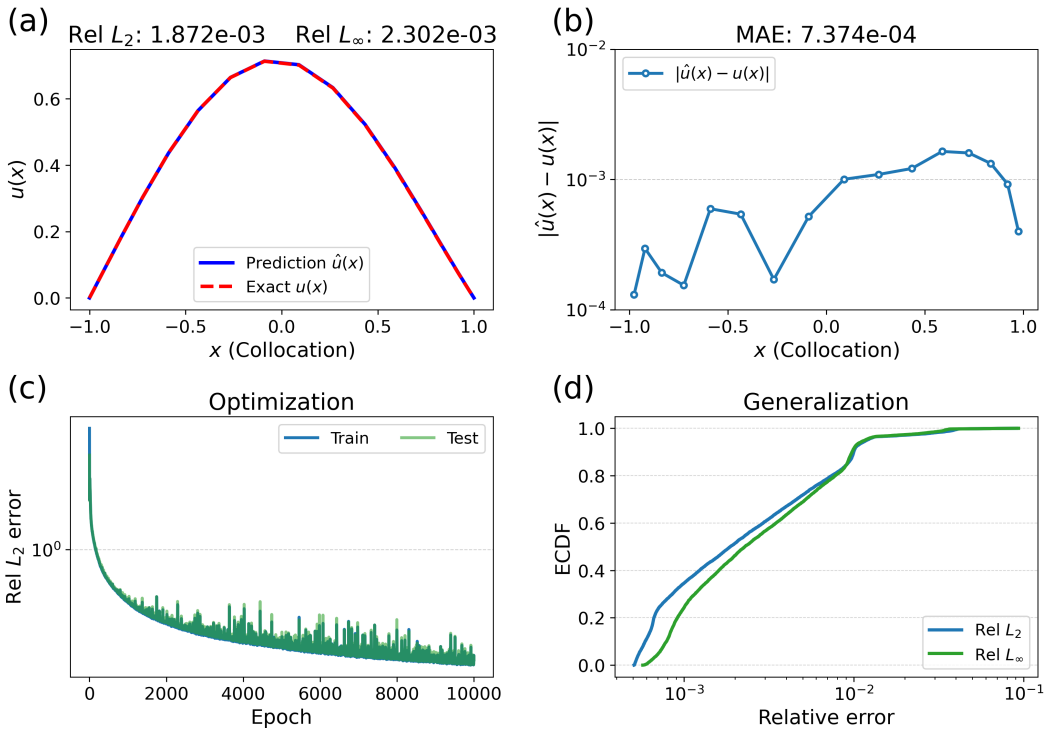


Figure H.1: (a) Representative OOD prediction of the solution $u(x)$ for a forcing function in Eq. 34. (b) Absolute error $|\hat{u}(x) - u(x)|$ for the same example. (c) Train and test relative L_2 errors on the shallow R_y construction. (d) Empirical cumulative distribution functions (ECDFs) of relative L_2 and L_∞ errors on 10,000 OOD forcing functions.

OOD Experiment. We first train NVQLS on transformed forcing vectors generated by the R_y construction. We then evaluate the trained model out of distribution (OOD) on 10,000 forcing functions in Eq. 34. This OOD evaluation tests whether a model trained using input states prepared by shallow circuits generalizes to the structured trigonometric forcing functions. The experiment is conducted on the one-dimensional Helmholtz equation with Dirichlet boundary conditions using a 4-qubit system corresponding to 18 collocation points, including the two boundary points. Each rotation angle is sampled independently from $U[0, 2\pi)$, and the angle network is a fully connected neural network with 21,060 parameters. Figure H.1 summarizes this evaluation. The model achieves low relative L_2 error on the 10,000 OOD forcing functions. This suggests that the structured trigonometric forcing functions used in the main experiments are compatible with shallow circuit preparation, providing empirical support for the state-preparation assumption used in the complexity analysis.

I Truncation Method

To further improve computational efficiency, we approximate the target linear system $A\alpha = F$ by truncating the Pauli decomposition of A, which results in a reduced system $\tilde{A}\tilde{\alpha} = F$. Specifically, instead of using the full decomposition $A = \sum_{l=1}^{L} c_l A_l$, we retain only $L' < L$ Pauli terms and define $\tilde{A} = \sum_{l=1}^{L'} c_l A_l$. Here, $\tilde{\alpha}$ denotes the solution obtained from the truncated system. This truncation significantly reduces the computational cost of the loss evaluation.

We first analyze the effect of truncating the Pauli decomposition of A by retaining terms in descending order of coefficient magnitude, where the truncation threshold is defined with respect to the coefficient size. We consider the one-dimensional Helmholtz equation with

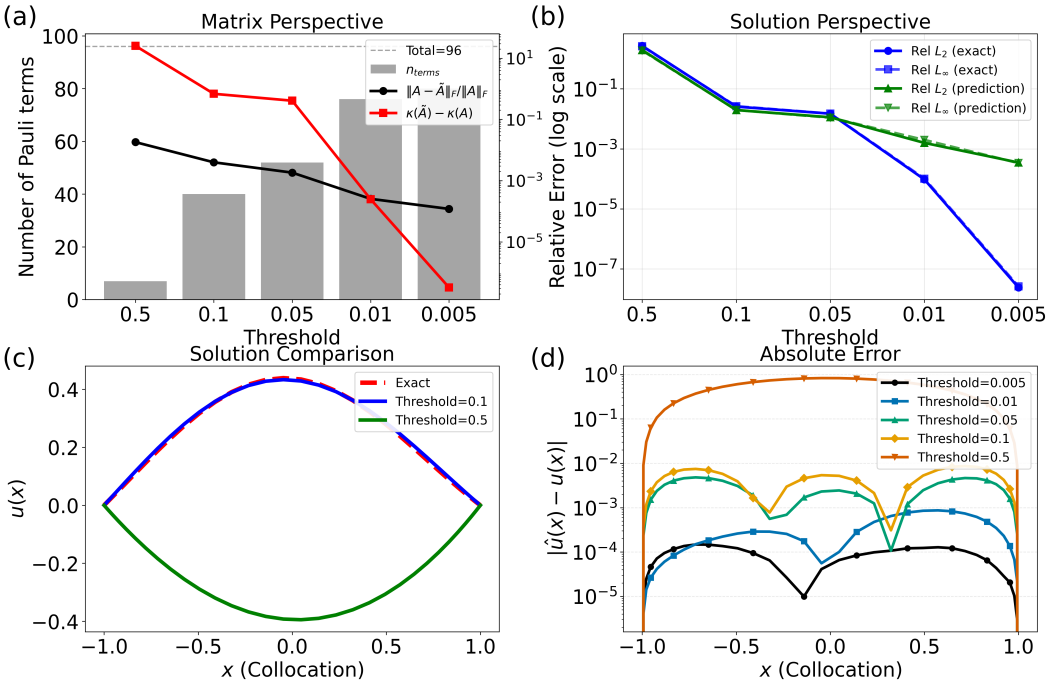


Figure I.1: Impact of the Pauli truncation threshold on the numerical properties of the truncated operator and the resulting solution accuracy for the one-dimensional Helmholtz equation with the homogeneous Dirichlet boundary condition and a squared wavenumber $k^2 = 4$. Top: (a) Relative Frobenius-norm error between A and $\tilde{A}$, condition number of $\tilde{A}$, and number of retained Pauli terms as functions of the truncation threshold. (b) Solution error of the theoretically reconstructed solution and the solution predicted by our model. Bottom: (c) Comparison of the solutions predicted by our model with the exact solution. (d) Absolute error for different truncation thresholds.

Dirichlet boundary conditions and $k^2 = 4$ as a representative example, and summarize the results in Fig. I.1. All learning experiments in this section are conducted using the same training setup as in Fig. J.1.

Figure I.1(a) shows that lowering the truncation threshold reduces the relative approximation error between A and the truncated operator $\tilde{A}$, measured in the Frobenius norm. At the same time, the condition number of $\tilde{A}$ approaches that of the full operator, while the number of retained Pauli terms increases. This indicates improved numerical conditioning of the truncated system with increased operator complexity. Figure I.1(b) compares the theoretically reconstructed solution obtained from $\tilde{\alpha}$ with the exact solution, together with the error of the solution predicted by our model using the truncated operator. For a threshold of 0.05, the predicted solution closely follows the theoretical reconstruction, whereas smaller thresholds lead to noticeable deviations due to increased sensitivity of the learning problem. Nevertheless, at a threshold of 0.01, the prediction error of our model remains on the order of 10^{-3}, which is numerically acceptable for this problem. Figure I.1(c) and (d) further support these observations. Except for the most aggressive truncation case (0.5), the predicted solutions (≤ 0.1) exhibit similar qualitative profiles to the exact solution. The corresponding spatial MAE highlights localized error amplification near the domain boundaries.

Based on the above analysis, we examine truncation thresholds that yield a theoretically induced solution error on the order of 10^{-3}, which we regard as numerically acceptable for the present problem and comparable to the observed prediction accuracy of our model. Figure I.2 summarizes the resulting scaling behavior of the truncated operator. At this accuracy level, the number of retained Pauli terms in $\tilde{A}$ scales linearly with the system size N^d, reducing the original $\mathcal{O}(N^d \log N^d)$ complexity to $\mathcal{O}(N^d)$.

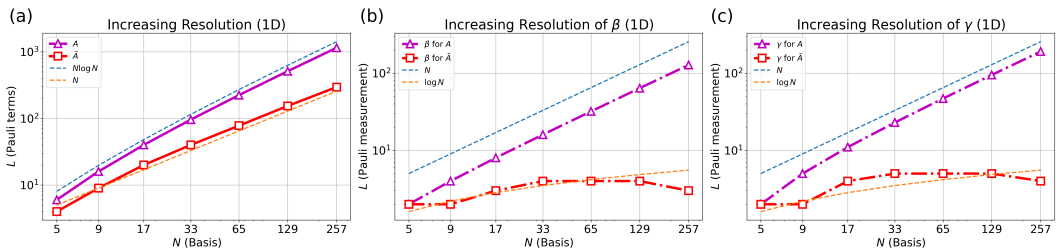


Figure I.2: Scaling behavior of the number of Pauli terms in the truncated operator $\tilde{A}$ and the corresponding measurement cost for loss evaluation in the one-dimensional Helmholtz equation with Dirichlet boundary conditions. (a) Number of retained Pauli terms. (b,c) Number of Pauli measurements required for evaluating β and γ.

Moreover, the number of Pauli measurements required for evaluating the loss components is reduced from $\mathcal{O}(N^d)$ to $\mathcal{O}(\log N^d)$, leading to an overall logarithmic measurement complexity. These results indicate that, when guided by an accuracy-based truncation criterion, substantial computational savings can be achieved while maintaining a solution accuracy that is sufficient for practical learning performance.

J Detailed Numerical Experiments

In this section, we provide the exact forms of the PDEs, their corresponding boundary conditions, and the resulting spectral matrices used in our experiments.

J.1 1D Equations

1D Reaction–diffusion Equation. We start with the one-dimensional reaction–diffusion equation with a diffusion coefficient ϵ under the homogeneous Dirichlet boundary condition:

$$\begin{aligned} -\epsilon u_{xx}(x) + u(x) &= f(x), \quad x \in \Omega := (-1, 1) \\ u(x) &= 0, \quad x \in \partial\Omega. \end{aligned} \tag{36}$$

Here, the Dirichlet conditions are realized via the basis functions

$$\phi_k(x) = L_k(x) - L_{k+2}(x). \tag{37}$$

1D Helmholtz Equation (Neumann BCs). The Helmholtz equation representing a wave propagation problem with a wavenumber k is given by:

$$\begin{aligned} u_{xx}(x) + k^2 u(x) &= f(x), \quad x \in \Omega := (-1, 1) \\ u_x(x) &= 0, \quad x \in \partial\Omega. \end{aligned} \tag{38}$$

Here, the homogeneous Neumann boundary condition is strongly enforced by constructing the basis functions as

$$\phi_j(x) = L_j(x) - \frac{j(j+1)}{(j+2)(j+3)} L_{j+2}(x). \tag{39}$$

1D Convection–diffusion Equation. We now focus on the convection–diffusion equation, characterized by a convection velocity ν and a small diffusion coefficient ϵ:

$$\begin{aligned} -\epsilon u_{xx}(x) + \nu u_x(x) &= f(x), \quad x \in \Omega := (-1, 1) \\ u(x) &= 0, \quad x \in \partial\Omega. \end{aligned} \tag{40}$$

Here, we set $\nu = -1$. Based on the chosen basis functions, the corresponding spectral matrices for each equation are given by:

$$\begin{aligned} A_{\text{RD 1D}} &= \epsilon S + M, \\ A_{\text{Helm 1D}} &= -S + k^2 M, \\ A_{\text{CD 1D}} &= \epsilon S - R, \end{aligned} \tag{41}$$

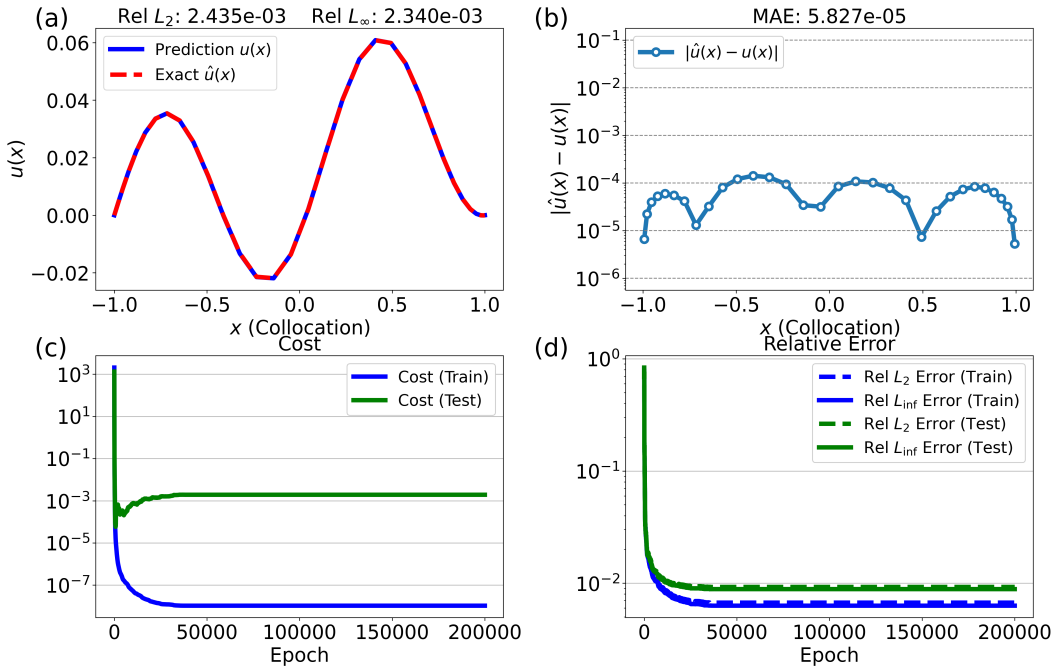


Figure J.1: Numerical results for the one-dimensional Helmholtz equation with the homogeneous Dirichlet boundary condition and a squared wavenumber $k^2 = 29.4$. Top: (a) example of the predicted solution $\hat{u}$ compared to the exact solution u, (b) absolute error between $\hat{u}$ and u. Bottom: (c) batch-wise training and test losses, (d) batch-wise relative L_2 and L_∞ errors over epochs.

where S, M, and R denote the stiffness, mass, and convection matrices, respectively.

1D Helmholtz Equation (Dirichlet BCs). To demonstrate the ability of the proposed method to handle various boundary conditions, we focus on the Helmholtz equation with another boundary condition—Dirichlet BC. The exact form of the equation is given by:

$$\begin{aligned} u_{xx}(x) + k^2 u(x) &= f(x), \quad x \in \Omega := (-1, 1) \\ u(x) &= 0, \quad x \in \partial\Omega. \end{aligned} \tag{42}$$

The corresponding matrix is $A = -S + k^2 M$, which is the same form of the spectral matrix of the Neumann BC. Given a squared wavenumber $k^2 = 29.4$, Fig. J.1 (a) compares the exact solution with the model' s prediction, showing a relative error below 0.3%, which confirms their strong consistency. The corresponding absolute error is depicted in Fig. J.1 (b), yielding an MAE of 5.827×10^{-5}. Figure J.1 (c)–(d) track the training and test dynamics, including both the loss curves and the relative errors. A steady decrease in relative error accompanies the reduction of training cost. Even though the test loss begins to rise early, the continued decrease in the test relative error indicates that further training remains beneficial for achieving better solution accuracy.

1D Wave Equation. We trained NVQLS on the 1D wave equation that is time-dependent and hyperbolic, defined as

$$u_{tt} - u_{xx} = f(x, t), \quad (x, t) \in [0, 1] \times [0, T], \tag{43}$$

with the homogeneous Dirichlet boundary conditions and zero initial conditions (i.e., $u(x, 0) = u_t(x, 0) = 0$). Here, we utilized the external forcing functions, which are parametrized by

$$f(x, t) = \left[1 + \frac{\pi^2 (1+\omega)^2 t^2}{2}\right] \sin(\pi(1+\omega)x) + \left[1 + \frac{\pi^2 (1-\omega)^2 t^2}{2}\right] \sin(\pi(1-\omega)x), \tag{44}$$

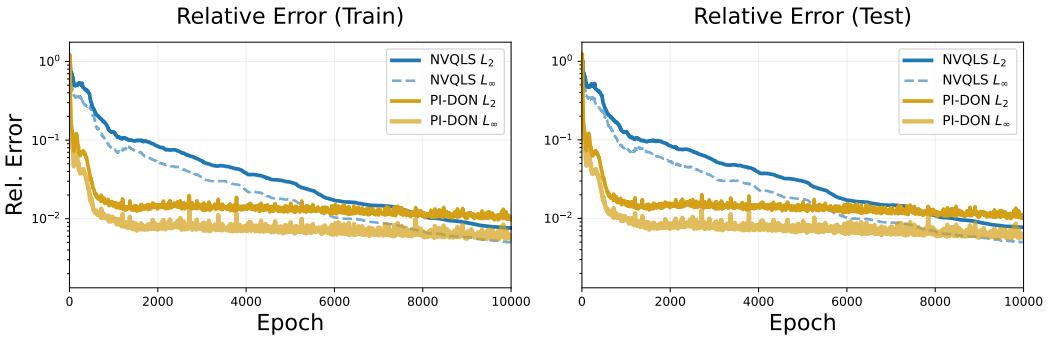


Figure J.2: Visualization of the training and testing error curves for the proposed NVQLS and the PI-DON baseline on the 1D wave equation.

where ω is sampled from a uniform distribution $U[1, 2)$. This setup evaluates the model's ability to learn a family of dynamic wave solutions with varying frequencies. The corresponding spectral matrix for the wave equation is given by:

$$A_{\text{Wave 1D}} = -M_x \otimes S_t + S_x \otimes M_t. \tag{45}$$

J.2 2D Equations

The remaining sections of the numerical experiments focus on results for various two-dimensional problems. For two-dimensional domains, the solution is expressed as a linear combination of basis functions constructed via the tensor product of the corresponding one-dimensional bases in x and y:

$$u(x, y) = \sum_{k,j=0}^{N-1} \alpha_{kj} \phi_k(x) \phi_j(y). \tag{46}$$

Similar to the one-dimensional cases, NVQLS is trained in an unsupervised manner to predict the corresponding solutions from the forcing function inputs in the two-dimensional setting.

2D Reaction–diffusion Equation. We first extend our study to the two-dimensional reaction–diffusion equation with a diffusion coefficient ϵ, subject to homogeneous Dirichlet boundary conditions:

$$\begin{aligned} -\epsilon \Delta u(x, y) + u(x, y) &= f(x, y), \quad (x, y) \in \Omega \\ u(x, y) &= 0, \quad (x, y) \in \partial\,\Omega. \end{aligned} \tag{47}$$

2D Helmholtz Equation. Next, we present the numerical setup for the two-dimensional Helmholtz equation. The specific form of the equation is characterized by the wavenumber k, given by:

$$\begin{aligned} \Delta u(x, y) + k^2 u(x, y) &= f(x, y), \quad (x, y) \in \Omega \\ \mathcal{B}(u) &= 0, \quad (x, y) \in \partial\,\Omega. \end{aligned} \tag{48}$$

Here, $\mathcal{B}(u) = 0$ denotes either the homogeneous Dirichlet or Neumann boundary condition.

2D Convection–diffusion Equation. Finally, we consider the convection–diffusion equation with homogeneous Dirichlet boundary conditions in two dimensions. The exact form is defined with a small diffusion coefficient ϵ and a convection velocity vector $\nu = (\nu_1, \nu_2)^T$:

$$\begin{aligned} -\epsilon \Delta u(x, y) + \nu \cdot \nabla u(x, y) &= f(x, y), \quad (x, y) \in \Omega \\ u(x, y) &= 0, \quad (x, y) \in \partial\,\Omega. \end{aligned} \tag{49}$$

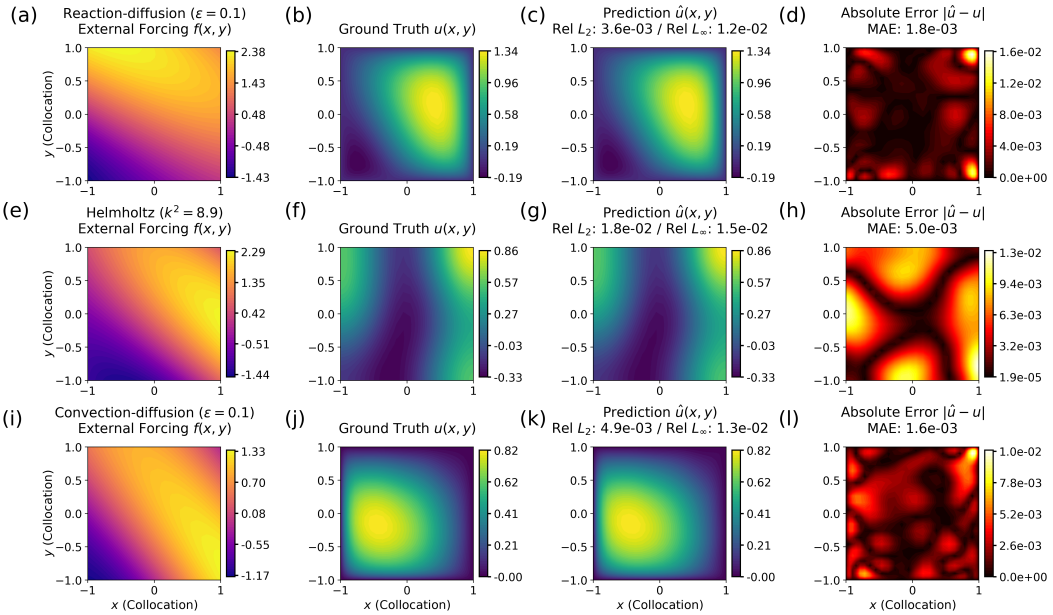


Figure J.3: Numerical results for two-dimensional PDEs. Columns from left to right display the external forcing f, the exact ground truth solution u, the NVQLS prediction $\hat{u}$, and the pointwise absolute error $|\hat{u} - u|$, respectively. Rows from top to bottom correspond to the evaluated physical systems: (a)-(d) the reaction–diffusion equation with $\epsilon = 0.1$ (Dirichlet BC), (e)-(h) the Helmholtz equation with $k^2 = 8.9$ (Neumann BC), and (i)-(l) the convection–diffusion equation with $\epsilon = 0.1$ (Dirichlet BC).

Based on the weak formulation and the tensor product structure, the corresponding spectral matrices for the 2D PDEs are formulated as:

$$\begin{aligned} A_{\text{RD 2D}} &= \epsilon\,(S \otimes M + M \otimes S) + M \otimes M \\ A_{\text{Helm 2D}} &= -S \otimes M - M \otimes S + k^2 M \otimes M \\ A_{\text{CD 2D}} &= \epsilon\,(S \otimes M + M \otimes S) + \nu_1\, R \otimes M + \nu_2\, M \otimes R^T. \end{aligned} \tag{50}$$

Figure J.3 illustrates the representative numerical results for these two-dimensional benchmarks, comparing the exact solutions against the NVQLS predictions. Specifically, panels (a), (b), (c), and (d) present the external forcing, the exact solution, the predicted solution, and the pointwise absolute error, respectively, for the reaction–diffusion equation with $\epsilon = 0.1$. Similarly, panels (e), (f), (g), and (h) display the corresponding results for the Helmholtz equation with $k^2 = 8.9$. Finally, panels (i), (j), (k), and (l) show the results for the convection–diffusion equation with a diffusion parameter $\epsilon = 0.1$ and a convection velocity $\nu = (-1, 0)$. Across all three physical systems, the panels reveal a strong agreement between the predictions and the exact solutions. The relative L_2 errors for all benchmarks are consistently maintained at the order of 10^{-2} or 10^{-3}. Notably, the framework achieves high fidelity with MAE values as low as 1.6×10^{-3} for the convection–diffusion equation, demonstrating the robustness of our framework for 2D spectral operator learning.

J.3 Spectral Operator Learning with Joint Coefficient-Forcing Inputs

This section introduces operator learning with joint parameter and forcing inputs, a framework in which the learned operator maps both a forcing function and a PDE parameter to the corresponding PDE solution. We demonstrate this approach using the Helmholtz equation with a homogeneous Dirichlet boundary condition given in Eq. (42).

To implement operator learning with joint inputs, the input instances are tuples of a forcing function $f^{(i)}$ and a wavenumber $k^{(i)}$, represented as $\{(k^{(i)}, f^{(i)})\}_{i=1}^{D}$ and mapped to angle

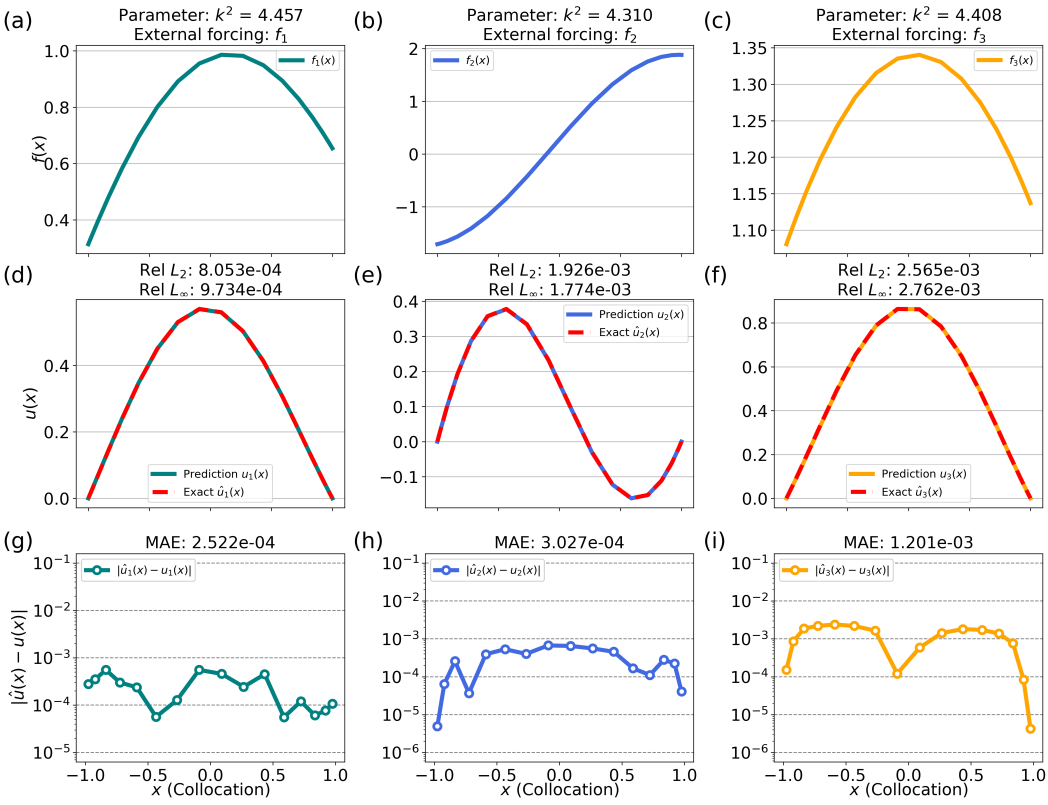


Figure J.4: Numerical examples of operator learning with joint parameter and forcing inputs for the Helmholtz equation under Dirichlet boundary conditions. Top: input pairs for the angle network: (a) case 1: a forcing function f_1 and a squared wavenumber $k^2 = 4.457$, (b) case 2: f_2 and $k^2 = 4.310$, (c) case 3: f_3 and $k^2 = 4.408$. Middle: exact solutions u_i and predicted solutions $\hat{u}_i$ for each case: (d) case 1, (e) case 2, (f) case 3. Bottom: absolute error plots $|\hat{u}_i - u_i|$ on collocation points excluding the boundary points (g) case 1, (h) case 2, (i) case 3.

parameters $\theta^{(i)} = g(k^{(i)}, f^{(i)}; w)$ via the angle network g. Given an input instance (k, f), the spectral method matrix for the Helmholtz equation, $A(k)$, is determined by the parameter k and expressed as a linear combination of the fixed stiffness and mass matrices, S and M. We next perform Pauli decompositions of these fixed matrices where the Pauli terms P_{l_1} and P_{l_2} can represent the same operator:

$$A = -S + k^2 M = \sum_{l_1} c_{l_1} P_{l_1} + k^2 \sum_{l_2} c_{l_2} P_{l_2}. \tag{51}$$

Since the fixed matrices S and M are independent of k, their Pauli decompositions need only be computed once and can be reused for any value of k. Denoting $|\hat{\alpha}\rangle = |\hat{\alpha}(k, f; w)\rangle$, we finally express the numerator of the proposed cost function in Eq. (7) as:

$$\sum_{l_1} \mathrm{Re}\left(c_{l_1} \langle F| P_{l_1} |\hat{\alpha}\rangle\right) + k^2 \sum_{l_2} \mathrm{Re}\left(c_{l_2} \langle F| P_{l_2} |\hat{\alpha}\rangle\right). \tag{52}$$

Similarly, the term inside the square root in the denominator of the cost function is given by the following expression, where Pauli terms P_{l_3}, P_{l_4}, P_{l_5}, and P_{l_6} represent the matrices $S^\dagger S$, $-S^\dagger M$, $-M^\dagger S$, and $M^\dagger M$, respectively, with their corresponding coefficients:

$$\sum_{l_3} c_{l_3} \langle \hat{\alpha}| P_{l_3} |\hat{\alpha}\rangle + k^2 \sum_{l_4} c_{l_4} \langle \hat{\alpha}| P_{l_4} |\hat{\alpha}\rangle + k^2 \sum_{l_5} c_{l_5} \langle \hat{\alpha}| P_{l_5} |\hat{\alpha}\rangle + k^4 \sum_{l_6} c_{l_6} \langle \hat{\alpha}| P_{l_6} |\hat{\alpha}\rangle. \tag{53}$$

Table J.1: Performance of the NVQLS framework on 1D and 2D PDEs, including the reaction–diffusion (RD), Helmholtz (Helm), convection–diffusion (CD), and wave equations. The mean and standard deviation (SD) are computed over 200 test samples. Here, $N_x/N_y/n$ denotes the numbers of basis functions per direction and qubits; for the wave equation, $N_x/N_t/n$ is reported instead, with N_t temporal basis functions.

PDE	**BC**	$N_x/N_y/n$	**Rel. L_2 Error**		**Rel. L_∞ Error**	
			Mean	SD	Mean	SD
1D RD	Dirichlet	16/-/4	1.9×10^{-4}	1.1×10^{-4}	2.0×10^{-4}	1.1×10^{-4}
1D Helmholtz	Neumann	16/-/4	2.5×10^{-3}	6.0×10^{-3}	2.5×10^{-3}	5.5×10^{-3}
1D CD	Dirichlet	16/-/4	1.0×10^{-3}	1.5×10^{-3}	1.1×10^{-3}	1.4×10^{-3}
1D Wave	Dirichlet	8/2/4	2.4×10^{-3}	1.1×10^{-3}	1.8×10^{-3}	6.7×10^{-4}
2D RD	Dirichlet	8/8/6	3.8×10^{-3}	1.1×10^{-3}	1.2×10^{-2}	3.4×10^{-3}
2D Helm	Neumann	8/8/6	2.1×10^{-2}	1.2×10^{-2}	2.3×10^{-2}	1.2×10^{-2}
2D CD	Dirichlet	8/8/6	6.6×10^{-3}	6.4×10^{-3}	1.3×10^{-2}	1.1×10^{-2}
2D Joint Helm	Dirichlet	8/8/6	3.0×10^{-3}	3.8×10^{-3}	3.2×10^{-3}	4.0×10^{-3}

Figure J.4 presents three representative examples of operator learning with joint parameter and forcing inputs for the Helmholtz equation, using Dirichlet boundary conditions. The top row, shown in Fig. J.4(a)–(c), presents three input instances, (k_1, f_1), (k_2, f_2), and (k_3, f_3), where the squared wavenumbers k_i^2 are sampled from the uniform distribution $U[4, 5)$ and the forcing functions are linear combinations of trigonometric functions. The middle and bottom rows (Fig. J.4(d)–(f) and (g)–(i), respectively) present NVQLS predictions versus exact solutions and absolute differences along with MAE values. The predictions exhibit small relative errors (below 0.3%) and MAE values (below 2×10^{-3}), demonstrating the model' s ability to approximate the solution operator across varying parameters.

J.4 Extension to Three-dimensional PDEs

The framework extends naturally to higher dimensions through a tensor-product basis. Using the multi-index $\mathcal{I} = \{0, \ldots, N-1\}^d$ and $\mathbf{k} = (k_1, \ldots, k_d) \in \mathcal{I}$, the d-dimensional solution is approximated as

$$\hat{u}(\mathbf{x}) = \sum_{\mathbf{k} \in \mathcal{I}} \hat{\alpha}_{\mathbf{k}} \Phi_{\mathbf{k}}(\mathbf{x}), \qquad \Phi_{\mathbf{k}} = \bigotimes_{i=1}^{d} \phi_{k_i}, \tag{54}$$

where $\Phi_{\mathbf{k}}$ is the tensor product of the one-dimensional basis functions. Using this formulation, we evaluated NVQLS on three-dimensional steady-state elliptic PDEs, namely the reaction–diffusion (RD), Helmholtz (Helm), and convection–diffusion (CD) equations. Specifically, their spectral matrices are given by

$$\begin{aligned} A_{\text{RD, 3D}} &= \epsilon \left(S \otimes M \otimes M + M \otimes S \otimes M + M \otimes M \otimes S\right) + M \otimes M \otimes M, \\ A_{\text{Helm, 3D}} &= -\left(S \otimes M \otimes M + M \otimes S \otimes M + M \otimes M \otimes S\right) + k^2 M \otimes M \otimes M, \\ A_{\text{CD, 3D}} &= \epsilon \left(S \otimes M \otimes M + M \otimes S \otimes M + M \otimes M \otimes S\right) - R \otimes M \otimes M, \end{aligned} \tag{55}$$

where $\nu = (-1, 0, 0)$. As reported in Table J.2, NVQLS attains relative L_2 and L_∞ errors on the order of 10^{-2}, indicating that the variational circuit can represent multidimensional solutions in the tested settings.

J.5 Applications to PDEs with Discontinuous Variable Coefficients

To illustrate the broader applicability of NVQLS, we include example experiments on PDEs whose coefficients have jump discontinuities, namely the steady-state Darcy flow and the time-dependent parabolic diffusion equation. Such coefficients produce solutions with non-differentiable kinks, which are challenging to capture for neural network-based solvers. Throughout this section, Φ and Φ' denote the matrices of basis functions and their derivatives evaluated at the LGL quadrature nodes, with $\Phi_{jk} = \phi_k(x_j)$ and $\Phi'_{jk} = \phi'_k(x_j)$, and $\boldsymbol{w} = [w_1, \ldots, w_{N_{\text{nodal}}}]^T$ denotes the corresponding quadrature weights.

Table J.2: Performance of NVQLS on three-dimensional steady-state elliptic PDEs. Here, N_x, N_y, and N_z denote the numbers of basis functions in the x-, y-, and z-directions, respectively, and n the required number of qubits.

PDE	BC	$N_x/N_y/N_z/n$	Rel. L_2 Error	Rel. L_∞ Error
3D RD	Dirichlet	4/4/4/6	1.7×10^{-2}	3.8×10^{-2}
3D Helmholtz	Neumann	4/4/4/6	1.0×10^{-2}	1.9×10^{-2}
3D CD	Dirichlet	4/4/4/6	1.6×10^{-2}	2.9×10^{-2}

Table J.3: Performance of NVQLS on PDEs with discontinuous variable coefficients over 200 test samples. Here, N_x and N_t denote the numbers of spatial and temporal basis functions, respectively, and n the required number of qubits.

PDE	BC	$N_x/N_t/n$	Rel. L_2 Error	Rel. L_∞ Error
Darcy flow	Dirichlet	16/-/4	5.0×10^{-4}	8.2×10^{-4}
Parabolic diffusion	Dirichlet / zero initial	8/2/4	8.1×10^{-3}	1.5×10^{-2}

Darcy Flow. We consider the one-dimensional Darcy flow equation with homogeneous Dirichlet boundary conditions,

$$-\frac{d}{dx}\left(a(x)\frac{du}{dx}\right) = f(x), \quad x \in [-1, 1], \tag{56}$$

where $a(x) \geq a_{\min} > 0$. With LGL quadrature, the weak formulation yields the spectral matrix $A = (\Phi')^T \operatorname{diag}(\boldsymbol{w} \circ \boldsymbol{a})\, \Phi'$, where $\circ$ denotes the element-wise product and $\boldsymbol{a} = [a(x_1), \ldots, a(x_{N_{\text{nodal}}})]^T$ contains the values of $a(x)$ at the quadrature nodes. When $a(x)$ jumps from a_1 to a_2 at a fixed interior point, the matrix separates into two fixed components,

$$A(a_1, a_2) = a_1(\Phi')^T \mathbf{W}_L \Phi' + a_2(\Phi')^T \mathbf{W}_R \Phi', \tag{57}$$

where, for a discontinuity located between the nodes x_d and x_{d+1}, $\mathbf{W}_L = \operatorname{diag}(w_1, \ldots, w_d, 0, \ldots, 0)$ and $\mathbf{W}_R = \operatorname{diag}(0, \ldots, 0, w_{d+1}, \ldots, w_{N_{\text{nodal}}})$. As in Section 4.2, the Pauli decompositions of the two fixed matrices are computed once and reused for all instances, while the angle network takes (a_1, a_2) as input.

Parabolic Diffusion. We next consider the time-dependent parabolic diffusion equation

$$c(x,t)\,\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(\kappa(x,t)\,\frac{\partial u}{\partial x}\right) = f(x,t), \quad (x,t) \in [0,1] \times [0,2], \tag{58}$$

with homogeneous Dirichlet boundary conditions and the zero initial condition. Here, the capacity coefficient is prescribed as $c(x,t) = 1 + x^2 t/2$, and the diffusion coefficient $\kappa(x,t)$ jumps from κ_1 to κ_2 in space while remaining constant in time. Using the space–time spectral method as for the wave equation, the spectral matrix is

$$A(\boldsymbol{c}, \boldsymbol{\kappa}) = \Psi^T \operatorname{diag}(\boldsymbol{w} \circ \boldsymbol{c})\, \Psi_t + \Psi_x^T \operatorname{diag}(\boldsymbol{w} \circ \boldsymbol{\kappa})\, \Psi_x, \tag{59}$$

where $\boldsymbol{w}$, $\boldsymbol{c}$, and $\boldsymbol{\kappa}$ contain the quadrature weights and the coefficient values at the space–time quadrature nodes, and $\Psi = \Phi_x \otimes \Phi_t$, $\Psi_x = \Phi_x' \otimes \Phi_t$, and $\Psi_t = \Phi_x \otimes \Phi_t'$, with the subscripts x and t indicating the spatial and temporal bases, respectively. Since $c(x,t)$ is fixed, the first term is a fixed matrix and the second term separates into components weighted by κ_1 and κ_2, so the same precomputed Pauli decompositions apply with (κ_1, κ_2) as the input to the angle network.

Numerical Results. Table J.3 reports the mean relative L_2 and L_∞ errors over 200 test samples. For the Darcy flow, NVQLS captures the derivative discontinuity induced by the coefficient jump with relative L_2 and L_∞ errors of 5.0×10^{-4} and 8.2×10^{-4}, respectively. For the parabolic diffusion equation, the corresponding errors are 8.1×10^{-3} and 1.5×10^{-2}, with larger errors concentrated in instances with severe jumps in the diffusion coefficient. A detailed analysis of variable-coefficient problems is left for future work.

Table K.1: L_2 state approximation errors across system sizes and variational depths. Here, K is the state dimension, $n = \log_2 K$ is the number of qubits, and l_V is the number of variational layers. Circuit parameters are optimized for individual normalized target states using Eq. (61).

K	$n = \log_2 K$	$l_V = 2n$	$l_V = 4n$	$l_V = 8n$
16	4	8.58×10^{-8}	5.54×10^{-8}	4.07×10^{-8}
32	5	1.03×10^{-7}	7.42×10^{-8}	4.49×10^{-8}
64	6	1.24×10^{-7}	9.31×10^{-8}	6.09×10^{-8}
128	7	3.44×10^{-4}	1.07×10^{-7}	7.18×10^{-8}
256	8	1.48×10^{-1}	1.26×10^{-7}	9.48×10^{-8}

Table K.2: Relative L_2 errors of NVQLS for different angle network depths and approximate trainable classical parameter counts. All configurations use five qubits and seven ansatz layers. Bold entries indicate the lowest error within each row.

Hidden layers	100k	150k	200k	250k	300k
2 (shallow)	4.55×10^{-3}	4.53×10^{-3}	3.73×10^{-3}	4.57×10^{-3}	$\mathbf{3.37 \times 10^{-3}}$
3 (base)	2.56×10^{-3}	2.23×10^{-3}	2.25×10^{-3}	$\mathbf{2.18 \times 10^{-3}}$	2.53×10^{-3}
4 (deep)	2.14×10^{-3}	1.90×10^{-3}	1.69×10^{-3}	1.87×10^{-3}	$\mathbf{1.52 \times 10^{-3}}$

K Architectural Analysis of the Ansatz and Angle Network

The performance of NVQLS depends jointly on the variational ansatz, which represents the spectral solution state, and the classical angle network, which maps PDE inputs to circuit parameters. We first examine the representational capacity of the ansatz and then evaluate sensitivity to the depth and parameter count of the angle network.

Structural and Empirical Support for the Shallow Ansatz. Our complexity analysis assumes that the target solution states can be approximated by an ansatz whose depth scales as $O(\log K)$, where K denotes the dimension of the spectral coefficient space. For $n = \log_2 K$ qubits and l_V variational layers, an architecture with $\mathcal{O}(n)$ parameterized rotations per layer has

$$n_\theta = \mathcal{O}(n\, l_V) = \mathcal{O}\big((\log K)^2\big) \qquad \text{when } l_V = \mathcal{O}(\log K). \tag{60}$$

To characterize the structure of this parameterization, consider the state $|x_\theta\rangle = U(\theta)|0\rangle$. Each of its K computational-basis amplitudes is a multivariate function of the circuit parameters. For a circuit comprising n_θ independently parameterized two-eigenvalue rotations interleaved with fixed gates, expanding the two spectral components of each rotation yields at most 2^{n_θ} candidate Fourier terms per amplitude. The coefficients of these terms are constrained by the circuit architecture, so this term-count bound alone does not guarantee approximation of arbitrary states. The adequacy of a shallow ansatz therefore depends on both its architecture and the target solution family.

We assess this adequacy through a state-approximation experiment. For each normalized target state $|x_i\rangle$, we optimize the circuit parameters by minimizing the phase-aware objective

$$\mathcal{L}_{\text{rep}}(\theta; x_i) = 1 - \text{Re}\left[\langle x_i|U(\theta)|0\rangle\right]. \tag{61}$$

This experiment evaluates the ability of the ansatz to approximate individual target states independently of the angle network.

Table K.1 reports the results for $K \in \{16, 32, 64, 128, 256\}$ and $l_V \in \{2n, 4n, 8n\}$. With $l_V = 2n$, the approximation error increases at larger system sizes, reaching 3.44×10^{-4} for $K = 128$ and 1.48×10^{-1} for $K = 256$. Increasing the depth to $l_V = 4n$ or $8n$ reduces the reported errors to approximately 10^{-7} or below across all tested sizes. These results support the adequacy of logarithmic-depth ansatzes for the tested target states, while showing that the depth prefactor is important for achieving high accuracy.

Table L.1: Performance of NVQLS on the one-dimensional Helmholtz equation as the relative total variation (RTV) of the forcing increases, using $N = 32$ basis functions. Ratios of RTV and the errors to their corresponding baseline values for RY mode 2 are indicated by parenthetical multipliers.

Forcing setting	RTV	Rel. L_2 error	Rel. L_∞ error
RY mode 2	0.0525 ($\times$1.00)	3.30×10^{-4} ($\times$1.00)	3.54×10^{-4} ($\times$1.00)
RY mode 3	0.0742 ($\times$1.41)	4.18×10^{-4} ($\times$1.27)	4.69×10^{-4} ($\times$1.32)
RY mode 4	0.1085 ($\times$2.07)	6.37×10^{-4} ($\times$1.93)	7.24×10^{-4} ($\times$2.05)

Table L.2: Performance of NVQLS on the one-dimensional Helmholtz equation as the wavenumber k increases, using $N = 16$ basis functions. The first column reports the squared wavenumber k^2. Ratios of the errors to the corresponding baseline values at $k^2 = 15.3$ are indicated by parenthetical multipliers.

$k^2 \uparrow$	cond(A)	Condition ratio	Rel. L_2 error	Rel. L_∞ error
15.3	14.2	$\times$1.00	4.71×10^{-3} ($\times$1.00)	4.55×10^{-3} ($\times$1.00)
49.5	25.6	$\times$1.81	5.71×10^{-3} ($\times$1.21)	5.24×10^{-3} ($\times$1.15)
103.8	56.4	$\times$3.98	5.93×10^{-3} ($\times$1.26)	5.63×10^{-3} ($\times$1.24)
138.4	76.5	$\times$5.39	6.82×10^{-3} ($\times$1.45)	6.72×10^{-3} ($\times$1.48)
177.8	97.2	$\times$6.86	7.64×10^{-3} ($\times$1.62)	7.55×10^{-3} ($\times$1.66)

Sensitivity to the Trainable Parameter Size. We next investigate the sensitivity of NVQLS to the depth and parameter count of the classical angle network while keeping the variational ansatz fixed. For a five-qubit system, the network maps a 32-dimensional transformed forcing vector to 105 quantum circuit parameters corresponding to seven ansatz layers. We compare architectures with two, three, and four hidden layers, adjusting their widths to obtain approximately 100k, 150k, 200k, 250k, and 300k trainable classical parameters.

Table K.2 shows that all tested configurations achieve relative prediction errors on the order of 10^{-3}, ranging from 1.52×10^{-3} to 4.57×10^{-3}. At comparable parameter counts, deeper networks consistently achieve lower errors. Increasing the parameter count, however, does not produce monotonic improvements within each depth setting. The lowest error, 1.52×10^{-3}, is obtained with four hidden layers and approximately 300k parameters. These results indicate that NVQLS maintains comparable error magnitudes across the tested angle network architectures, while benefiting from increased network depth.

L Performance of NVQLS in Challenging Regimes

To assess the convergence stability and solution accuracy of NVQLS in challenging regimes, we consider two sources of difficulty: increased variation in the input forcing functions and increased sensitivity of the discretized PDE system. We quantify forcing variation using relative total variation (RTV) and system sensitivity using the condition number of the spectral matrix.

Highly Oscillating Forcing. We consider the one-dimensional Helmholtz equation with $N = 32$ basis functions. The forcing functions are generated using the shallow embedding defined in Eq. (32), with higher RY modes introducing higher-frequency components. As shown in Table L.1, increasing RTV from 0.0525 to 0.1085 raises the relative L_2 error from 3.30×10^{-4} to 6.37×10^{-4}, corresponding to a 1.93-fold increase. Both relative L_2 and L_∞ errors remain below 10^{-3} across the three settings, indicating that NVQLS maintains high accuracy as forcing variability increases within the tested family.

Ill-conditioned Loss Landscape. To assess how NVQLS performance scales with problem parameters that may affect optimization difficulty, we vary the wavenumber in the Helmholtz equation and decrease the diffusion coefficient in the convection–diffusion equation.

Table L.3: Performance of NVQLS on the one-dimensional convection–diffusion equation as the diffusion coefficient ϵ decreases, using $N = 64$ basis functions. The eigenvalue imaginary-to-real ratio $\rho(A)$ is defined in Eq. (62). Ratios of the errors to their corresponding baseline values at $\epsilon = 0.1$ are indicated by parenthetical multipliers.

$\epsilon \downarrow$	cond(A)	$\rho(A)$	Rel. L_2 error	Rel. L_∞ error
0.1	21.30	2.0362	8.97×10^{-4} ($\times$1.00)	1.19×10^{-3} ($\times$1.00)
0.05	15.94	3.6181	1.69×10^{-3} ($\times$1.88)	2.21×10^{-3} ($\times$1.86)
0.01	13.13	12.0531	1.67×10^{-2} ($\times$18.62)	1.99×10^{-2} ($\times$16.72)
0.005	16.95	19.6892	5.68×10^{-3} ($\times$6.33)	8.38×10^{-3} ($\times$7.04)

We first evaluate the robustness of NVQLS on the one-dimensional Helmholtz equation across increasing wavenumbers k. Larger wavenumbers correspond to higher spatial frequencies and, over the tested range, increase the condition number of the spectral matrix. This experiment therefore examines how NVQLS's convergence and solution accuracy scale with worsening matrix conditioning.

We use $N = 16$ basis functions and uniformly sample trigonometric forcings of the form given in Eq. (34), training each model for 30,000 epochs. As shown in Table L.2, increasing k^2 from 15.3 to 177.8 raises cond(A) from 14.2 to 97.2, approximately a 6.9-fold increase. Over the same sweep, the relative L_2 error increases only from 4.71×10^{-3} to 7.64×10^{-3}, a factor of 1.62, while the relative L_∞ error remains comparable, reaching 7.55×10^{-3} at the largest wavenumber. These results demonstrate favorable error scaling and support stable convergence to accurate solutions within a fixed training budget as the Helmholtz system becomes increasingly ill-conditioned.

We next consider the one-dimensional convection–diffusion equation with $N = 64$ basis functions and vary the diffusion coefficient over $\epsilon \in \{0.1, 0.05, 0.01, 0.005\}$. Decreasing ϵ produces increasingly convection-dominated solutions with sharp boundary layers. Alongside cond(A), we report the maximum ratio of the absolute imaginary part to the absolute real part across the eigenvalues, defined by

$$\rho(A) := \max_j \frac{|\mathrm{Im}\, \lambda_j(A)|}{|\mathrm{Re}\, \lambda_j(A)|}, \tag{62}$$

where $\lambda_j(A)$ denotes an eigenvalue of A. This quantity measures the relative magnitude of the imaginary and real parts of the eigenvalues and is distinct from a matrix condition number.

Table L.3 shows that $\rho(A)$ increases from 2.0362 to 19.6892, whereas the reported matrix condition numbers remain between 13.13 and 21.30. The prediction errors vary non-monotonically: the relative L_2 error reaches 1.67×10^{-2} at $\epsilon = 0.01$ and decreases to 5.68×10^{-3} at $\epsilon = 0.005$. Thus, these results demonstrate that the method retains useful accuracy in the tested convection-dominated regimes, while also revealing sensitivity to the diffusion parameter.

M Hardware-Inspired Noisy-Training Evaluation

We evaluate how gate and readout noise affect NVQLS training accuracy on the one-dimensional Helmholtz equation. We consider a noiseless reference and noisy training at two error levels: rates informed by device characteristics (hardware-inspired) and uniformly reduced rates (reduced noise).

Gate noise is modeled using uniform Pauli error channels. Let p_{1q} and p_{2q} denote the total error probabilities for one-qubit and two-qubit gates, respectively. After each noisy one-qubit gate, an X, Y, or Z error occurs with probability $p_{1q}/3$ each. After each CX gate, each of the 15 nonidentity two-qubit Pauli operators is applied with probability $p_{2q}/15$. Readout noise independently flips each measured bit with probability p_{ro}. These noise channels are included in every loss and gradient evaluation during training. Expectation values are computed exactly using density-matrix simulation, without finite-shot sampling.

As a simple measure of accumulated noise, we compute $P_{\geq 1}$, the probability of at least one modeled gate or readout error during a circuit execution. For the circuit considered here, assuming independent error events gives

$$P_{\geq 1} = 1 - (1 - p_{1q})^{213}(1 - p_{2q})^{178}(1 - p_{\mathrm{ro}})^{5}. \tag{63}$$

The exponents count 213 noisy single-qubit gates, 178 CNOT gates, and 5 readouts. At the hardware-inspired rates, $P_{\geq 1} \approx 57.1\%$. Multiplying all three error rates by 0.0119 gives the reduced-noise condition, for which $P_{\geq 1} \approx 1.0\%$.

Table M.1: NVQLS training loss and test relative L_2 error for the one-dimensional Helmholtz equation at different gate and readout error probabilities.

Setting	$(p_{1q}, p_{2q}, p_{\mathrm{ro}})$	Loss	Test rel. L_2 error
Ideal	$(0, 0, 0)$	4.09×10^{-9}	2.44×10^{-3}
Hardware-inspired	$(3.73 \times 10^{-4}, 3.75 \times 10^{-3}, 1.95 \times 10^{-2})$	3.32×10^{1}	2.68×10^{-1}
Reduced noise	$(4.44 \times 10^{-6}, 4.46 \times 10^{-5}, 2.32 \times 10^{-4})$	1.52×10^{-2}	2.79×10^{-2}

Table M.1 shows that the noiseless reference achieves a test relative L_2 error of 2.44×10^{-3}. At the hardware-inspired rates, the error is 2.68×10^{-1}, indicating a substantial gap from the noiseless reference. At the reduced rates, the error decreases to 2.79×10^{-2}, approximately 9.6 times smaller than at the hardware-inspired rates.

These results show that NVQLS training with the present circuit is sensitive to gate and readout noise. Reducing the modeled error rates substantially improves prediction accuracy, although a gap from the noiseless reference remains. Further evaluation across noise levels and circuit designs would help clarify how accumulated errors affect prediction accuracy and guide the development of circuits with fewer gates and reduced error accumulation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All major claims regarding the computational efficiency and predictive accuracy of the NVQLS framework are directly supported by the theoretical complexity analysis in Section 3 and the numerical experiments provided in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations and potential challenges of the proposed NVQLS, including its sensitivity to quantum gate noise and scalability to higher-dimensional systems, are discussed in the Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The

authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The conditional estimates of computational cost and classical working memory are summarized in Remark 3.1. The cost model, assumptions, detailed derivations, and limitations are provided in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, all theoretical and empirical details required to reconstruct the study are provided. The core NVQLS framework and problem formulations are explicitly defined in the main text and Appendix, while the exact experimental conditions, including architectures and optimization details, are fully documented in Appendices E and H.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Source code, pre-generated datasets, and a comprehensive README file with step-by-step instructions for reproduction are included in the supplementary material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the experimental setup in Section 4 and Appendices E and H. These sections describe the data generation procedure, data

splits, ansatz and angle-network architectures, optimizers, training budget, and the validation-based procedure used for hyperparameter selection.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In Table J.1., numerical results for the relative L^2 and L_∞ error are reported with their mean and standard deviation calculated across multiple independent test samples to ensure statistical significance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides the main compute-resource information in the appendix. The simulation details specify the training budget, software framework, and hardware configuration, including the CPU, RAM, GPU model, number of GPUs, and GPU memory in Table E.1, and the training epochs and total execution time of each experiment in Table E.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research methodology and its applications strictly adhere to the NeurIPS Code of Ethics. Our work focuses entirely on computational methods and mathematical frameworks, raising no ethical concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper presents a foundational method for scientific computing, whose potential benefits include more efficient simulation of physical and engineering systems under the stated assumptions.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk models or datasets requiring misuse safeguards. The experiments use synthetic PDE instances generated from specified mathematical distributions.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the prior methods and software libraries used in the experiments. No external datasets, pretrained models, or third-party visual assets are used; the PDE instances and figures are generated by the authors from the mathematical setup described in the paper.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release a new dataset, pretrained model, or other standalone asset. The experimental PDE instances are synthetically generated from the mathematical distributions described in the paper and appendix.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, human-subject experiments, or human-generated annotations. All experiments are conducted on synthetically generated PDE instances specified by mathematical distributions.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, crowdsourcing, user studies, or human-generated data. Therefore, IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as an important, original, or non-standard component of the core methodology, experiments, or scientific claims in this work.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.