LoRPS: Solving High Dimensional Integral Equations With Low-Rank Polynomials Sum

Anonymous authors
Paper under double-blind review

Abstract

Solving high-dimensional integral equations is a core challenge in science and engineering, primarily due to the curse of dimensionality. Classical numerical solvers, which rely on discretizing the problem domain, suffer from computational costs that grow exponentially with dimension. Monte Carlo methods, often used in machine learning, sidestep this scaling by providing convergence rates independent of dimension. However, they require a large number of samples to accurately approximate integrals. We show that naive sampling can bias the loss function, resulting in inaccurate solutions, especially in high-dimensional settings. We introduce Low Rank Polynomials Sum (LoRPS), a method that can scale to high dimension and solve the challenge of accurate integral estimation. LoRPS leverages a low-rank, separable polynomial structure that allows the integrals in the loss function to be computed analytically, avoiding any sampling-induced error while maintaining minimal computational overhead. We prove that under mild assumptions, LoRPS mitigates the curse of dimensionality. On challenging high-dimensional benchmarks, it consistently achieves higher accuracy while using at least 3.5× less memory than existing methods.

1 Introduction

Solving integral equations (IEs) is a longstanding challenge with applications across diverse domains, including quantum physics Laskin (2000), ecology Humphries et al. (2010), and finance Ros Oton (2014); Nolan (1999). Closed form solutions, while ideal, are often impractical or do not exist for many complex and significant IEs Fefferman (2000); Valtonen & Karttunen (2006), necessitating the development of numerical approximation methods. Several classical numerical techniques have been developed to address this challenge Godunov & Bohachevsky (1959); Huebner et al. (2001); Boyd (2001). These approaches typically rely on discretizing the problem domain into smaller sections, such as grids, and reformulating the original equation into a set of simpler subproblems often solved via quadrature. They are effective in low-dimensional settings and inherently avoid sampling-induced bias by providing deterministic approximations of the integrals. However, they do not scale well with increasing dimensionality, since the number of required discretization points grows exponentially with the number of dimensions, making them impractical for high-dimensional problems.

Recent advances in machine learning have inspired learning-based methods for solving integral equations Saleh et al. (2023); Lu et al. (2021). One of the most prominent approaches is the physics-informed neural network (PINN) framework Raissi et al. (2019), which trains neural networks by minimizing a loss function defined as an integral over the domain. In the case of IEs, the integrand itself contains another integral, reflecting the structure of the equation. Both the outer and inner integrals are typically approximated using Monte Carlo sampling. While each Monte Carlo estimate is unbiased on its own, the fact that the same samples are used both to optimize the model and to evaluate the inner integral introduces a dependency that biases the overall loss. This bias can cause the model to converge to an incorrect solution, a problem that becomes especially pronounced in high-dimensional settings, as we show later in section 3.

In this work, we introduce Low-Rank Polynomials Sum (LoRPS), a novel method for efficiently solving high-dimensional integral equations. Like PINNs, LoRPS is trained using gradient-based optimization. However, it takes a fundamentally different approach to evaluating the loss function. Instead of relying

on sampled collocation points, LoRPS expresses the residual as an exact integral over the domain, made tractable by its low-rank, separable polynomial structure. This formulation enables analytic integration along each dimension, eliminating the need for sampling altogether. As a result, LoRPS avoids the variance-related errors inherent in Monte Carlo-based integral estimation commonly used in machine learning approaches. Furthermore, by eliminating the need for collocation points and leveraging its low-rank structure, LoRPS scales efficiently to high-dimensional problems that are otherwise intractable for classical solvers due to the curse of dimensionality.

While LoRPS shares its training paradigm with PINNs, it also shares similarities to classical numerical methods. For example, the finite element method employs polynomial basis functions, but applies them piecewise over a discretized mesh. In high dimensions, this mesh refinement becomes increasingly costly, as the number of elements needed to maintain resolution grows exponentially with the dimension. This causes finite element methods to suffer from the curse of dimensionality. In contrast, LoRPS uses a global polynomial basis across the entire domain, avoiding domain partitioning altogether. Spectral methods also utilize global basis functions, such as Chebyshev polynomials or Fourier series, but they lack the low-rank, separable structure that distinguishes LoRPS. As a result, spectral methods require a number of coefficients that grows exponentially with dimension, while LoRPS only needs a number of parameters that scales linearly with the number of dimensions.

Furthermore, we present a strong theoretical foundation for LoRPS, showing that it can approximate any continuous function. For analytic target functions, the number of parameters required to achieve a given accuracy grows only polynomially with the dimensionality of the domain. This polynomial scaling, together with the ability to train without collocation points, enables LoRPS to scale efficiently to high dimensions where many existing methods become intractable.

Our contributions in this work are threefold. (1) We introduce LoRPS, a new approach for efficiently solving high-dimensional integral equations, and we will release an open-source implementation to support further research[1]. (2) We establish a rigorous theoretical framework for LoRPS, demonstrating its universal approximation capabilities and its ability to scale to high dimensions with parameter efficiency for analytic functions. (3) We conduct extensive experiments on a variety of high dimensional IEs, empirically validating the scalability and effectiveness of LoRPS in solving challenging problems.

2 Related Work

Classical Numerical Solvers for partial differential equations (PDEs) and IEs include finite difference Godunov & Bohachevsky (1959), finite element Huebner et al. (2001), finite volume Eymard et al. (2000), and spectral methods Boyd (2001). They discretize the domain into grids or elements and reformulate the equations into algebraic systems of equations. While effective in low dimensions, they become impractical in high-dimensional settings due to the curse of dimensionality, where computational cost and memory scale exponentially with dimension.

To address this, sparse grid techniques Bungartz & Griebel (2004) have been introduced to extend classical solvers to higher dimensions. These methods use hierarchical constructions of low-dimensional tensor grids to reduce complexity, allowing coarse discretizations to approximate higher-dimensional behavior Shen & Yu (2010); Conrad & Marzouk (2013); Bieri (2011); Wang et al. (2016). Although sparse grids significantly reduce the number of required points, they still exhibit exponential dependence through logarithmic factors and remain limited to moderately high dimensions. In contrast, LoRPS uses a low-rank, separable polynomial representation with analytic integration and parameter-efficient training, enabling it to scale more effectively to high-dimensional IEs.

Machine Learning Methods have been proposed to solve various high-dimensional PDEs Han et al. (2017); Yu et al. (2018); Nam et al. (2024); Zang et al. (2020); Beck et al. (2021); Raissi (2024). However, these methods are often limited to specific types of equations, such as parabolic or elliptic equations, and cannot address high-dimensional IEs.

[1]GitHub repository will be available upon publication.

PINNs and their variants Raissi et al. (2019); Wong et al. (2022); Bu & Karpatne (2021); Zhao et al. (2023) are widely used learning-based methods for solving PDEs and IEs. These models train neural networks using gradient-based optimization, embedding the governing equations and boundary conditions into the loss function. Extensions such as hp-VPINNs Kharazmi et al. (2021), VarNet Khodayi-Mehr & Zavlanos (2020), and VPINNs Kharazmi et al. (2019) incorporate polynomial test functions and formulate the loss as weighted integrals of the PDE residual. These integrals are typically evaluated with numerical quadrature, introducing approximation errors and causing computational costs to grow rapidly with dimension. Monte Carlo sampling is often used as an alternative, but in the context of IEs, it introduces variance that can lead to biased solutions. Techniques such as the Double-Sampling Trick Saleh et al. (2023) and Quasi Monte Carlo Dick et al. (2013); Giles & Waterhouse (2009) reduce this variance but do not eliminate it. In contrast, LoRPS computes integrals analytically, avoiding both quadrature and sampling, resulting in unbiased solutions and better scalability in high-dimensional problems.

Several studies have aimed to improve the efficiency of PINNs. Domain decomposition methods Jagtap & Karniadakis (2020); Moseley et al. (2023); Jagtap et al. (2020) train separate networks on subdomains to enable parallelization. Separable PINNs Cho et al. (2024) assign a neural network to each input variable and combine their outputs through separable formulations. Although these architectures reduce computational cost, they remain limited in high-dimensional settings. Other works Hu et al. (2024b;a); Yu et al. (2022); Shi et al. (2024) have focused on improving the optimization process to enhance scalability and accuracy. These efforts are complementary to our approach, as LoRPS also employs gradient-based optimization.

Tensor Train Decomposition Based Methods represent high-dimensional tensors by factorizing them into a sequence of low-dimensional cores, significantly reducing the number of parameters from exponential to linear in the number of dimensions under bounded rank assumptions Oseledets (2011). Recent extensions apply this framework to PDEs using hierarchical decompositions with dynamic orthogonality and rank-adaptive time integration Dektor & Venturi (2020); Dektor et al. (2021). LoRPS shares the low-rank modeling objective but uses a sum of separable polynomial terms instead of chained tensor cores.

A related line of work models PDE solutions as sums of separable factors using neural or spectral parameterizations Wang et al. (2022; 2024). These approaches are formulated around differential-operator residual or variational objectives and therefore do not apply to integral-equation operators without additional machinery. In contrast, LoRPS is designed specifically for integral equations and enables exact analytical computation of the operator integrals and derivatives within its polynomial-sum representation. This eliminates the numerical errors introduced by quadrature in neural network models Rivera et al. (2022); Taylor & Pardo (2025) and circumvents the substantial memory and computational overhead associated with building gradient graphs for high-order derivatives.

Neural Operator Learning solves entire classes of partial differential equations and IEs, generalizing efficiently to problem variations without requiring re-solving. DeepONet Lu et al. (2019) introduced function-to-function mappings, enabling fast partial differential equations solutions after training. Fourier Neural Operators Li et al. (2020a) extend this by learning in the frequency domain for global efficiency, while graph-based neural operators Li et al. (2020b) enhance integration approximation. However, these methods discretize input spaces, making them susceptible to the curse of dimensionality and impractical in high-dimensional settings.

3 Problem Formulation

We consider IEs of the form

$$f(x) + \int_{\mathcal{D}(x)} K(x, y)u(y)dy - u(x) = 0, \quad x \in \Omega \tag{1}$$

This formulation includes Fredholm equations and Volterra equations which are widely used in various scientific fields Simons et al. (2006); Daddi-Moussa-Ider et al. (2019); Brunner (2017).

In this formulation K is the kernel function, $f(x)$ is a known function and $u(x)$ is the unknown. The domain Ω is a subset of $\mathbb{R}^d$ and $\mathcal{D}(x) \subset \Omega$; throughout, we consider rectangular domains, i.e., $\Omega = \Omega_1 \times \cdots \times \Omega_d$. This

choice is convenient for our setting and covers many practically relevant problems. For instance, numerous physical models are posed on rectangular domains Binous et al. (2017); Lamelas (2022); Pereira et al. (2018), and others, such as the time-independent Schrodinger equation for many-body systems, are often solved on rectangular domains for practical reasons Gerard et al. (2022); Gao & Günnemann (2024). We assume the problem has a smooth solution $u(x)$ with d variables.

Monte Carlo based methods, which are common in machine learning approaches such as PINN, approximate the solution function $u(x)$ by a neural network $u_\theta(x)$, where θ represents the network parameters. For notational simplicity, we define:

$$h_\theta(x, y) := K(x, y)u_\theta(y) \tag{2}$$

As standard in the field, training involves minimizing the loss function that minimizes the error for every $x \in \Omega$ on average. The standard loss function is defined as:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(f(\tilde{x}_i) + \int_{\mathcal{D}(\tilde{x}_i)} h_\theta(\tilde{x}_i, y)\, dy - u_\theta(\tilde{x}_i)\right)^2 \tag{3}$$

Since computing exact integrals is usually impractical, standard machine learning methods replace it with an additional expectation as estimation of the integral:

$$\widehat{\mathcal{L}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(f(\tilde{x}_i) + \frac{A_{\tilde{x}_i}^{-1}}{M}\sum_{j=1}^{M} h_\theta(\tilde{x}_i, \hat{y}_j) - u_\theta(\tilde{x}_i)\right)^2 \tag{4}$$

Here, the normalization factor

$$A_x := |\mathcal{D}(x)|^{-1} = \left(\int_{\mathcal{D}(x)} 1\, d\hat{y}\right)^{-1} \tag{5}$$

ensures that the discrete sum approximates the continuous integral as an expectation. The sets $\{\tilde{x}_i\}_{i=1}^N \subset \Omega$ and $\{\hat{y}_j\}_{j=1}^M \subset \mathcal{D}(\tilde{x}_i)$ are sampled independently and identically distributed (i.i.d.). However, this approximation introduces a bias into the loss function (full derivation in appendix B):

$$\widehat{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + \frac{1}{N}\sum_{i=1}^{N}\frac{A_{\tilde{x}_i}^{-2}}{M}\,\mathbb{V}_{P(y|\tilde{x}_i)}\left[h_\theta(\tilde{x}_i, y)\right] \tag{6}$$

Where $\mathbb{V}_{P(y|\tilde{x}_i)}[\cdot]$ denotes the variance under the sampling distribution $P(y|\tilde{x}_i)$.

The additional variance introduced by the approximation can bias the optimal solution. As a result, minimizing the approximate loss may favor smoother functions and lead to suboptimal or incorrect solutions. Consider the following simple example:

$$u(x) - \int_{\mathcal{D}} u(y)\left(\sum_{i=1}^{d} x_i y_i^3 y_{\phi(i+1)} y_{\phi(i+2)} + \alpha y_i\right) dy = 1 \tag{7}$$

for all $x \in \Omega = \left[-\frac{1}{2}, \frac{1}{2}\right]^d$, where

$$d = 20, \quad \mathcal{D} = \Omega, \quad \phi(i) = ((i-1) \bmod d) + 1 \tag{8}$$

In this equation, the kernel is an odd function over the symmetric domain $\mathcal{D}$. As a result, when $u(x)$ is an even function, the integral term vanishes. This symmetry implies that the exact solution is simply:

$$u(x) = 1, \quad \forall x \in \Omega \tag{9}$$

Table 1: Performance comparison of Monte Carlo-based methods across different values of α, which controls the variance in Equation 7. We report relative L^2 error over five seeds. LoRPS remains accurate even at high variance levels, while the other methods deteriorate. Additional experimental details are provided in Section 5 and the appendix.

Method	$\alpha = 1$	$\alpha = 25$	$\alpha = 50$	$\alpha = 75$
Uniform	$5.00e{-3} \pm 1.68e{-3}$	$9.86e{-1} \pm 4.68e{-3}$	$1.00e{+0} \pm 6.20e{-3}$	$1.00e{+0} \pm 1.50e{-3}$
Sobol	$1.60e{-3} \pm 6.02e{-6}$	$3.80e{-2} \pm 1.05e{-4}$	$7.37e{-2} \pm 5.49e{-4}$	$1.06e{-1} \pm 1.38e{-3}$
Double	$4.93e{-3} \pm 1.09e{-3}$	$4.49e{+0} \pm 7.66e{+0}$	$1.01e{+0} \pm 1.11e{-1}$	$1.42e{+0} \pm 1.39e{+0}$
LoRPS	$\mathbf{1.13e{-4} \pm 1.18e{-4}}$	$\mathbf{1.49e{-4} \pm 9.29e{-5}}$	$\mathbf{3.44e{-4} \pm 2.95e{-4}}$	$\mathbf{9.08e{-4} \pm 8.99e{-4}}$

Despite the simplicity of the solution, classical numerical solvers struggle to solve this problem due to the high dimensionality and the non-separable nature of the kernel, which induces strong coupling among the variables. The parameter α serves as a hyperparameter that controls the variance of the integral term. As illustrated in table 1, increasing α leads to higher variance, causing Monte Carlo-based methods to fail to converge to the correct solution.

4 Low-Rank Polynomials Sum

4.1 Method

To tackle the challenges outlined in Section 3, we introduce a new class of trainable models called Low-Rank Polynomial Sum (LoRPS), defined as follows:

$$\begin{aligned} g_\theta(x) &= \sum_{i=1}^{s} \prod_{j=1}^{d} f_{ij}(x_j; \theta_{ij}) \\ &= f_{11}(x_1; \theta_{11}) f_{12}(x_2; \theta_{12}) \cdots f_{1d}(x_d; \theta_{1d}) + \cdots \\ &\quad + f_{s1}(x_1; \theta_{s1}) f_{s2}(x_2; \theta_{s2}) \cdots f_{sd}(x_d; \theta_{sd}) \end{aligned} \tag{10}$$

Each function $f_{ij}(\cdot; \theta_{ij})$ is a one-dimensional polynomial (not restricted to monomials) of degree k; in particular, we have that

$$f_{ij}(x_j) = \theta_{ij}^T Q(x_j) \quad \text{where} \quad Q(z) = [z^0, \ldots, z^k]^T \tag{11}$$

The overall set of LoRPS parameters is $\theta = \{\theta_{ij}\}$ which are trainable. The hyperparameters s and k control the number of separable polynomials in the sum and the degree of the one-dimensional polynomials, respectively. In the context of solving IEs, the goal will be for the solution function $u(x)$ to be approximated by a LoRPS $g_\theta(x)$, where θ represents the LoRPS model's learnable parameters.

Thanks to its structure, LoRPS can be interpreted as a low rank approximation (specifically of rank s) of the solution function to the IE. It provides a compact and efficient representation of high-dimensional polynomials. Instead of expanding the full multivariate polynomial, which would require $(k+1)^d$ coefficients for a degree k polynomial in d dimensions, LoRPS maintains a factorized form that uses only $s(k+1)d$ independent trainable parameters. This factorization imposes a low-rank structure on the function representation, significantly reducing the number of parameters and computational cost.

The primary advantage of LoRPS is its natural compatibility with integration and differentiation, stemming from the linearity of these operations, the separable structure of LoRPS, and the simplicity of integrating and differentiating polynomials. This property significantly simplifies computations. In the case of integration, recall that the domain Ω, assumed to be rectangular, can be decomposed into domains for each variable, i.e. $\Omega = \Omega_1 \times \cdots \times \Omega_d$, and let us define $\Omega_j = [a_j, b_j]$. Furthermore, define $R(z) = [z, \frac{1}{2}z^2, \ldots, \frac{1}{k+1}z^{k+1}]^T$. In

this case, we have that the integration of a LoRPS is given by

$$\begin{aligned}\int_\Omega g_\theta(x)dx &= \sum_{i=1}^{s}\prod_{j=1}^{d}\int_{\Omega_j} f_{ij}(x_j)dx_j \\ &= \sum_{i=1}^{s}\prod_{j=1}^{d}\theta_{ij}^T(R(b_j)-R(a_j))\end{aligned} \tag{12}$$

Thus, this integration is computed analytically as we have just demonstrated, leveraging the polynomial structure of LoRPS. This allows the computation to be reduced to d one-dimensional integrals instead of a single d-dimensional integral. This reduction is essential, since numerical integration methods such as quadrature become increasingly inefficient in high dimensions due to the curse of dimensionality.

As a result, LoRPS allows us to train on the entire problem domain, independent of its dimensionality, by using a loss function defined as the integral over the domain instead of relying on collocation points:

$$\mathcal{L}(\theta) = \int_\Omega \Big(f(x) + \int_{\mathcal{D}(x)} K(x,y)u_\theta(y)\,dy - u_\theta(x)\Big)^2 dx \tag{13}$$

When g_θ is expressed as a LoRPS and the functions involved, such as f and K, are polynomial, any combination of these functions through integration, differentiation, multiplication, or addition will preserve the LoRPS structure. This follows from the closure properties of polynomials under these operations. The assumption of polynomial structure applies throughout the setting considered here, and the treatment of non-polynomial components, such as a non-polynomial K, is discussed in Section 4.3. Under these conditions, the loss integral can be evaluated analytically, which avoids the variance-related bias of Monte Carlo sampling and the exponential memory and computational cost of high-dimensional numerical integration.

4.2 Universal Approximation and Convergence Rates

In this section, we establish the theoretical foundation of LoRPS. We begin by showing that LoRPS exhibits universal approximation capabilities. Our main theoretical contribution is a convergence rate analysis demonstrating that LoRPS can approximate analytic functions with a rate that does not depend on the number of dimensions. This result indicates that LoRPS can mitigate the curse of dimensionality for this class of functions. To the best of our knowledge, such guarantees have not been previously established for sparse (best n-term) approximation in high-dimensional settings. Here, we extend existing convergence rate results to this regime and apply them specifically to LoRPS.

Theorem 1. *Let $u : \Omega \to \mathbb{R}$ be a continuous function defined on a compact domain $\Omega \subseteq \mathbb{R}^d$. For sufficiently large s and k, the Low Rank Separable Polynomials Sum representation:*

$$g_\theta(x) = \sum_{i=1}^{s}\prod_{j=1}^{d} f_{ij}(x_j;\theta_{ij}) \tag{14}$$

can approximate $u(x)$ arbitrarily well in the L^∞-norm. Specifically, for any $\epsilon > 0$, there exist values of s, k, and coefficients $\theta = \{\theta_{ij}\}$ such that:

$$\sup_{x\in\Omega} |u(x) - g_\theta(x)| < \epsilon \tag{15}$$

Proof. This result follows from the fact that for sufficiently large s and k, LoRPS can represent any multivariate polynomial. The proof then follows by direct application of the Stone–Weierstrass theorem. □

Despite this theoretical ability of LoRPS to approximate all continuous functions, the question remains as to whether it can do so *efficiently*. It is well known that for certain function spaces, such as the Sobolev space H^m, the curse of dimensionality is unavoidable. For example, Blanchard & Bennouna (2022) demonstrates

that even neural networks fail to overcome this limitation for such spaces. Similarly, LoRPS is subject to the curse of dimensionality when applied to functions in the Sobolev space H^m.

However, for other function classes, specifically analytic functions, LoRPS exhibits much more favorable behavior. In this setting, we show that LoRPS achieves rapid convergence, with a rate that is independent of the dimension d. To establish this result, we restrict our attention to functions defined on the domain $\Omega_1 := [-1, 1]^d$, and leverage results from the theory of Legendre polynomial approximation. In particular, by applying Stechkin's inequality DeVore (1998); Cohen et al. (2011) and proving the summability of Legendre coefficients, we demonstrate that LoRPS maintains computational tractability in high dimensions for this important function class. The theoretical results over Ω_1 apply to other rectangular tensor-product domains, since they can be reduced to Ω_1 by coordinate-wise affine rescaling, resulting only in changes to the implicit constants.

Theorem 2. *If u is a real analytic function on an open neighborhood of Ω_1 then there exists a LoRPS representation g_θ with s terms which is ε-close to u in the sense that $\|u - g_\theta\|_{L^2} \leq \varepsilon$ as long as $s = O(\varepsilon^{-2})$.*

Proof. See the appendix. □

Theorem 2 demonstrates that the asymptotic convergence rate with respect to the number of terms s is independent of the dimension d. However, it remains to be understood how the parameter k, which controls the polynomial degree in LoRPS, scales with d for accurate approximation. In what follows, we show that LoRPS achieves exponential convergence with respect to k.

Theorem 3. *If u is a real analytic function defined on Ω_1, extensible to a complex polydisk $\mathcal{D} := \mathcal{D}_{r_1} \times \mathcal{D}_{r_2} \times \cdots \times \mathcal{D}_{r_d}$ where $D_r = \{z \mid r > |z|\}$ and $r_i > 1$ for every dimension $i = 1, 2, \ldots, d$ then there exists a LoRPS representation g_θ using terms up to degree k which is ε-close to u in the sense that $\|u - g_\theta\|_{L^2} \leq \varepsilon$ as long as $k = O(\log \frac{1}{\varepsilon})$.*

Proof. See the appendix. □

Theorem 3 shows that the required polynomial degree k in each univariate component grows only logarithmically with the desired accuracy ε. This implies that the primary complexity bottleneck in LoRPS is not the degree k, but rather the number of rank-1 terms s needed to achieve a given approximation error.

4.2.1 Comments and Implications

Theorem 2 establishes a convergence rate of $O(\varepsilon^{-2})$ for LoRPS that is independent of the dimension d, highlighting its theoretical ability to mitigate the curse of dimensionality.

In practice, many physical systems are well-approximated by sparse, low-degree polynomials. As noted in Lin et al. (2017), this is due in part to the structure of fundamental equations: for example, the Hamiltonian in the Standard Model is of degree 4, and many governing equations in physics (e.g., Maxwell, Navier–Stokes, Alfvén, and magnetism models) involve polynomials of degree 2–4 in the field variables. Additionally, the local nature of physical interactions often induces sparsity in polynomial representations.

The analyticity assumption in Theorem 2 is not overly restrictive. Many PDEs admit analytic solutions when their boundary conditions and coefficients are analytic, including elliptic PDEs with constant or real-analytic coefficients Treves et al. (2022) and analogues of the heat equation. By the Cauchy–Kovalevskaya theorem, even certain nonlinear PDEs admit local analytic solutions. Moreover, all linear Fredholm integral equations of the second kind with analytic kernels and right-hand sides yield analytic solutions Kress (1989).

4.3 Solving Equations with Non-Polynomial Terms

In Section 4.1, we implicitly assumed that the problems solved by LoRPS involve only polynomial terms. This assumption ensures that the final form of the equation remains a polynomial when LoRPS is used as the solution function, as polynomials are closed under addition, multiplication and differentiation. This

assumption is crucial, as non-polynomial terms can complicate integration and other operations that are otherwise straightforward when dealing with polynomials.

However, LoRPS can still address problems involving non-polynomial terms by approximating them as polynomials before solving the equation. As demonstrated in Section 4.2, LoRPS can approximate any continuous function. Therefore, separate LoRPS models can be trained to represent the non-polynomial terms prior to solving the equation; we also empirically validate this approach in Section 5 by solving equations that include such terms (and more details are provided in the Appendix C.4).

The training of these models relies on Monte Carlo methods, as in PINNs. Once the required approximations are learned, the resulting LoRPS models are substituted into the original equation, allowing another LoRPS model to solve the problem effectively. As shown in the experiments, this approach performs well across a range of problems involving non-polynomial terms.

While the reliance on Monte Carlo sampling may appear circular, it is important to note that in many high-dimensional problems, the terms being approximated often live in significantly lower-dimensional spaces than the full solution and can therefore be learned efficiently with Monte Carlo. For example, in the many-body Schrodinger equation, the potential decomposes into pairwise interactions, each depending on only six variables, whereas the wavefunction is defined over the full high-dimensional configuration space of all particles. This dimensionality gap keeps the sampling burden manageable, and moreover the resulting auxiliary approximations can be treated as a one-time preprocessing cost that is amortized over the many optimization steps (and repeated evaluations) of the main LoRPS solve, which then proceeds with exact analytic integration on the full domain.

Moreover, fitting individual terms directly with a simple MSE is often much easier than minimizing a loss function derived from the full PDEs or IEs Krishnapriyan et al. (2021); Rathore et al. (2024). The resulting loss landscapes are typically smoother and more tractable, leading to more stable and efficient training.

5 Experiments

5.1 Experimental Setup

We evaluate our method on three high-dimensional integral equations: a convolution equation, non-local diffusion equation, and the heat equation with memory. These equations are commonly used in various scientific fields, including signal processing Simons et al. (2006), heat conduction with memory effects Miller (1978), and biology Humphries et al. (2010); GM (1996). Full definitions of the equations, along with their boundary and initial conditions, are provided in the appendix.

We compare LoRPS with three Monte Carlo-based strategies: uniform sampling (standard Monte Carlo), Sobol sampling Owen (1998); Sobol (1967), and double sampling Saleh et al. (2023). Each method is evaluated in combination with a variety of neural network architectures, including the classical PINN framework Raissi et al. (2019), as well as more advanced models. These include FLS Wong et al. (2022), a method that enhances the representational power of neural networks by introducing specially designed sinusoidal activation functions; QRes Bu & Karpatne (2021), which employs quadratic residual connections; and PINNsFormer Zhao et al. (2023), a transformer-based architecture that incorporates self-attention mechanisms and wavelet-inspired activation functions.

Because the considered problems are posed in high dimensional domains, classical numerical solvers such as finite element or spectral methods become computationally impractical, both in terms of memory and runtime. Therefore, we focus our evaluation exclusively on machine learning based methods that rely on Monte Carlo sampling, which scales more favorably in high dimensions and allows efficient estimation of integral based losses. To ensure consistency and fair comparison each baseline is trained for a maximum of 120 minutes per equation on an NVIDIA L4 GPU.

The primary evaluation metric is the average relative L^2 error, defined as $\frac{\|\hat{u}-u\|_2}{\|u\|_2}$, where $\hat{u}$ denotes the model prediction and u is the reference solution. This metric is standard in the literature and is consistent with

prior work Cho et al. (2024); Nam et al. (2024); Wu et al. (2024). Further implementation details, including baseline hyperparameters, and additional results are provided in the appendix.

Table 2: Comparison of memory usage and error metrics across different architectures for three high-dimensional integral equations. For each method, the reported error is the average at the epoch where its loss reached the minimum, using its best sampling strategy. Memory is the maximum observed during training. Results are averaged over five seeds, with training curves shown in Figure 1.

Method	Convolution Eq.			Non-Local Diffusion Eq.			Heat with Memory Eq.		
	Mem. (GB)	Rel. L^2	RMSE	Mem. (GB)	Rel. L^2	RMSE	Mem. (GB)	Rel. L^2	RMSE
FLS	17.5	$1.0e{-3}$	$1.5e{-3}$	8.2	2.60	$2.4e{-3}$	11.0	1.96	$1.2e{-3}$
PINN	17.5	$1.3e{-3}$	$1.2e{-3}$	8.0	2.28	$2.3e{-3}$	10.4	2.20	$1.4e{-3}$
QRes	9.9	$2.2e{-2}$	$3.3e{-2}$	10.8	2.93	$2.8e{-3}$	12.1	1.20	$7.0e{-4}$
PINNsFormer	13.9	$6.9e{-2}$	$1.05e{-1}$	10.7	11.6	$1.1e{-2}$	2.7	1.52	$8.0e{-4}$
LoRPS	**0.45**	**2.35e−4**	**3.6e−4**	**0.60**	**1.8e−2**	**1.8e−5**	**0.30**	**1.2e−4**	**7.6e−8**

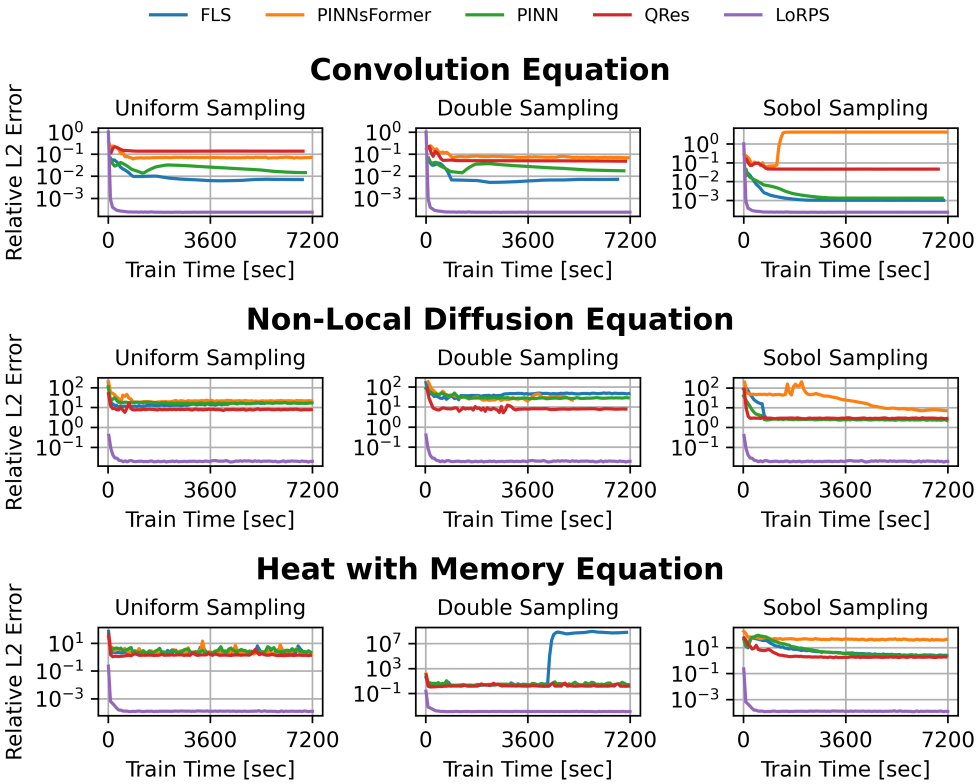

Figure 1: Relative L^2 errors for all baselines, evaluated over five independent trials using Monte Carlo integration with 10^5 uniformly sampled unseen points from Ω. Results are reported for three high-dimensional integral equations across different Monte Carlo sampling strategies and neural architectures. LoRPS consistently achieves the lowest error.

5.2 Results

Figure 1 shows the evolution of the relative L^2 test error during training. LoRPS consistently achieves errors at least an order of magnitude lower than competing approaches. On the heat-with-memory task under the double-sampling strategy, FLS was unstable in one of five runs: after initially decreasing, the test error

increased sharply, which skewed the mean. In that run, the loss exhibited large oscillations, indicating that the double-sampling estimator can be unstable in this setting. PINNsFormer performs worst overall, likely due to the quadratic scaling of self-attention in the number of collocation points, which limits the usable sample size in high dimensions.

We get a summary of the final performance in table 2. For each method, we report the test error averaged over five runs, evaluated at the epoch of minimum training loss under its best-performing sampling strategy; we also report peak memory usage. LoRPS achieves the lowest errors while using substantially less memory than the baselines (3.5×–25× less), yielding the best accuracy–efficiency trade-off.

Table 3: Memory consumption and accuracy of LoRPS when solving the d-dimensional convolution integral equation.

Dimension d	5	10	15	20	25	30
Memory (MB)	260	280	339	450	630	890
Rel. L^2 Error	$3.5e{-4}$	$2.8e{-4}$	$2.5e{-4}$	$2.35e{-4}$	$2.1e{-4}$	$4.2e{-4}$

Table 4: Inference-speed comparison against the baselines. Forward-pass times were measured on an NVIDIA L4 GPU using batches of 20,000 random points in 20 dimensions and averaged over 100 runs.

Method	Forward time (s)	Slowdown vs. LoRPS
LoRPS	$\mathbf{2.64e{-4} \pm 1.1e{-5}}$	**1.0**×
PINN	$8.38e{-4} \pm 1.00e{-4}$	3.17×
FLS	$8.59e{-4} \pm 1.3e{-5}$	3.25×
QRes	$3.705e{-3} \pm 2.9e{-5}$	14.0×
PINNsFormer	$1.65023e{-1} \pm 7.71e{-4}$	625×

5.3 Robustness and Stability Analysis

We study the sensitivity of LoRPS to its key hyperparameters: the rank s and the maximum degree k. We conduct this analysis on the heat-with-memory equation. Figure 2 reports the test error across a grid of (s, k) values. As expected for an oscillatory solution with trigonometric components, accurate approximation requires sufficiently large s and k (typically $s \gtrsim 1$ and $k \gtrsim 5$); beyond this regime, performance is consistently strong, indicating stable behavior. We further evaluate scalability by solving the convolution equation for dimensions $d \in \{5, 10, 15, 20, 25, 30\}$ (Table 3); in all cases, peak memory stays below 1 GB while maintaining low relative L_2 error.

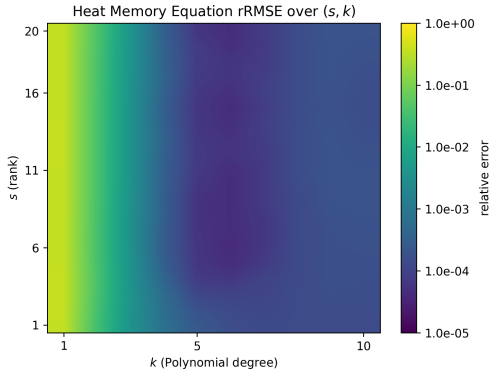

Figure 2: Hyperparameter sensitivity analysis on the heat with memory equation. The heatmap displays the relative L^2 error across varying rank s and polynomial degree K.

6 Limitations

Despite the effectiveness of LoRPS in solving high-dimensional integral equations where existing methods often struggle, the method has several limitations. First, LoRPS is designed for problems with smooth solutions, consistent with the assumptions used in our theoretical analysis. Since the approximation space is based on polynomial expansions, we do not expect the method to perform well on solutions with discontinuities, sharp localized features, or other forms of low regularity. Similarly, highly oscillatory solutions may require a large polynomial degree in order to be represented accurately, which can increase the computational cost.

Another limitation is that the current formulation is restricted to rectangular domains. In some cases, this restriction can be mitigated through suitable changes of variables, such as spherical, polar, or cone-type transformations, which map more general geometries to rectangular computational domains. However, extending LoRPS to handle complex geometries more directly remains an important direction for future work.

7 Conclusion

We presented Low-Rank Polynomials Sum, a new model designed to solve high-dimensional integral equations efficiently. LoRPS is trained with a loss function expressed as an integral over the problem domain, and its polynomial structure allows this integral to be computed analytically without incurring significant computational cost. We established theoretical guarantees showing that LoRPS can approximate any smooth function on a compact domain, and for analytic target functions, the convergence rate of LoRPS is independent of the dimensionality. This property enables LoRPS to scale to high-dimensional problems where the computational cost of classical methods would explode with the demand for higher precision. Extensive experiments on challenging high-dimensional equations confirm that LoRPS consistently outperforms existing approaches while requiring substantially less memory.

References

Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. Deep splitting method for parabolic pdes. *SIAM Journal on Scientific Computing*, 43(5):A3135–A3154, 2021.

Marcel Bieri. A sparse composite collocation finite element method for elliptic spdes. *SIAM Journal on Numerical Analysis*, 49(6):2277–2301, 2011.

Housam Binous, Slim Kaddeche, and Ahmed Bellagi. Solution of six chemical engineering problems using the chebyshev orthogonal collocation technique. *Computer Applications in Engineering Education*, 25(4): 594–607, 2017.

Moise Blanchard and Mohammed Amine Bennouna. Shallow and deep networks are near-optimal approximators of korobov functions. In *International conference on learning representations*, 2022.

John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

Hermann Brunner. *Volterra integral equations: an introduction to theory and applications*, volume 30. Cambridge University Press, 2017.

Jie Bu and Anuj Karpatne. Quadratic residual networks: A new class of neural networks for solving forward and inverse problems in physics involving pdes. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 675–683. SIAM, 2021.

Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Albert Cohen and Ronald DeVore. Approximation of high-dimensional parametric pdes. *Acta Numerica*, 24: 1–159, 2015.

Albert Cohen, Ronald Devore, and Christoph Schwab. Analytic regularity and polynomial approximation of parametric and stochastic elliptic pde's. *Analysis and Applications*, 9(01):11–47, 2011.

Patrick R Conrad and Youssef M Marzouk. Adaptive smolyak pseudospectral approximations. *SIAM Journal on Scientific Computing*, 35(6):A2643–A2670, 2013.

Abdallah Daddi-Moussa-Ider, Badr Kaoui, and Hartmut Löwen. Axisymmetric flow due to a stokeslet near a finite-sized elastic membrane. *Journal of the Physical Society of Japan*, 88(5):054401, 2019.

Alec Dektor and Daniele Venturi. Dynamically orthogonal tensor methods for high-dimensional nonlinear pdes. *Journal of Computational Physics*, 404:109125, 2020.

Alec Dektor, Abram Rodgers, and Daniele Venturi. Rank-adaptive tensor methods for high-dimensional nonlinear pdes. *Journal of Scientific Computing*, 88(2):36, 2021.

Ronald A DeVore. Nonlinear approximation. *Acta numerica*, 7:51–150, 1998.

Josef Dick, Frances Y Kuo, and Ian H Sloan. High-dimensional integration: the quasi-monte carlo way. *Acta Numerica*, 22:133–288, 2013.

Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000.

Charles L Fefferman. Existence and smoothness of the navier-stokes equation. *The millennium prize problems*, 57:67, 2000.

Nicholas Gao and Stephan Günnemann. Neural pfaffians: Solving many many-electron schrödinger equations. *Advances in Neural Information Processing Systems*, 37:125336–125369, 2024.

Leon Gerard, Michael Scherbela, Philipp Marquetand, and Philipp Grohs. Gold-standard solutions to the schrödinger equation using deep learning: How much physics do we need? *Advances in Neural Information Processing Systems*, 35:10282–10294, 2022.

Michael B Giles and Benjamin J Waterhouse. Multilevel quasi-monte carlo path simulation. *Advanced Financial Modelling, Radon Series on Computational and Applied Mathematics*, 8:165–181, 2009.

VISWANTHAN GM. Levy flight search patterns of wandering albatrosses. *Nature*, 381:413415, 1996.

Sergei K Godunov and I Bohachevsky. Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics. *Matematičeskij sbornik*, 47(3):271–306, 1959.

Jiequn Han, Arnulf Jentzen, et al. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in mathematics and statistics*, 5(4):349–380, 2017.

Zheyuan Hu, Zekun Shi, George Em Karniadakis, and Kenji Kawaguchi. Hutchinson trace estimation for high-dimensional and high-order physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 424:116883, 2024a.

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

Kenneth H Huebner, Donald L Dewhirst, Douglas E Smith, and Ted G Byrom. *The finite element method for engineers*. John Wiley & Sons, 2001.

Nicolas E Humphries, Nuno Queiroz, Jennifer RM Dyer, Nicolas G Pade, Michael K Musyl, Kurt M Schaefer, Daniel W Fuller, Juerg M Brunnschweiler, Thomas K Doyle, Jonathan DR Houghton, et al. Environmental context explains lévy and brownian movement patterns of marine predators. *Nature*, 465(7301):1066–1069, 2010.

Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5), 2020.

Ameya D Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.

Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*, 2019.

Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021.

Reza Khodayi-Mehr and Michael Zavlanos. Varnet: Variational neural networks for the solution of partial differential equations. In *Learning for dynamics and control*, pp. 298–307. PMLR, 2020.

Steven G. Krantz and Harold R. Parks. *A Primer of Real Analytic Functions*. Birkhäuser, 1992.

R Kress. Linear integral equations, 1989.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

Chami Lamelas. *Fluid Solver for Incompressible Flow in a Rectangular Domain with Arbitrarily-placed Inclusions*. PhD thesis, Brandeis University, 2022.

Nikolai Laskin. Fractional quantum mechanics and lévy path integrals. *Physics Letters A*, 268(4-6):298–305, 2000.

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

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168:1223–1247, 2017.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

RK Miller. An integrodifferential equation for rigid heat conductors with memory. *Journal of Mathematical Analysis and Applications*, 66(2):313–332, 1978.

Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Finite basis physics-informed neural networks (fbpinns): a scalable domain decomposition approach for solving differential equations. *Advances in Computational Mathematics*, 49(4):62, 2023.

Hong Chul Nam, Julius Berner, and Anima Anandkumar. Solving poisson equations using neural walk-on-spheres. *arXiv preprint arXiv:2406.03494*, 2024.

JP Nolan. Fitting data and assessing goodness-of-fit with stable distributions. department of mathematics and statistics, american university, 1999.

Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Art B Owen. Scrambling sobol'and niederreiter–xing points. *Journal of complexity*, 14(4):466–489, 1998.

FD Quesada Pereira, C Gómez Molina, A Alvarez Melcon, VE Boria, and Marco Guglielmi. Novel spatial domain integral equation formulation for the analysis of rectangular waveguide steps close to arbitrarily shaped dielectric and/or conducting posts. *Radio Science*, 53(4):406–419, 2018.

Maziar Raissi. Forward–backward stochastic neural networks: deep learning of high-dimensional partial differential equations. In *Peter Carr Gedenkschrift: Research Advances in Mathematical Finance*, pp. 637–655. World Scientific, 2024.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training pinns: A loss landscape perspective. *arXiv preprint arXiv:2402.01868*, 2024.

Jon A Rivera, Jamie M Taylor, Ángel J Omella, and David Pardo. On quadrature rules for solving partial differential equations using neural networks. *Computer Methods in Applied Mechanics and Engineering*, 393:114710, 2022.

Xavier Ros Oton. Integro-differential equations: Regularity theory and pohozaev identities. 2014.

Ehsan Saleh, Saba Ghaffari, Timothy Bretl, Luke Olson, and Matthew West. Learning from integral losses in physics informed neural networks. *arXiv preprint arXiv:2305.17387*, 2023.

Volker Scheidemann. *Introduction to Complex Analysis in Several Variables*. Birkhäuser Cham, 2023.

Jie Shen and Haijun Yu. Efficient spectral sparse grid methods and applications to high-dimensional elliptic problems. *SIAM Journal on Scientific Computing*, 32(6):3228–3250, 2010.

Zekun Shi, Zheyuan Hu, Min Lin, and Kenji Kawaguchi. Stochastic taylor derivative estimator: Efficient amortization for arbitrary differential operators. *Advances in Neural Information Processing Systems*, 37: 122316–122353, 2024.

Frederik J Simons, FA Dahlen, and Mark A Wieczorek. Spatiospectral concentration on a sphere. *SIAM review*, 48(3):504–536, 2006.

Ilya M Sobol. Distribution of points in a cube and approximate evaluation of integrals. *USSR Computational mathematics and mathematical physics*, 7:86–112, 1967.

Jamie M Taylor and David Pardo. Stochastic quadrature rules for solving pdes using neural networks. *arXiv preprint arXiv:2504.11976*, 2025.

François Treves et al. *Analytic partial differential equations*. Springer, 2022.

Mauri J Valtonen and Hannu Karttunen. *The three-body problem*. Cambridge University Press, 2006.

Haiyong Wang and Shuhuang Xiang. On the convergence rates of legendre approximation. *Mathematics of Computation*, 81(278):861–877, 2012.

Yifan Wang, Pengzhan Jin, and Hehu Xie. Tensor neural network and its numerical integration. *arXiv preprint arXiv:2207.02754*, 2022.

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. *Journal of Scientific Computing*, 101(3):1–29, 2024.

Zixuan Wang, Qi Tang, Wei Guo, and Yingda Cheng. Sparse grid discontinuous galerkin methods for high-dimensional elliptic equations. *Journal of Computational Physics*, 314:244–263, 2016.

Jian Cheng Wong, Chin Chun Ooi, Abhishek Gupta, and Yew-Soon Ong. Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, 5(3):985–1000, 2022.

Haixu Wu, Huakun Luo, Yuezhou Ma, Jianmin Wang, and Mingsheng Long. Ropinn: Region optimized physics-informed neural networks. *arXiv preprint arXiv:2405.14369*, 2024.

Bing Yu et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.

Zhiyuan Zhao, Xueying Ding, and B Aditya Prakash. Pinnsformer: A transformer-based framework for physics-informed neural networks. *arXiv preprint arXiv:2307.11833*, 2023.

A Theoretical Section Proofs

Before stating any of the relevant theorems, it is necessary to introduce some notation. We begin by considering functions $\breve{u}$ which can be expressed as expansions of basis functions:

$$\breve{u} = \sum_{\nu \in \mathcal{F}} c_\nu \phi_\nu \tag{16}$$

where Ω_1 is the hypercube $[-1,1]^d$, $\phi_\nu : \Omega_1 \to \mathbb{R}$ are basis functions, and $c_\nu \in \mathbb{R}$ are the corresponding coefficients, with $\mathcal{F}$ being a countable index set.

We will be interested in understanding how accurately a given function u can be expressed as in Equation (16). The following definitions will prove useful in describing the quality of the basis expansion approximation.

Definition 4. *Let $(\Lambda_n)_{n\geq 1}$ be a sequence of finite subsets of the index set $\mathcal{F}$. This sequence is called an exhaustion of $\mathcal{F}$ if, for every $\nu \in \mathcal{F}$, there exists an integer n_0 such that $\nu \in \Lambda_n$ for all $n \geq n_0$.*

Definition 5. *The series in Equation (16) is said to conditionally converge to u under a given norm $\|\cdot\|$ if there exists an exhaustion $(\Lambda_n)_{n\geq 1}$ of $\mathcal{F}$ such that:*

$$\lim_{n\to\infty} \left\| u - \sum_{\nu \in \Lambda_n} c_\nu \phi_\nu \right\| = 0. \tag{17}$$

Definition 6. *The series in Equation (16) is said to unconditionally converge to u in the same norm if and only if equation equation 17 is satisfied for every exhaustion $(\Lambda_n)_{n\geq 1}$ of $\mathcal{F}$.*

Finally, we have a standard definition of ℓ^p, which we state here for the reader's convenience:

Definition 7. *For $0 < p < \infty$, the space ℓ^p is defined as the subset of $\mathbb{R}^{\mathbb{N}}$, the space of all sequences, consisting of elements $(x_n)_{n\in\mathbb{N}}$ that satisfy the condition:*

$$\sum_{n=1}^{\infty} |x_n|^p < \infty. \tag{18}$$

Armed with the previous definitions, we are ready to attack the problem of the convergence rate of approximations based on the LoRPS model class. To prove our main result, we will need three results. The first theorem is proven in Cohen & DeVore (2015), while we prove the following two lemmata.

Let us begin with the first result we will need. Theorem 3.3 from Cohen & DeVore (2015) states the following:

Theorem 8. *Let $(c_\nu)_{\nu\in\mathcal{F}}$ be an orthonormal basis of $L^2(U,\mu)$ for some given measure μ on U, and let $u \in L^2(U,V,\mu)$. Then, the inner products*

$$u_\nu := \int u(y)\phi_\nu(y)d\mu(y),\ \nu \in \mathcal{F} \tag{19}$$

are elements of V, and the series in Equation (16) converges unconditionally towards u in $L^2(U,V,\mu)$, with the error given by

$$\left\| u - \sum_{\nu\in\Lambda_n} c_\nu\phi_\nu \right\|_{L^2(U,V,\mu)} \leq \left(\sum_{\nu\notin\Lambda_n} |c_\nu|_V^2 \right)^{\frac{1}{2}}. \tag{20}$$

for any exhaustion $(\Lambda_n)_{n\geq 1}$.

In our case of a Legendre polynomial basis, we will of course apply Theorem (8) with the uniform probability measure over $\Omega_1 = [-1,1]^d$.

The second result we require is a generalization of Stechkin's Lemma; indeed, Stechkin's Lemma corresponds to the special case $q = 2$. We provide the proof in this instance.

Lemma 9. *Let $0 < p < q < \infty$ and let $(c_\nu)_{\nu\in\mathcal{F}} \in \ell^p(\mathcal{F})$ be a sequence of real numbers. Then, if Λ_n is a set of indices which corresponds to the n largest values of $|c_\nu|$, one has*

$$\left(\sum_{\nu\notin\Lambda_n} |c_\nu|^q \right)^{\frac{1}{q}} \leq (n+1)^{-\left(\frac{1}{p}-\frac{1}{q}\right)} \left(\sum_{\nu\in\mathcal{F}} |c_\nu|^p \right)^{\frac{1}{p}} \tag{21}$$

Proof. Let $(c_k)_{k\geq 1}$ be the decreasing rearrangement of the sequence $(c_\nu)_{\nu\in\mathcal{F}}$, sorted according to descending absolute value. From the definition of Λ_n, we have:

$$\sum_{\nu\notin\Lambda_n} |c_\nu|^q = \sum_{k\geq n+1} |c_k|^q \leq |c_{n+1}|^{q-p} \sum_{k\geq n+1} |c_k|^p \leq |c_{n+1}|^{q-p} \sum_{\nu\in\mathcal{F}} |c_\nu|^p \tag{22}$$

On the other hand, we also have:

$$(n+1)|c_{n+1}|^p \leq \sum_{k=1}^{n+1} |c_k|^p \leq \sum_{\nu\in\mathcal{F}} |c_\nu|^p \tag{23}$$

Now, dividing this inequality by $(n+1)$, we obtain:

$$|c_{n+1}|^p \leq \frac{\sum_{\nu\in\mathcal{F}} |c_\nu|^p}{n+1}. \tag{24}$$

Substituting this bound into the previous inequality, we have:

$$\sum_{\nu\notin\Lambda_n} |c_\nu|^q \leq \left(|c_{n+1}|^p\right)^{\frac{q-p}{p}} \sum_{\nu\in\mathcal{F}} |c_\nu|^p \leq \left(\frac{\sum_{\nu\in\mathcal{F}} |c_\nu|^p}{n+1} \right)^{\frac{q-p}{p}} \sum_{\nu\in\mathcal{F}} |c_\nu|^p \tag{25}$$

Finally, raising both sides to the power of $1/q$, we recover the desired inequality. □

To obtain concrete results concerning the summability of c_ν we take the $\{\phi_\nu\}$ to be tensor products of Legendre polynomials and we will need to define $\mathcal{F} := \mathbb{N}_0^d$ and define various relations and operations on its elements.

In each dimension, the Legendre polynomial basis we use is the standard one over $[-1, 1]$ with normalization $P_n(1) = 1$.

Definition 10. *Let α and β be elements of $\mathcal{F}$.*

(1) $|\alpha| := \sum_{i=1}^d \alpha_i$.

(2) $\alpha! := \prod_{i=1}^d \alpha_i!$.

(3) $\rho^\alpha := \prod_{i=1}^d \rho_i^{\alpha_i}$

(4) $\alpha \leq \beta := \alpha_i \leq \beta_i$ *for all* $i = 1, 2, \ldots, d$.

(5) $\alpha < \beta := \alpha \leq \beta$ *and in addition* $|\alpha| < |\beta|$.

(6) $D^\alpha := \frac{d^{|\alpha|}}{d^{\alpha_1} x_1 d^{\alpha_2} x_2 \ldots d^{\alpha_d} x_d}$.

Theorem 11. *If f is analytic inside and on a Bernstein polyellipse $\mathcal{E} := \mathcal{E}_{\rho_1} \times \mathcal{E}_{\rho_2} \times \cdots \times \mathcal{E}_{\rho_d}$ with foci ± 1 in each dimension and major semiaxis and minor semiaxis summing to $\rho_i > 1$ for every dimension $i = 1, 2, \ldots, d$, then for each ν where $\nu \in \mathcal{F}$, the coefficient c_ν of the corresponding Legendre polynomial in the expansion for f satisfies*

$$|c_\nu| \leq \sup_{z \in \mathcal{E}} |f(z)| \prod_{i=1}^d \frac{(2\nu_i + 1)\ell(\mathcal{E}_{\rho_i})}{\pi \rho_i^{\nu_i + 1}(1 - \rho_i^{-2})} \tag{26}$$

where $\ell(\mathcal{E}_{\rho_i})$ denotes the length of the circumference of $\mathcal{E}_{\rho_i}$.

Proof. This follows by a simple extension of Theorem 2.1 from Wang & Xiang (2012) to the multidimensional case via applying a multidimensional version of the Cauchy integral formula (for example, Theorem 1.3.3 from Scheidemann (2023)). □

Theorem 12. *If f is a real analytic function on an open neighborhood of Ω_1 and $\{c_\nu\}$ are the coefficients of the expansion for f in the Legendre polynomial basis $\{\phi_\nu\}$, then*

$$\left(\sum_{\nu \in \mathcal{F}} |c_\nu|^p \right)^{\frac{1}{p}} < \infty \tag{27}$$

for all $p > 0$.

Proof. By a standard complexification result for real analytic functions, such a f extends holomorphically to some open neighborhood $\mathcal{U}$ of Ω_1 in $\mathbb{C}^d$ (see for example, Krantz & Parks (1992), p. 162). Since Ω_1 is compact and is the limiting case of the family of Bernstein polyellipses $\mathcal{E} := \mathcal{E}_{\rho_1} \times \mathcal{E}_{\rho_2} \times \cdots \times \mathcal{E}_{\rho_d}$ with foci ± 1 in each dimension and major semiaxis and minor semiaxis summing to $\rho_i > 1$ for every dimension $i = 1, 2, \ldots, d$, $\mathcal{U}$ must contain a sufficiently small Bernstein polyellipse strictly contained in the neighborhood over which f is real analytic. This enables the use of Theorem 11. Absorbing all terms which do not depend on the index of

summation ν into a generic constant we have

$$
\begin{aligned}
|c_\nu| &\leq C \prod_{i=1}^{d} \frac{(2\nu_i + 1)}{\rho_i^{\nu_i}} \\
|c_\nu|^p &\leq C^p \prod_{i=1}^{d} \frac{(2\nu_i + 1)^p}{\rho_i^{p\nu_i}} \\
\sum_{\nu \in \mathcal{F}} |c_\nu|^p &\leq C^p \sum_{\nu \in \mathcal{F}} \prod_{i=1}^{d} \frac{(2\nu_i + 1)^p}{\rho_i^{p\nu_i}} \\
\sum_{\nu \in \mathcal{F}} |c_\nu|^p &\leq C^p \prod_{i=1}^{d} \sum_{j \in \mathbb{N}_0} \frac{(2j + 1)^p}{\rho_i^{pj}} \\
\sum_{\nu \in \mathcal{F}} |c_\nu|^p &\leq C^p \prod_{i=1}^{d} (1 + 3^p \mathrm{Li}_{-p}(\frac{1}{\rho_i^p})) < \infty
\end{aligned}
$$

where $\mathrm{Li}_{-s}(z)$ is the polylogarithm function. □

We are now ready to prove Theorems 2 and 3.

Proof of Theorem 2. Theorem 8 can be applied to f using the Legendre polynomial basis $\{\phi_\nu\}$ and combined with Lemma 9 where $n = s$, $p = 1$ and $q = 2$ to obtain

$$
\epsilon = \left\| u - \sum_{\nu \in \Lambda_n} c_\nu \phi_\nu \right\|_{L^2(U,V,\mu)} \leq (s+1)^{-\left(1-\frac{1}{2}\right)} \sum_{\nu \in \mathcal{F}} |c_\nu|
$$

Application of Theorem 12 then shows that there exists a constant C which does not depend on s such that

$$
\epsilon \leq C(s+1)^{-\frac{1}{2}}
$$

which can be rearranged to the desired result. □

Proof of Theorem 3. Let κ be the multiindex, all of whose terms are k. Because u is exactly represented by its Taylor series, we have

$$
\epsilon = \left\| u - \sum_{\nu \leq \kappa} c_\nu \phi_\nu \right\|_{L^2(U,V,\mu)} \leq \sum_{\nu \not\leq \kappa} \left\| \frac{D^\nu u(0)}{\nu!} x^\nu \right\|_{L^2(U,V,\mu)} \leq \sum_{|\nu| > k} \left\| \frac{D^\nu u(0)}{\nu!} x^\nu \right\|_{L^2(U,V,\mu)}
$$

and the derivatives of u can be bounded using well-known Cauchy estimates (see Scheidemann (2023), Theorem 1.3.3)

$$
|D^\nu u(z)| \leq \frac{\nu!}{r^\nu} \left\| u|_{\mathcal{D}} \right\|_\infty
$$

so

$$
\epsilon \leq K \sum_{|\nu| > k} \left| \frac{x}{r} \right|^\nu \leq K \sum_{|\nu| > k} \left(\frac{1}{r} \right)^\nu \leq K \sum_{|\nu| > k} C^{|\nu|}
$$

where $C = \max \frac{1}{r_i} < 1$. It is well-known that the number of multiindexes with total degree j is

$$
\binom{j + d - 1}{d - 1}
$$

so this gives us

$$\epsilon \leq K \sum_{j=k+1}^{\infty} \binom{j+d-1}{d-1} C^j \leq K \sum_{j=k+1}^{\infty} \frac{(j+d-1)^{(d-1)}}{(d-1)!} C^j$$

$$\epsilon \leq K_d \sum_{j=k+1}^{\infty} (j+d-1)^{(d-1)} C^j$$

$$\epsilon \leq K_{C,d} C^k (k+d)^{(d-1)} \sum_{m=1}^{\infty} (\frac{(m+k+d-1)}{k+d})^{(d-1)} C^m$$

Since

$$\frac{(m+k+d-1)}{k+d} \leq m \text{ for all } m \geq 1$$

the sum converges and is bounded above by a polylogarithm independent of k and we have

$$\epsilon = O(C^k (k+d)^{(d-1)})$$

which when inverted after taking k asymptotically large gives us the desired result. □

B Integral Equation Bias Derivation

We begin by defining the expectation and variance of the kernel-weighted solution term:

$$h_\theta(x,y) := K(x,y) u_\theta(y)$$

$$s_\theta(x) := f(x) + \int_{\mathcal{D}(x)} K(x,y) u_\theta(y)\, dy - u_\theta(x)$$

$$\begin{aligned} t_\theta(x) := \mathbb{E}_{P(y|x)} \left[h_\theta(x,y) \right] &= \mathbb{E}_{P(y|x)} \left[K(x,y) u_\theta(y) \right] \\ &= A_x^{-1} \int_{\mathcal{D}(x)} K(x,y) u_\theta(y)\, dy \end{aligned} \tag{28}$$

$$\tag{29}$$

where A_x is a normalization term converting the integral to an expectation:

$$A_x := |\mathcal{D}(x)|^{-1} = \left(\int_{\mathcal{D}(x)} 1\, d\hat{y} \right)^{-1} \tag{30}$$

Then

$$\mathbb{V}_{P(\hat{y}|\tilde{x}_i)} \left[K(\tilde{x}_i, \hat{y}) u_\theta(\hat{y}) \right] = \mathbb{E}_{P(\hat{y}|\tilde{x}_i)} \left[h_\theta(\tilde{x}_i, \hat{y})^2 \right] - \mathbb{E}_{P(\hat{y}|\tilde{x}_i)} \left[h_\theta(\tilde{x}_i, \hat{y}) \right]^2. \tag{31}$$

The true loss function is then:

$$\begin{aligned} \mathcal{L}(\theta) &= \frac{1}{N} \sum_{i=1}^{N} \left(f(\tilde{x}_i) + \int_{\mathcal{D}(\tilde{x}_i)} K(\tilde{x}_i, y) u_\theta(y)\, dy - u_\theta(\tilde{x}_i) \right)^2 \\ &= \frac{1}{N} \sum_{i=1}^{N} \left(f(\tilde{x}_i) + A_{\tilde{x}_i}^{-1} \mathbb{E}_{P(\hat{y}|\tilde{x}_i)} \left[h_\theta(\tilde{x}_i, \hat{y}) \right] - u_\theta(\tilde{x}_i) \right)^2. \end{aligned} \tag{32}$$

The corresponding Monte Carlo loss approximation is:

$$\widehat{\mathcal{L}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{P(\hat{y}_{1:M}|\tilde{x}_i)}\left[\left(f(\tilde{x}_i) + \frac{A_{\tilde{x}_i}^{-1}}{M}\sum_{j=1}^{M} K(\tilde{x}_i,\hat{y}_j)u_\theta(\hat{y}_j) - u_\theta(\tilde{x}_i)\right)^2\right]. \tag{33}$$

To analyze the bias introduced by this approximation, we add and subtract $A(\tilde{x}_i)^{-1}t(\tilde{x}_i)$ inside the square:

$$\begin{aligned}\widehat{\mathcal{L}}(\theta) &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{P(\hat{y}_{1:M}|\tilde{x}_i)}\left[\left(f(\tilde{x}_i) + \frac{A_{\tilde{x}_i}^{-1}}{M}\sum_{j=1}^{M} h_\theta(\tilde{x}_i,\hat{y}_j) - u_\theta(\tilde{x}_i)\right)^2\right] \\ &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{P(\hat{y}_{1:M}|\tilde{x}_i)}\left[\left(s_\theta(\tilde{x}_i) + A_{\tilde{x}_i}^{-1}\left(\frac{1}{M}\sum_{j=1}^{M} h_\theta(\tilde{x}_i,\hat{y}_j) - t(\tilde{x}_i)\right)\right)^2\right].\end{aligned}$$

Note that

$$\mathbb{E}_{P(\hat{y}_{1:M}|\tilde{x}_i)}\left[\frac{1}{M}\sum_{j=1}^{M} h_\theta(\tilde{x}_i,\hat{y}_j)\right] = t(\tilde{x}_i) \tag{34}$$

which means when we expand the square the cross-product term's expectation is zero. Therefore,

$$\begin{aligned}\widehat{\mathcal{L}}(\theta) &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{P(\hat{y}_{1:M}|\tilde{x}_i)}\left[s_\theta(\tilde{x}_i)^2 + A_{\tilde{x}_i}^{-2}\left(\frac{1}{M}\sum_{j=1}^{M} h_\theta(\tilde{x}_i,\hat{y}_j) - t_\theta(\tilde{x}_i)\right)^2\right] \\ &= \mathcal{L}(\theta) + \frac{1}{N}\sum_{i=1}^{N} A_{\tilde{x}_i}^{-2}\mathbb{E}_{P(\hat{y}_{1:M}|\tilde{x}_i)}\left[\left(\frac{1}{M}\sum_{j=1}^{M} h_\theta(\tilde{x}_i,\hat{y}_j) - t_\theta(\tilde{x}_i)\right)^2\right] \\ &= \mathcal{L}(\theta) + \frac{1}{N}\sum_{i=1}^{N} \frac{A_{\tilde{x}_i}^{-2}}{M}\,\mathbb{V}_{P(y|\tilde{x}_i)}\left[h(\tilde{x}_i,y)\right]\end{aligned} \tag{35}$$

where we used the fact that the variance of a Monte Carlo estimator based on i.i.d. samples is equal to the variance of the integrand divided by the number of samples M.

C Experimental Details

In this section, we provide additional experimental details and numerical results to supplement those presented in the main paper. Specifically, we report the root mean squared error (RMSE) as a measure of model performance, along with details on the number of collocation points utilized during training. To verify the robustness of our results, we carried out five independent trials, each with a different seed (seeds 0-4). Optimization was performed using the Adam optimizer for the first 1000 steps, after which we switched to the more robust LBFGS optimizer for the remainder of the training.

C.1 Convolution Equation

Convolution-type integral equations play a central role in many areas of applied mathematics and physics, including image processing, population dynamics, and quantum mechanics. These equations often involve kernels that capture localized or long-range interactions between points in a domain. Their mathematical

structure makes them ideal benchmarks for evaluating the performance and generalization capabilities of numerical solvers, especially in high dimensions. We evaluate the performance of LoRPS and baseline methods on the following convolution equation:

$$u(x) - \frac{1}{d}\int_{\mathcal{D}} \left(\sum_{i=1}^{d} e^{-\frac{(x_i-y_i)^2}{\sigma^2}}\right) u(y)\,dy = \exp\left(\frac{\|x\|^2}{\sigma^2}\right) - \frac{1}{d}\cdot C^{d-1}\cdot \sum_{i=1}^{d}\left[e^{-\frac{x_i^2}{\sigma^2}}\cdot \frac{\sigma^2 \sinh\left(\frac{x_i}{\sigma^2}\right)}{x_i}\right], \tag{36}$$

where the constant is given by:

$$C = \sigma\sqrt{\pi}\cdot \operatorname{erfi}\left(\frac{1}{2\sigma}\right), \tag{37}$$

and the domain, parameters and dimension are:

$$\sigma = 2, \quad \mathcal{D} = \Omega = \left[-\frac{1}{2}, \frac{1}{2}\right]^d, \quad d = 20 \tag{38}$$

The exact solution to this equation is known analytically:

$$u(x) = \exp\left(\frac{\|x\|^2}{\sigma^2}\right) \tag{39}$$

C.2 Heat with Memory Equation

Volterra-type heat equations extend the classical heat equation by introducing time-dependent memory effects, typically modeled through integral terms that accumulate the history of spatial diffusion. These equations arise in materials with thermal memory, viscoelasticity, and anomalous transport, and they pose a significant challenge for numerical solvers due to their integro-differential structure. We consider the following memory-augmented heat equation:

$$\frac{\partial u}{\partial t}(x,t) = \int_0^t \Delta u(x,s)\,ds + f(x,t), \tag{40}$$

defined on the space-time domain:

$$x \in \Omega = [0,1]^d, \quad t \in [0,T], \tag{41}$$

with homogeneous Dirichlet boundary conditions:

$$u(x,t) = 0 \quad \text{for } x \in \partial([0,1]^d), \text{ and all } t \in [0,T], \tag{42}$$

and zero initial condition:

$$u(x,0) = 0. \tag{43}$$

The exact solution is given by:

$$u(x,t) = \left(\prod_{j=1}^{d} \sin(\pi x_j)\right) t^2, \tag{44}$$

which satisfies the boundary and initial conditions by construction. The forcing term $f(x,t)$ is derived analytically from the exact solution and has the form:

$$f(x,t) = \left(2t + \frac{\pi^2 d}{3} t^3\right) \cdot \prod_{j=1}^{d} \sin(\pi x_j). \tag{45}$$

In our experiments, we set $d = 19$ and $T = 1$. This problem provides a smooth, factorized, and fully explicit benchmark that can be used to evaluate the accuracy and efficiency of numerical solvers for time-dependent integro-differential equations in high dimensions.

C.3 Non-Local Diffusion Equation

Non-local diffusion equations generalize classical diffusion models by incorporating integral terms that capture interactions over a finite or infinite range, rather than only local gradients. These equations are important in modeling phenomena where spatially distributed effects play a key role, such as anomalous transport, image inpainting, peridynamics, and collective dynamics in biological and ecological systems. The presence of both differential and integral operators makes them a valuable testbed for evaluating the flexibility and expressiveness of numerical solvers. We consider the following non-local diffusion equation:

$$\Delta u(x) + \int_{\mathcal{D}} \left(\sum_{j=1}^{d} x_j^2 \cdot \prod_{i=1}^{d} \cos\left(\frac{\pi}{2} y_i\right) \right) u(y)\, dy = -\left(\frac{\pi^2}{4} \cdot d \cdot u(x) \right) + \sum_{j=1}^{d} x_j^2, \tag{46}$$

with domain:

$$\mathcal{D} = \Omega = [-1, 1]^d, \tag{47}$$

and homogeneous Dirichlet boundary conditions:

$$u(x) = 0 \quad \text{on } \partial([-1, 1]^d). \tag{48}$$

The exact solution is given by:

$$u(x) = \prod_{j=1}^{d} \cos\left(\frac{\pi}{2} x_j\right), \tag{49}$$

which satisfies the boundary conditions and enables exact computation of residuals. In our experiments, we use dimension $d = 20$.

C.4 Non-Polynomial Terms

As discussed in Section 4.3, when the governing equation contains *non-polynomial* components, the LoRPS closure properties no longer apply directly and the loss cannot be evaluated in closed form. We address this by a *projection* step: each non-polynomial term is approximated by an *auxiliary* LoRPS model trained with standard regression on Monte Carlo samples from the domain. We then substitute the auxiliary approximation back into the equation, which converts the full problem into a polynomial form and restores exact analytic evaluation. Since this step is approximate, it can become a dominant error source, and we therefore quantify its impact.

We conduct an ablation study on the heat-with-memory equation, focusing on the non-polynomial forcing term given by a product of sine functions. The ablation controls the quality of the projected term and measures the downstream effect on the final IE solution. Concretely, we first fit an auxiliary LoRPS model to the forcing term and use its output as the projected input to the *main* LoRPS IE solver. Because the target term is separable, we fix the auxiliary rank to $s = 1$ and sweep only the polynomial degree k, which directly determines the approximation capacity of the projection model.

We can see in table 5 a clear coupling between *projection error* and *solution error*: coarse projections (large auxiliary relative L_2 error) lead to large errors in the IE solution, whereas improving the projection rapidly reduces the final relative L_2 error. Notably, the projection error decreases primarily at odd degrees (yielding a "paired" trend), which is expected since the sine function is odd and is therefore better captured as odd powers are added. Overall, increasing k improves projection fidelity and correspondingly reduces solution error, with the largest gains up to $k \approx 7$. Beyond this point, performance largely saturates, suggesting that once the forcing term is approximated sufficiently well, the remaining error is dominated by other factors such as optimization difficulty rather than by the projection itself.

The results in table 2 further indicate that the projection step is what allows LoRPS to operate outside purely polynomial regimes: as soon as the non-polynomial components are captured with even moderate fidelity, the overall solution accuracy stays high and the projection no longer constitutes the bottleneck.

Table 5: Heat-with-memory hyperparameter sweep: auxiliary projection relative L_2 error and resulting IE solution relative L_2 error versus polynomial degree k (rank fixed to $s = 1$).

(s, k)	Projection rel. L_2 error	Heat-with-memory rel. L_2
$(1, 1)$	4.7×10^{-1}	9.9×10^{-1}
$(1, 2)$	4.6×10^{-1}	9.9×10^{-1}
$(1, 3)$	4.3×10^{-2}	1.3×10^{-1}
$(1, 4)$	4.3×10^{-2}	1.4×10^{-1}
$(1, 5)$	1.3×10^{-3}	2.7×10^{-3}
$(1, 6)$	1.3×10^{-3}	3.1×10^{-3}
$(1, 7)$	2.08×10^{-5}	1.3×10^{-4}
$(1, 8)$	2.08×10^{-5}	9.7×10^{-5}
$(1, 9)$	2.194×10^{-7}	1.07×10^{-4}
$(1, 10)$	7.904×10^{-7}	1.4×10^{-4}

C.5 Baselines Implementation

To ensure a rigorous and meaningful comparison, we implemented all baseline methods by closely adhering to the official or publicly available code provided by the original authors. Whenever applicable, we reproduced the exact model architectures as specified in the corresponding publications. Hyperparameter values were also adopted directly from the original works to maintain consistency (see Table 6). Furthermore, the number of collocation points used for each baseline method is detailed in Table 7. By carefully replicating these implementations, we aim to attribute any observed performance differences solely to methodological distinctions, rather than discrepancies in experimental design or tuning.

Table 6: Model hyperparameters and their corresponding parameter counts used for all equations. In the value and parameters columns, each entry is presented in the format $a/b/c$, where a, b, and c correspond to the convolution, non-local diffusion, and heat equations, respectively. Values were adopted from previous works, with reductions applied only when necessary to meet memory constraints.

Model	**Hyperparameter**	**Value**	**Parameters**
PINNs	Hidden layers	4 / 4 / 4	135k / 135k / 9.7k
	Hidden size	256 / 256 / 64	
FLS	Hidden layers	4 / 4 / 4	135k / 135k / 9.7k
	Hidden size	256 / 256 / 64	
QRes	Hidden layers	4 / 4 / 4	401k / 401k / 7.7k
	Hidden size	256 / 256 / 32	
PINNsFormer	# of encoders	1 / 1 / 1	170k / 170k / 142k
	# of decoders	1 / 1 / 1	
	Embedding size	16 / 16 / 8	
	Heads	2 / 2 / 2	
	Hidden size	64 / 64 / 16	
LoRPS	k	8 / 8 / 8	3.2k / 3.2k / 3.2k
	s	20 / 20 / 20	

Table 7: The number of collocation points used with each baselines for each equation.

Method	**Convolution**	**Nonlocal Diffusion**	**Heat with Memory**
PINN	15k	10k	1k
FLS	15k	10k	1k
QRes	15k	9k	1k
PINNsFormer	12k	3k	0.1k
LoRPS	-	-	-

C.6 Additional Results

In this section, we present results for additional neural network architectures that were not included in the main text. These experiments provide a broader comparison across baseline models and sampling methods.

Table 8: Performance comparison of Monte Carlo-based methods across different values of α, which controls the variance in Equation 7, measured in terms of relative L^2 error.

Arch.	**Sampling**	$\alpha = 1$	$\alpha = 25$	$\alpha = 50$	$\alpha = 75$
QRes	Uniform	$6.51e{-3} \pm 3.49e{-3}$	$1.53e{-1} \pm 1.66e{-2}$	$2.62e{-1} \pm 7.53e{-2}$	$4.14e{-1} \pm 1.36e{-1}$
	Sobol	$1.56e{-3} \pm 1.14e{-5}$	$3.76e{-2} \pm 6.62e{-4}$	$7.34e{-2} \pm 2.71e{-3}$	$9.46e{-2} \pm 2.23e{-2}$
	Double	$6.29e{-3} \pm 5.06e{-3}$	$1.65e{-1} \pm 1.44e{-1}$	$1.17e{+0} \pm 1.94e{+0}$	$1.13e{+0} \pm 1.33e{+0}$
PINNsFormer	Uniform	$7.23e{-3} \pm 2.95e{-3}$	$9.84e{-1} \pm 9.33e{-3}$	$9.92e{-1} \pm 8.74e{-3}$	$9.99e{-1} \pm 4.16e{-3}$
	Sobol	$1.61e{-3} \pm 1.18e{-5}$	$3.81e{-2} \pm 4.23e{-5}$	$7.38e{-2} \pm 3.81e{-4}$	$1.18e{-1} \pm 1.89e{-2}$
	Double	$7.72e{-3} \pm 1.64e{-3}$	$6.11e{+0} \pm 1.04e{+1}$	$3.63e{+0} \pm 6.06e{+0}$	$5.31e{+0} \pm 3.84e{+0}$
FLS	Uniform	$5.37e{-3} \pm 2.80e{-3}$	$9.84e{-1} \pm 9.68e{-3}$	$9.98e{-1} \pm 4.56e{-3}$	$9.99e{-1} \pm 3.35e{-3}$
	Sobol	$1.61e{-3} \pm 2.11e{-5}$	$3.81e{-2} \pm 1.23e{-4}$	$7.32e{-2} \pm 3.30e{-4}$	$1.10e{-1} \pm 2.78e{-3}$
	Double	$4.89e{-3} \pm 5.93e{-4}$	$4.56e{-1} \pm 1.80e{-1}$	$1.30e{+0} \pm 7.42e{-1}$	$5.09e{+13} \pm 1.02e{+14}$
PINN	Uniform	$5.00e{-3} \pm 1.68e{-3}$	$9.86e{-1} \pm 4.68e{-3}$	$1.00e{+0} \pm 6.20e{-3}$	$1.00e{+0} \pm 1.50e{-3}$
	Sobol	$1.60e{-3} \pm 6.02e{-6}$	$3.80e{-2} \pm 1.05e{-4}$	$7.37e{-2} \pm 5.49e{-4}$	$1.06e{-1} \pm 1.38e{-3}$
	Double	$4.93e{-3} \pm 1.09e{-3}$	$4.49e{+0} \pm 7.66e{+0}$	$1.01e{+0} \pm 1.11e{-1}$	$1.42e{+0} \pm 1.39e{+0}$
LoRPS	Exact	$\mathbf{1.13e{-4} \pm 1.18e{-4}}$	$\mathbf{1.49e{-4} \pm 9.29e{-5}}$	$\mathbf{3.44e{-4} \pm 2.95e{-4}}$	$\mathbf{9.08e{-4} \pm 8.99e{-4}}$