SetPINNs: Set-based Physics-Informed Neural Networks

Mayank Nagda
RPTU University
Kaiserslautern-Landau
Germany

Phil Ostheimer
RPTU University
Kaiserslautern-Landau
Germany

Thomas Specht
RPTU University
Kaiserslautern-Landau
Germany

Frank Rhein
Karlsruhe
Institute of Technology
Germany

Fabian Jirasek
RPTU University
Kaiserslautern-Landau
Germany

Stephan Mandt
University of
California, Irvine
USA

Marius Kloft
RPTU University
Kaiserslautern-Landau
Germany

Sophie Fellenz
RPTU University
Kaiserslautern-Landau
Germany

Abstract

Physics-Informed Neural Networks (PINNs) solve partial differential equations using deep learning. However, conventional PINNs perform pointwise predictions that neglect dependencies within a domain, which may result in suboptimal solutions. We introduce SetPINNs, a framework that effectively captures local dependencies. With a finite element-inspired sampling scheme, we partition the domain into sets to model local dependencies while simultaneously enforcing physical laws. We provide a rigorous theoretical analysis showing that SetPINNs yield unbiased, lower-variance estimates of residual energy and its gradients, ensuring improved domain coverage and reduced residual error. Extensive experiments on synthetic and real-world tasks show improved accuracy, efficiency, and robustness. Code available at `https://github.com/mayanknagda/set\_based\_pinns`

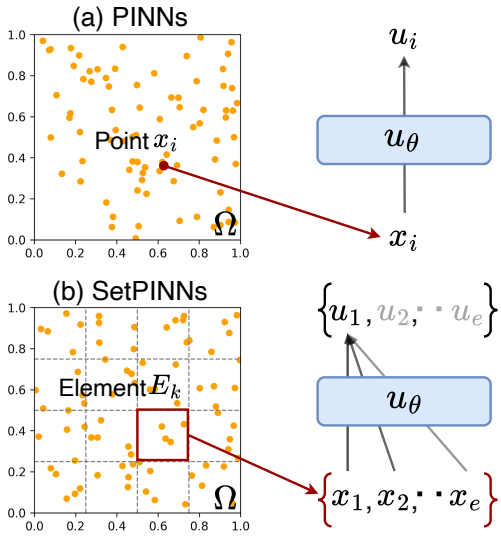

Figure 1: Comparison of (a) standard PINNs and (b) SetPINNs. Standard PINNs predict $u_\theta(x_i)$ independently at uniformly sampled points, often missing domain structure. SetPINNs partition the domain into elements and jointly predict on local sets, ensuring better coverage, stronger inductive bias, and improved physical consistency.

1 INTRODUCTION

Modeling complex physical systems by solving partial differential equations (PDEs) is a central challenge across science and engineering. Deep learning approaches, especially PINNs (Raissi et al., 2019), have emerged as a promising paradigm for directly integrating physical laws (PDEs) into the learning process. PINNs have been successfully applied to many physical systems, including fluid mechanics (Cai et al., 2021) and climate forecasting (Verma et al., 2024), among others.

Conventional PINNs and variants employ a multilayer perceptron (MLP) and perform pointwise prediction over the domain. Whilst remarkably successful in var-

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

ious scenarios, recent research has shown that PINNs may fail in scenarios where PDE solutions exhibit high-frequency or multiscale features (Fuks and Tchelepi, 2020; Leiteritz and Pflüger, 2021; Krishnapriyan et al., 2021; Mojgani et al., 2022; Daw et al., 2022; Wang et al., 2022; Zhao et al., 2024; Wu et al., 2024). In such cases, the optimization process tends to get stuck in local minima, resulting in overly smooth suboptimal approximations of ground-truth solutions.

To mitigate these *failure modes*, researchers have proposed data interpolation techniques and tailored training strategies (Krishnapriyan et al., 2021; Wang et al., 2021a). Research has recently focused on exploiting inherent domain dependencies (such as temporal or regional correlations) (Wu et al., 2024; Zhao et al., 2024). Although these methods show promise, they typically capture only a subset of domain dependencies (e.g., temporal but not spatial), leaving local interactions unmodeled and thereby limiting generalization and robustness across diverse PDE regimes.

Orthogonal to existing approaches, we explicitly model local dependencies, an aspect often overlooked in prior research. As shown in Figure 1, conventional PINNs operate at isolated points, predicting each location independently. This leads to two key limitations: (1) uniform sampling can leave large portions of the domain unsupervised, and (2) the model cannot exploit correlations among neighboring points. We hypothesize that these shortcomings contribute to instability and poor generalization.

To overcome these challenges, we propose SetPINNs—a new paradigm for physics-informed learning that is domain-agnostic and explicitly incorporates local dependencies. Inspired by finite element methods, SetPINNs partition the domain into multiple elements and draw sets of collocation points from each element. These sets are then processed by an attention-based architecture, enabling the framework to capture local dependencies effectively. Unlike traditional PINNs, SetPINNs are also well-suited for permutation-invariant systems, where modeling interactions among neighboring points is natural, broadening their applicability across diverse physical problems. In summary, our key contributions are as follows:

- **Novel Framework.** We introduce SetPINNs, a set-based approach that models local dependencies by processing collocation point sets drawn from localized regions of the domain.
- **Theoretical Guarantees.** We provide a rigorous analysis showing that SetPINNs yield unbiased, lower-variance estimates of PDE residuals and their gradients, ensuring improved domain coverage and more stable optimization.
- **Empirical Validation.** Across diverse PDEs and real-world tasks, SetPINNs reduce errors, converge reliably, and show lower variance than state-of-the-art models, demonstrating superior accuracy, efficiency, and robustness.

2 RELATED WORK

Physics-Informed Neural Networks. Neural networks for solving PDEs have been studied for decades (Lagaris et al., 1998; Psichogios and Ungar, 1992), but advances in deep learning led to a resurgence through PINNs (Raissi et al., 2019). PINNs incorporate the PDE residual into the loss, penalizing violations by enforcing constraints at collocation points. They have been applied in both *white-box* settings, which rely only on collocation points, and *grey-box* settings, which combine collocation points with data. Applications include cardiovascular simulation (Raissi et al., 2020), fluid mechanics (Cai et al., 2021), and climate forecasting (Verma et al., 2024), among others. Extensions include architectural innovations (Bu and Karpatne, 2021; Wong et al., 2022), optimization techniques (Wang et al., 2022; Wu et al., 2024), regularization methods (Krishnapriyan et al., 2021), and advanced sampling strategies (Wu et al., 2023), all improving accuracy and applicability.

Failure Modes of PINNs. Conventional PINNs, typically built on MLPs, make pointwise predictions and face difficulties in complex PDE regimes. These challenges stem from gradient stiffness due to multi-scale interactions, which restrict learning rates and hinder convergence (Wang et al., 2021b). Studies have shown that PINNs often fail for PDEs with high-frequency or multiscale features (Raissi, 2018; Fuks and Tchelepi, 2020; Krishnapriyan et al., 2021; Wang et al., 2022; Leiteritz and Pflüger, 2021; Zhao et al., 2024), leading to local minima and overly smooth, physically inaccurate solutions.

Mitigating Failure Modes. Approaches to address these limitations include training strategies, data interpolation, architectural changes, and domain-aware modeling. Seq2Seq training (Krishnapriyan et al., 2021) and related strategies (Mao et al., 2020; Wang et al., 2021a, 2022) are often costly and unstable. Data-driven methods (Han et al., 2018; Zhu et al., 2019; Chen et al., 2021) require real or synthetic data, which may be scarce. Architectural variants such as QRes (Bu and Karpatne, 2021) and FLS (Wong et al., 2022) improve convergence but face scalability issues. In domain-specific settings HANNA enforces hard thermodynamic consistency for activity-coefficient prediction (Specht et al., 2024). Domain-

aware approaches like FBPINNs (Moseley et al., 2023), PINNsFormer (Zhao et al., 2024), and RoPINNs (Wu et al., 2024) capture only subsets of domain structure, e.g., temporal without spatial or regional without intra-region interactions. In contrast, SetPINNs directly learn local dependencies from point sets via attention, providing greater flexibility, improved domain coverage, and better accuracy without manual decomposition. Concurrently, finite-element-augmented PINN approaches have been explored: FE-PINNs (Sunil and Sills, 2026) use finite-element surrogate modeling, and Finite-PINN (Li et al., 2025) encodes geometric structure via Laplace–Beltrami eigenfunctions. These methods address the pointwise limitation from a mesh-based perspective, whereas SetPINNs operate directly on point sets without requiring basis-function construction.

Deep Learning on Sets. Permutation-invariant models such as Deep Sets (Zaheer et al., 2017; Wagstaff et al., 2022) and Set Transformers (Lee et al., 2019) enable learning from unordered inputs and have advanced applications in chemistry, vision, and particle physics (Serviansky et al., 2020; Zhang et al., 2023). Despite the prevalence of set-structured data, their use in physics-informed learning remains limited. SetPINNs bridge this gap by integrating set-based modeling with physical constraints, enabling accurate, flexible, and domain-consistent learning.

3 METHODOLOGY

This section presents preliminaries in Section 3.1 and the proposed SetPINNs approach in Section 3.2.

3.1 Preliminaries

We briefly review PINNs and FEMs, highlighting the core principles and connection to SetPINNs.

Problem Setting and PINNs. We consider a differential equation defined over a domain $\Omega \subset \mathbb{R}^d$, with solution $u : \mathbb{R}^d \to \mathbb{R}^m$. The domain's interior, boundary, and optionally an initial subset are denoted by Ω, $\partial\Omega$, and Ω_0, respectively. Differential operators $\mathcal{O}_\Omega$, $\mathcal{O}_{\partial\Omega}$, and $\mathcal{O}_{\Omega_0}$ encode the governing equation, boundary conditions (BCs), and initial conditions (ICs). For example, the heat equation is written as $\mathcal{O}_\Omega(u)(x) = u_t - u_{xx}$. The complete problem formulation is:

$$\begin{aligned} &\mathcal{O}_\Omega(u)(x) = 0, x \in \Omega; \qquad \mathcal{O}_{\Omega_0}(u)(x) = 0, x \in \Omega_0; \\ &\mathcal{O}_{\partial\Omega}(u)(x) = 0, x \in \partial\Omega, \end{aligned} \tag{1}$$

PINNs (Raissi et al., 2019) approximate the solution u with a neural network u_θ, trained to minimize the residuals of the governing constraints:

$$\mathcal{L}(u_\theta) = \sum_{X \in \{\Omega, \Omega_0, \partial\Omega\}} \frac{\lambda_X}{N_X} \sum_{i=1}^{N_X} \left\| \mathcal{O}_X(u_\theta)(x_X^{(i)}) \right\|^2, \tag{2}$$

where N_X is the number of collocation points in region X, and λ_X weights each term. Additional data-based terms may be added, as discussed in Appendix B.1.

Finite Element Methods (FEMs). FEMs (Zienkiewicz et al., 2005) solve PDEs by partitioning the domain Ω into N non-overlapping elements (e.g., triangles or quadrilaterals) and approximating the solution as a linear combination of local basis functions: $u_h(x) = \sum_{i=1}^{N} U_i \phi_i(x)$, where $\{\phi_i(x)\}$ are element-wise basis functions and U_i are coefficients. A weighted residual approach is used to derive a global system of algebraic equations, which is then solved for an approximate solution. FEMs exhibit two important properties: (1) *local dependency*, as neighboring points are coupled through overlapping basis support and (2) *global consistency* through inter-element coupling. By explicitly modeling local interactions, FEMs have shown superior accuracy over PINNs in certain complex scenarios (Grossmann et al., 2024). However, PINNs remain attractive for their flexibility, lower complexity, and ability to incorporate real-world data. With SetPINNs, we combine the flexibility of PINNs with the locality principles of FEMs by extending pointwise predictions to element-based predictions.

3.2 SetPINNs

This section presents the SetPINNs framework. We begin by introducing the element-aware sampling strategy that ensures better domain coverage and forms point sets, then describe the architecture (Figure 2) to model local dependencies, and finally derive the set-based physics loss.

3.2.1 Domain Partitioning and Sampling

The domain is partitioned into non-overlapping elements, following the standard FEM approach.

Definition 3.1 (Domain Partitioning)**.** *Let $\Omega \subset \mathbb{R}^d$ be an open, bounded domain. We partition Ω into non-overlapping subdomains (elements) $\{E_k\}_{k=1}^K$ such that*

$$\bigcup_{k=1}^{K} E_k = \Omega, \quad E_i \cap E_j = \varnothing \text{ for } i \neq j.$$

We assume that the partition is quasi-uniform, i.e., $|E_k| \approx h^d$ for some $h > 0$.

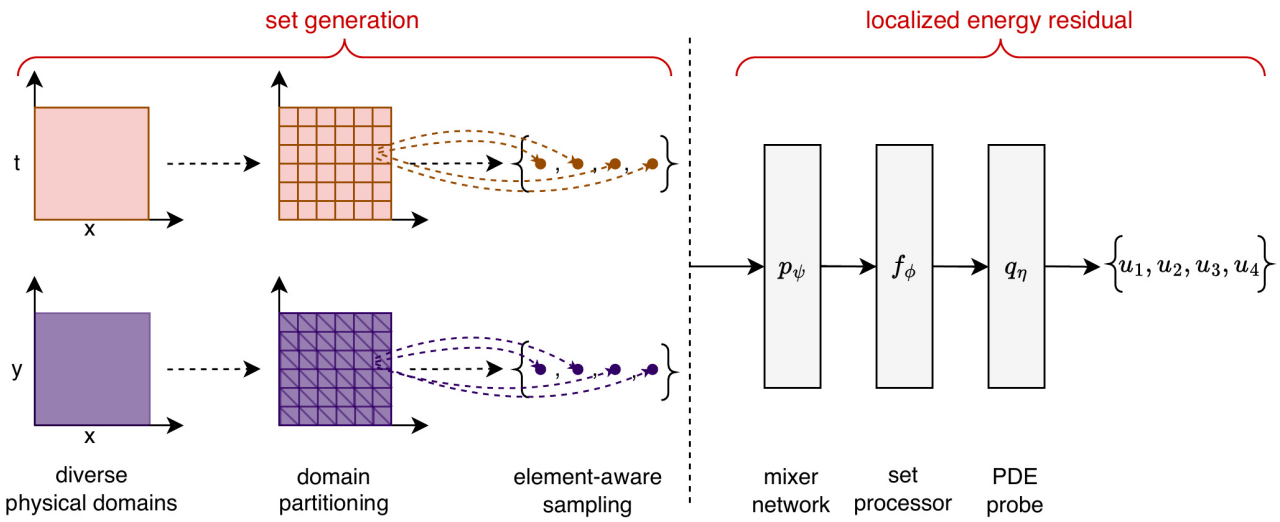

Figure 2: Overview of the SetPINNs framework. On the left, we exemplify the set generation process for diverse physical domains. Sets are formed from neighboring points in the domain. Each set is then processed through a Mixer Network, Set Processor, and PDE Probe to predict the solution for each element in the set. The parameters of SetPINNs are learned using a localized energy residual.

After partitioning, collocation points are sampled within each element.

Definition 3.2 (Element-Aware Sampling (EAS))**.** *Within each element E_k, we uniformly sample m_k collocation points $\{x_{k,i}\}_{i=1}^{m_k} \subset E_k$. Let $M = \sum_{k=1}^{K} m_k$ denote the total number of collocation points. We require that the sampling density is uniform across elements, i.e.,*

$$\frac{m_k}{|E_k|} = \frac{m_j}{|E_j|} \quad \text{for all } k, j,$$

which guarantees that each subdomain is neither over- nor under-sampled relative to its size.

The element-aware sampling strategy offers two key advantages: it ensures better domain coverage and produces localized point sets. These sets enable the architecture to model local dependencies.

3.2.2 Architecture

The SetPINN architecture is designed to capture local dependencies and make joint predictions over multiple points within each element. As shown in Figure 2, each element E_k yields a set $\mathcal{S}_k$ of collocation points, which is passed through three main components: a Mixer Network, an attention-based Set Processor, and a PDE Probe. We now describe each of these in detail.

Set Generator. Given the domain partitioning (Def. 3.1) and element-aware sampling (Def. 3.2), each element E_k yields a set of collocation points that represent the subdomain Ω_k. The set generator collects these points into a set $\mathcal{S}_k$, defined as:

$$\Omega_k \xrightarrow{\text{set-generator}} \mathcal{S}_k := \{x_k^1, x_k^2, \ldots, x_k^{m_k}\},$$

where each $x_k^i \in \Omega_k$. Predicting the PDE solution within Ω_k is thus formulated as a set-to-set learning task.

Mixer Network. Each point $x_k^i \in \mathcal{S}_k$ lies in a low-dimensional coordinate space $\mathbb{R}^d$, which does not adequately capture expressive patterns for PDE approximation. To address this, a pointwise embedding network p_ψ maps each input into a higher-dimensional representation space $\mathbb{R}^r$, producing a transformed set $\mathcal{M}_k = \{\mathcal{M}_k^i\}_{i=1}^{m_k}$, where $\mathcal{M}_k^i \in \mathbb{R}^r$. This embedding increases representational capacity and prepares the set for downstream interaction modeling via attention.

Set Processor. To model interactions within each set $\mathcal{M}_k$, we apply a permutation-equivariant attention module f_ϕ, implemented using a Transformer encoder block (Vaswani et al., 2017). For each point i, the processor aggregates contextual information from all other points in the set using:

$$\mathcal{O}_k^i = \text{MLP}\left(\sum_{j=1}^{m_k} \alpha(x_k^i, x_k^j) \cdot W_V \mathcal{M}_k^j\right), \text{for } i = 1, .., m_k,$$

where $\alpha(x_k^i, x_k^j)$ are attention weights computed via scaled dot-product similarity between learned embed-

dings of $\mathcal{M}_k^i$ and $\mathcal{M}_k^j$, and $W_V \in \mathbb{R}^{r\times r}$ is a learnable weight matrix. The MLP is shared across all points, ensuring permutation equivariance.

Unlike sequence-based models, we avoid positional encodings to maintain equivariance under permutations of the input set, an essential property for consistent set-based learning.

PDE Probe. The output features $\mathcal{O}_k = \{\mathcal{O}_k^i\}_{i=1}^{m_k}$, $\mathcal{O}_k^i \in \mathbb{R}^r$ are decoded into the final PDE solution predictions $u_k = \{u_k^i\}_{i=1}^{m_k}$, $u_k^i \in \mathbb{R}^c$ through a pointwise decoder q_η. This module consists of a shallow MLP applied independently to each $\mathcal{O}_k^i$, mapping the feature space $\mathbb{R}^r$ to the solution space $\mathbb{R}^c$. As each prediction is conditioned on a context-aware representation, the model produces locally consistent and structure-informed approximations of the solution within each element.

3.2.3 Learning Objective

Conventional PINNs enforce PDE constraints at individual collocation points, treating all locations uniformly. In contrast, SetPINN aggregates residuals within each element, promoting local physical consistency while preserving global balance.

For any constraint region $X \in \{\Omega, \Omega_0, \partial\Omega\}$, we define the *localized residual energy* over an element E_k as:

$$\begin{aligned}\mathcal{E}_X(E_k, u_\theta) &:= \int_{E_k} \left\|\mathcal{O}_X(u_\theta)(x)\right\|^2 dx \\ &\approx \frac{|E_k|}{m_k}\sum_{i=1}^{m_k}\left\|\mathcal{O}_X(u_\theta)(x_k^{(i)})\right\|^2,\end{aligned} \tag{3}$$

where the integral is approximated using m_k collocation points within E_k.

The total SetPINNs loss is then the average localized energy across all elements:

$$\mathcal{L}_{\text{SetPINNs}} = \sum_{X\in\{\Omega,\,\Omega_0,\,\partial\Omega\}} \frac{\lambda_X}{K}\sum_{k=1}^{K}\mathcal{E}_X(E_k, u_\theta), \tag{4}$$

where $\theta = \{\psi, \phi, \eta\}$ denotes the learnable parameters of the Mixer Network p_ψ, Set Processor f_ϕ, and PDE Probe q_η, which collectively define the function u_θ used to predict the solution. Training follows standard PINN protocols and is detailed in Appendix B.2 for completeness.

4 THEORETICAL ANALYSIS

We present a theoretical analysis showing that compared to pointwise PINNs with global uniform sampling (GUS), SetPINNs with element-aware sampling (EAS) yield unbiased, lower-variance estimates of the residual energy and its gradients. These results imply improved domain coverage and more stable optimization signals, which are essential for effective training in physics-informed models.

Assumptions. Our results assume (A1) a quasi-uniform partition (Def. 3.1), (A2) uniform within-element sampling with proportional density (Def. 3.2), and (A3) independent draws across elements/iterations; full statements and proofs appear in the Appendix A.

We now quantify coverage via a simple coverage ratio, aligning with residual-based error estimation ideas in numerical analysis (Quarteroni and Valli, 2008).

Definition 4.1 (Coverage Ratio)**.** *Let u_θ be a neural network with residual $R_\theta(x) = \mathcal{O}_\Omega(u_\theta)(x)$. The coverage ratio is defined as the ratio $\frac{\hat{I}}{I}$ of the true residual energy $I = \int_\Omega |R_\theta(x)|^2\,dx$ to its discrete estimate $\hat{I}$, where $\hat{I}_{\text{GUS}} = \frac{|\Omega|}{M}\sum_{j=1}^{M}|R_\theta(x_j)|^2$ and $\hat{I}_{\text{EAS}} = \sum_{k=1}^{K}\frac{|E_k|}{m_k}\sum_{i=1}^{m_k}|R_\theta(x_{k,i})|^2$ are the estimates under Global Uniform Sampling (GUS) and Element-Aware Sampling (EAS), respectively.*

A coverage ratio close to 1 suggests the sampled points effectively capture the magnitude and distribution of the residual. A much larger value suggests the sampling strategy underestimates the true residual integral, often due to missing regions of high residual.

We first establish that both EAS and GUS provide unbiased estimates of the residual energy.

Lemma 4.2 (Unbiased Estimation via Sampling)**.** *Under uniform random sampling, where points are drawn within each element E_k for EAS and over the entire domain Ω for GUS, the estimators $\hat{I}_{\text{EAS}}$ and $\hat{I}_{\text{GUS}}$ (as defined in Definition 4.1) are unbiased estimators of the true integral $\int_\Omega |R_\theta(x)|^2 dx$.*

Proof Sketch. Both $\hat{I}_{\text{GUS}}$ and $\hat{I}_{\text{EAS}}$ are scaled sums of samples. For GUS, sampling is from $\mathcal{U}(\Omega)$; for EAS, sampling is from $\mathcal{U}(E_k)$ within each element. By linearity of expectation, both estimators yield $\int_\Omega |R_\theta(x)|^2 dx$ as their expected value. Complete proof is available in Appendix A.2. □

We now show that EAS yields a statistically more reliable estimator of the residual energy.

Theorem 4.3 (Variance Reduction with EAS)**.** *Under the same assumptions as in Lemma 4.2, EAS provides an estimator with equal or lower variance:* $\text{Var}(\hat{I}_{\text{EAS}}) \le \text{Var}(\hat{I}_{\text{GUS}})$.

Proof Sketch. Calculation of variances shows $\text{Var}(\hat{I}_{\text{EAS}}) \le \text{Var}(\hat{I}_{\text{GUS}})$ reduces to proving

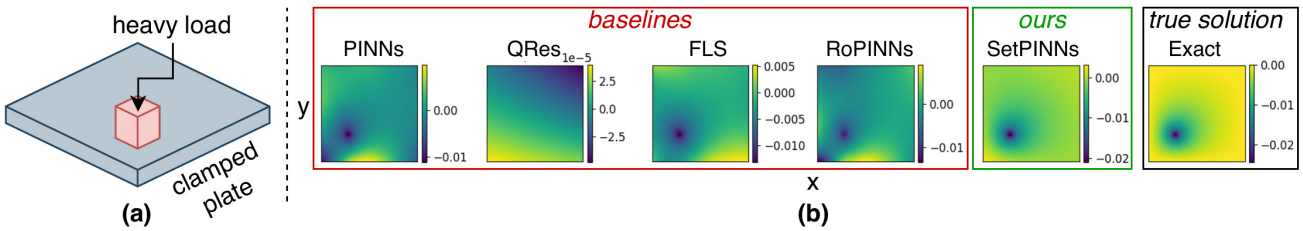

Figure 3: Setup and performance comparison for a clamped plate with a localized heavy load. (a) Illustration of the plate and load placement, representing a common structural engineering scenario. (b) Comparison of SetPINNs and baseline methods against the true solution, highlighting SetPINNs' ability to capture high-stress regions due to element-aware sampling and a set-based architecture.

$\sum_{k=1}^{K} \frac{1}{|E_k|} \left(\int_{E_k} |R_\theta(x)|^2 \, dx \right)^2 \geq \frac{1}{|\Omega|} \left(\int_\Omega |R_\theta(x)|^2 \, dx \right)^2$, which is true in stratified sampling (Caflisch, 1998). Full proof in Appendix A.2. □

Since both $\hat{I}_{\text{EAS}}$ and $\hat{I}_{\text{GUS}}$ are unbiased estimators of the true total PDE residual energy $I(\theta)$ (Lemma 4.2), the lower variance of $\hat{I}_{\text{EAS}}$ (Theorem 4.3) directly implies a lower Mean Squared Error for the estimator $\hat{I}_{\text{EAS}}$ (i.e., $\mathbb{E}[(\hat{I}_{\text{EAS}} - I(\theta))^2]$ is smaller). Thus, EAS yields a statistically more reliable estimate of $I(\theta)$, suggesting a coverage ratio $C_{\text{EAS}}(\theta)$ closer to 1. This fundamental advantage in estimation quality extends to the gradients crucial for optimization.

Next, we analyze the variance of the stochastic gradients used during training. The following theorem shows that EAS leads to a reduced gradient variance compared to GUS.

Theorem 4.4 (Gradient Variance Reduction with EAS)**.** *Let $G_{EAS}(\theta) = \nabla_\theta \hat{I}_{EAS}(\theta)$ and $G_{GUS}(\theta) = \nabla_\theta \hat{I}_{GUS}(\theta)$ be the stochastic gradient estimators of $\nabla_\theta I(\theta)$ derived from EAS and GUS based loss formulations, respectively. Under the same assumptions as Theorem 4.3, EAS yields a gradient estimator with lower (or equal) total variance:* $\text{Tr}(\text{Cov}(G_{\text{EAS}}(\theta))) \leq \text{Tr}(\text{Cov}(G_{\text{GUS}}(\theta)))$.

Proof Sketch. The variance of each gradient component estimator under EAS and GUS is calculated. The comparison hinges on the inequality $\sum_{k=1}^{K} |E_k| \mu_{g_p,k}^2 \geq |\Omega| \mu_{g_p}^2$ (where $\mu_{g_p,k}$ and μ_{g_p} are mean gradient components in element E_k and domain Ω, respectively), which follows from Jensen's inequality. This extends to all components, and thus the trace. (Complete proof in Appendix A.3). □

Lower gradient variance under EAS theoretically explains the smoother loss and more stable optimization we observe empirically.

Connection to Set-based Processing: Our theoretical analysis (Theorems 4.3 and 4.4) reveals that EAS yields lower-variance estimates for both the PDE residual energy and, critically, its gradients compared to global uniform sampling. SetPINNs are architecturally designed to exploit this advantage. The prospect of reduced gradient variance suggests a more stable optimization signal (Bottou et al., 2018), which can lead to more efficient and reliable convergence. By processing element-wise point sets where EAS ensures more reliable local physical information (a consequence of Theorem 4.3), the SetPINN's attention mechanism is poised to more effectively model local dependencies. This synergy provides a strong basis for SetPINNs to achieve accurate robust performance, which we will evaluate empirically in Section 5.

5 EXPERIMENTS

To empirically demonstrate the effectiveness of SetPINNs, we evaluate its performance against baseline methods on both standard PDE benchmarks (Section 5.1) and real-world tasks (Section 5.2).

Baselines. We compare SetPINNs against several strong baselines, including standard MLP-based PINNs (Raissi et al., 2019), First-Layer Sine networks (FLS) (Wong et al., 2022), Quadratic Residual Networks (QRes) (Bu and Karpatne, 2021), FBPINNs (Moseley et al., 2023), PINNsFormer (Zhao et al., 2024), and RoPINNs (Wu et al., 2024), covering recent advances in both architectural design and domain-aware modeling. All models are trained with approximately equal parameter counts to ensure fair comparison. Performance is evaluated using the relative Root Mean Squared Error (rRMSE) (Wu et al., 2024), a standard metric for physics-informed regression.

Implementation Details. For baseline models, we use the authors' official open-source implementations. SetPINNs employ a simple encoder-only Transformer (Vaswani et al., 2017) to capture local set-level dependencies, trained using the objective in Equation 4 with

Table 1: SetPINNs as a robust and accurate method across white-box and grey-box settings. rRMSE scores are reported for each task and model. Best results are **bold**; second-best are underlined.

Task	**SetPINNs (ours)**	PINNs (JCP'19)	QRes (ICDM'21)	FLS (TAI'22)	FBPINNs (ACM'23)	PINNsFormer (ICLR'24)	RoPINNs (NeurIPS'24)
White-box PDE Benchmarks							
1D-Wave	**0.078**	0.414	0.527	0.119	0.139	0.363	0.172
1D-Reaction	0.061	0.979	0.983	0.039	0.057	0.030	**0.017**
Convection	**0.031**	0.899	0.925	0.197	0.836	0.522	0.720
3D-Helmholtz	**0.072**	0.281	0.217	0.189	0.238	N/A	0.132
Navier-Stokes	**0.218**	5.703	2.235	3.016	0.952	0.841	0.935
Harmonic	**0.025**	0.342	0.162	0.123	0.117	0.109	0.092
Plate	**0.324**	1.467	1.003	1.234	1.692	N/A	1.375
Grey-box Benchmarks							
Act. Coeff.	**0.090**	0.158	0.169	0.152	N/A	N/A	N/A
Agg. Break.	**0.220**	0.451	0.512	0.451	N/A	N/A	N/A

weighting $\lambda_X = 1$. All models are trained on NVIDIA A100 GPUs. All methods are trained with equal collocation budget (M). Training proceeds in two stages: Adam (Kingma and Ba, 2014) for initial optimization, followed by L-BFGS (Schmidt, 2005) for fine-tuning. Complete training and hyperparameter details are provided in Appendix E.1. SetPINNs maintain practical time/memory scaling by keeping element set sizes small ($m_k \ll M$); runtime/memory details are summarized in Appendix D/E.3.

5.1 White-box PDE Benchmarks

Setup. We test SetPINNs on diverse white-box PDE benchmarks: 1D reaction-diffusion, 1D wave, convection, Navier-Stokes, harmonic Poisson, a clamped plate, and the 3D Helmholtz equation. These benchmarks present unique modeling challenges, including sharp gradients (reaction, wave), false diffusion (convection), coupled nonlinearities (Navier-Stokes), high-frequency oscillations (Helmholtz), and boundary sensitivity (harmonic). For 2D tasks, the domain Ω is discretized into a 50×50 grid and partitioned into 625 elements of size 2×2. For the 3D Helmholtz equation, the domain is discretized into a $50 \times 50 \times 50$ grid and partitioned into 25^3 cubic elements of size $2 \times 2 \times 2$. SetPINNs sample 4 points per element for set-based inputs; baselines use the same total points via global uniform sampling (2500 points). All models use identical data splits (denser test sets) and are trained purely from physics constraints (PDEs, ICs/BCs). PINNsFormer results are N/A for two equations due to the absence of a temporal dimension. PDE setup and failure modes are detailed in Appendix C.

Qualitative Evaluation. A particularly challenging case is the clamped plate scenario from structural engineering, where a pinned plate is subjected to a concentrated load at its center (Timoshenko et al., 1959; Harris and Sabnis, 1999). This setup induces highly localized stress, making it a strong test of a model's ability to resolve sharp spatial features. Figure 3 shows that standard PINNs and other baselines struggle in the high-stress region as well as boundary due to sparse or uneven supervision, whereas SetPINNs capture the stress concentration more precisely. This improvement arises from two key aspects: first, element-aware sampling prevents under-sampling in critical zones by ensuring uniform representation across the domain; second, the set-based architecture allows the model to capture local PDE structure more faithfully. This aligns with our theoretical findings that element-aware sampling reduces estimator variance, validating its practical advantage in resolving sharp localized features.

Quantitative Evaluation. In Table 1 we report the relative root mean squared error (rRMSE). SetPINNs consistently outperform baseline methods across most tasks, achieving lower errors. Improvements are particularly pronounced on tasks that involve localized features, sharp gradients, or under-resolved dynamics, demonstrating the robustness and accuracy of the SetPINN framework in solving challenging physical systems. Appendix E.4 reports averaged results over ten runs, with variance and two-tailed t-tests, further confirming SetPINNs' lower variance and higher accuracy.

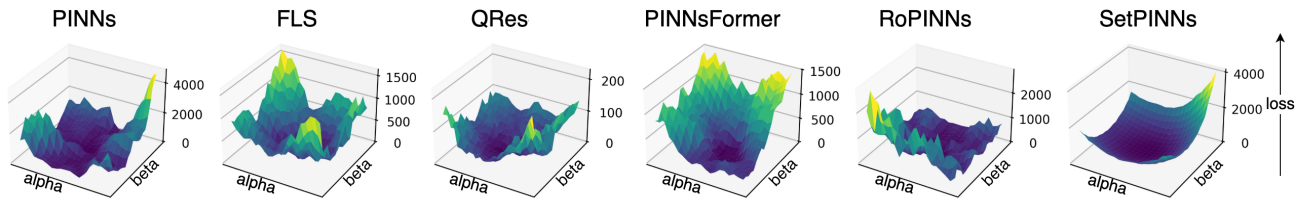

Figure 4: Loss landscape of SetPINNs compared to baselines. Baseline models exhibit sharp cones in their loss landscape, whereas SetPINNs demonstrate a significantly smoother loss surface. This indicates that SetPINNs optimize more stably, reducing sensitivity to failure modes.

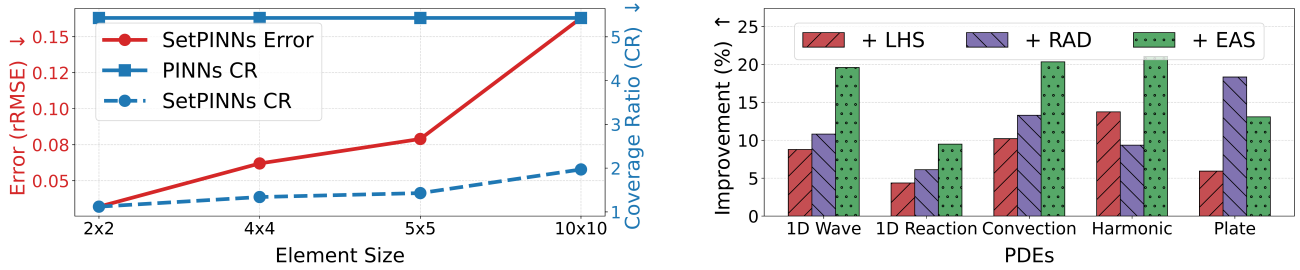

Figure 5: SetPINN design and sampling analysis. Left: Error and Coverage Ratio (CR) vs. element size; smaller elements reduce intra-set distances and improve both metrics. Right: Relative rRMSE gains of vanilla PINNs with advanced sampling (EAS, LHS, RAD) over uniform sampling, with EAS providing the largest benefits.

5.2 Grey-box (Real World) Chemical Process Engineering Tasks

Setup. We address two chemical process engineering tasks: predicting activity coefficients (AC) and modeling agglomerate breakage (AB). Both tasks use sparse, real-world data that irregularly cover the domain, and both are permutation-invariant systems regarding their constituent molecules or particles.

For AC prediction, thermodynamic consistency requires the Gibbs–Duhem relation, which couples the activity coefficients of both components across composition $x_1 \in [0, 1]$:

$$x_1 \left(\frac{\partial \ln \gamma_1}{\partial x_1} \right)_T + (1 - x_1) \left(\frac{\partial \ln \gamma_2}{\partial x_1} \right)_T = 0. \quad (5)$$

Experimental AC data are obtained from the Dortmund Data Bank (DDB) (DDB, 2023). For AB, the population balance equation governs the temporal evolution of the particle-size distribution $u(x, t)$:

$$\frac{\partial u(x,t)}{\partial t} = \int_x^\infty f(x, x')\, r(x')\, u(x', t)\, dx' - r(x)\, u(x, t), \quad (6)$$

where r is the breakage rate and f is the daughter-size distribution kernel. Further details on data preparation, featurization, and train/test splits are provided in Appendix F.

Standard PINNs require fixed-size, ordered inputs and therefore struggle with variable-cardinality, unordered sets. SetPINNs handle this naturally via permutation-equivariant set processing. FBPINNs, PINNsFormer, and RoPINNs are also N/A for grey-box tasks, as their designs impose structural assumptions that do not transfer to these settings: FBPINNs rely on fixed spatial domain decomposition that does not accommodate the variable-cardinality, permutation-invariant inputs of chemical systems; PINNsFormer requires a temporal pseudo-sequence structure absent in steady-state mixture problems; and RoPINNs apply rotational position embeddings that assume a fixed spatiotemporal grid, which is incompatible with the irregular, composition-space domains of these tasks.

Results. Table 1 shows SetPINNs achieve the lowest rRMSE and variance on both tasks. For AC, SetPINNs (0.090 ± 0.00) enforce Gibbs–Duhem and outperform UNIFAC (Fredenslund et al., 1975) (0.166). This highlights SetPINNs' advantage in learning from sparse, irregular real-world data in permutation-invariant chemical systems while respecting physical laws.

6 DISCUSSION

We analyze the factors behind SetPINNs' improved performance through additional experiments and highlight limitations and future directions.

Smoother Loss Landscape and Stable Optimization. SetPINNs exhibit smoother loss landscapes (Figure 4) when visualized along dominant Hessian directions (Li et al., 2018; Krishnapriyan et al., 2021). This characteristic suggests enhanced optimization stability, facilitating better gradient flow and more consistent convergence. Theoretically, this is supported by our variance reduction result (Theorem 4.4), which shows that EAS reduces the total variance of gradient estimates. Such smoothness can act as an implicit regularizer, potentially reducing sensitivity to hyperparameter choices.

Impact of Element Size and EAS. Domain partitioning is central to the SetPINNs framework. To assess its effect, we evaluate how varying element sizes influence prediction accuracy on the reaction equation. For this analysis, the domain is discretized into a 100×100 grid and partitioned into $n \times n$ square elements, where each element forms a point set. We sample $m_k = n^2$ points from each element to preserve the proportional allocation. As shown in Figure 5 (left), larger elements result in higher errors due to greater inter-point distances, which weaken local interactions. In contrast, smaller elements better preserve locality, leading to improved accuracy.

We also study the effectiveness of EAS in Figure 5 (right), comparing relative rRMSE improvements of vanilla PINNs with various strategies: Latin Hypercube Sampling (LHS), Residual-based Adaptive Distribution (RAD) (Wu et al., 2023), and EAS, over uniform sampling. LHS is a widely used stratified method, while RAD adaptively targets high-residual regions. Among these, EAS achieves the best performance gain, underscoring its contribution to improved coverage. Importantly, however, even with advanced sampling, baselines still fall short of SetPINNs (shown in Appendix G). While sampling enhances coverage, the larger gains stem from the set-based architecture itself, which explicitly models local dependencies within each element. Additional ablation results and hyperparameter guidance are provided in Appendix G. To further disentangle the contributions of sampling and architecture, we note that Table 8 shows EAS improves vanilla PINNs by approximately 17% on average, while the full SetPINNs framework achieves approximately 78% improvement, confirming that the set-based architecture is the primary driver of performance gains.

Limitations and Future Directions. Our theoretical analysis assumes uniform within-element sampling and proportional allocation. While experiments already include irregular domains, exploring adaptive or non-uniform allocation strategies could yield further improvements. SetPINNs introduce hyperparameters tied to domain partitioning (element number and size) and the set architecture (Transformer depth, attention heads). Our ablations (Appendix G) show robust trends across reasonable ranges, but optimal configurations may be PDE-dependent. Additionally, while SetPINNs' runtime is comparable to recent baselines such as RoPINNs, the set-based architecture incurs overhead relative to plain MLPs (Appendix E.3), which may matter in low-complexity settings. Finally, although our complexity analysis (Appendix D) shows that per-point cost is independent of domain dimensionality, and we validate on 3D problems, future work could investigate more efficient set processing for very high-dimensional domains. An interesting future direction is also to combine local set-based processing with autoregressive physics-informed modeling for time-dependent PDEs, as explored in recent work on PIANO (Nagda et al., 2025).

7 CONCLUSION

By integrating element-aware sampling and set-based learning, SetPINNs offer a principled way to capture local PDE features. Theoretical analysis demonstrates that this combination reduces both estimator and gradient variance, leading to improved residual coverage and more stable optimization. Across a wide range of experiments, the method demonstrates superior accuracy and stability over conventional PINNs and state-of-the-art variants. In doing so, it presents a compelling direction for the next generation of physics-informed neural networks, suggesting that explicitly modeling local dependencies in the domain can yield substantial gains in both performance and reliability.

ACKNOWLEDGEMENT

The main part of this work was conducted within the DFG Research Unit FOR 5359 (ID 459419731) on Deep Learning on Sparse Chemical Process Data. SF and MK further acknowledge support by the DFG through TRR 375 (ID 511263698), SPP 2298 (ID 441826958), and SPP 2331 (ID 441958259), by the Carl-Zeiss Foundation through the initiatives AI-Care and Process Engineering 4.0, and by the BMFTR award 01IS24071A. FJ gratefully acknowledges financial support by the DFG through the Emmy-Noether program (grant 528649696). SM acknowledges funding from the National Science Foundation (NSF) through NSF CAREER Award IIS-2047418, IIS2007719, the NSF LEAP Center, and the Hasso Plattner Research Center at UCI.

References

(2023). Dortmund data bank.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311.

Bu, J. and Karpatne, A. (2021). 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)*, pages 675–683. SIAM.

Caflisch, R. E. (1998). Monte carlo and quasi-monte carlo methods. *Acta numerica*, 7:1–49.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. (2021). Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738.

Chen, Z., Liu, Y., and Sun, H. (2021). Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1):6136.

Daw, A., Bu, J., Wang, S., Perdikaris, P., and Karpatne, A. (2022). Mitigating propagation failures in physics-informed neural networks using retain-resample-release (r3) sampling. *arXiv preprint arXiv:2207.02338*.

Fredenslund, A., Jones, R. L., and Prausnitz, J. M. (1975). Group-contribution estimation of activity coefficients in nonideal liquid mixtures. *AIChE Journal*, 21(6):1086–1099.

Fuks, O. and Tchelepi, H. A. (2020). Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1).

Grossmann, T. G., Komorowska, U. J., Latz, J., and Schönlieb, C.-B. (2024). Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics*, 89(1):143–174.

Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510.

Harris, H. G. and Sabnis, G. (1999). *Structural modeling and experimental techniques*. CRC press.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR.

Leiteritz, R. and Pflüger, D. (2021). How to avoid trivial solutions in physics-informed neural networks. *arXiv preprint arXiv:2112.05620*.

Li, H., Miao, Y., Khodaei, Z. S., and Aliabadi, M. (2025). Finite-pinn: A physics-informed neural network with finite geometric encoding for solid mechanics. *Journal of the Mechanics and Physics of Solids*, 203:106222.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31.

Mao, Z., Jagtap, A. D., and Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789.

Mojgani, R., Balajewicz, M., and Hassanzadeh, P. (2022). Lagrangian pinns: A causality-conforming solution to failure modes of physics-informed neural networks. *arXiv preprint arXiv:2205.02902*.

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

Nagda, M., Abijuru, J., Ostheimer, P., Mandt, S., Kloft, M., and Fellenz, S. (2025). Piano: Physics-informed autoregressive network. *arXiv preprint arXiv:2508.16235*.

Psichogios, D. C. and Ungar, L. H. (1992). A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511.

Quarteroni, A. and Valli, A. (2008). *Numerical approximation of partial differential equations*, volume 23. Springer Science & Business Media.

Raissi, M. (2018). Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19(25):1–24.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differen-

tial equations. *Journal of Computational physics*, 378:686–707.

Raissi, M., Yazdani, A., and Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030.

Schmidt, M. (2005). minfunc: unconstrained differentiable multivariate optimization in matlab. *Software available at http://www. cs. ubc. ca/˜ schmidtm/-Software/minFunc. htm*.

Serviansky, H., Segol, N., Shlomi, J., Cranmer, K., Gross, E., Maron, H., and Lipman, Y. (2020). Set2graph: Learning graphs from sets. *Advances in Neural Information Processing Systems*, 33:22080–22091.

Specht, T., Nagda, M., Fellenz, S., Mandt, S., Hasse, H., and Jirasek, F. (2024). Hanna: hard-constraint neural network for consistent activity coefficient prediction. *Chemical science*, 15(47):19777–19786.

Sunil, P. and Sills, R. B. (2026). Fe-pinns: Finite-element-based physics-informed neural networks for surrogate modeling. *APL Machine Learning*, 4(1):016106.

Timoshenko, S., Woinowsky-Krieger, S., et al. (1959). *Theory of plates and shells*, volume 2. McGraw-hill New York.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Verma, Y., Heinonen, M., and Garg, V. (2024). ClimODE: Climate forecasting with physics-informed neural ODEs. In *The Twelfth International Conference on Learning Representations*.

Wagstaff, E., Fuchs, F. B., Engelcke, M., Osborne, M. A., and Posner, I. (2022). Universal approximation of functions on sets. *Journal of Machine Learning Research*, 23(151):1–56.

Wang, S., Teng, Y., and Perdikaris, P. (2021a). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081.

Wang, S., Teng, Y., and Perdikaris, P. (2021b). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*. Publisher: Society for Industrial and Applied Mathematics.

Wang, S., Yu, X., and Perdikaris, P. (2022). When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768.

Wong, J. C., Ooi, C. C., Gupta, A., and Ong, Y.-S. (2022). Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, 5(3):985–1000.

Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671.

Wu, H., Luo, H., Ma, Y., Wang, J., and Long, M. (2024). Ropinn: Region optimized physics-informed neural networks. In *Advances in Neural Information Processing Systems*.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. *Advances in neural information processing systems*, 30.

Zhang, H., Chen, J., Rondinelli, J. M., and Chen, W. (2023). Molsets: Molecular graph deep sets learning for mixture property modeling. *arXiv preprint arXiv:2312.16473*.

Zhao, Z., Ding, X., and Prakash, B. A. (2024). PINNsformer: A transformer-based framework for physics-informed neural networks. In *The Twelfth International Conference on Learning Representations*.

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., and Perdikaris, P. (2019). Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81.

Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z. (2005). *The finite element method: its basis and fundamentals*. Elsevier.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] (See Sections 3.1–3.2 with Definitions 3.1–3.2 and Eqs. 1–4.)
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] (Variance/gradient variance analysis in Section 4; complexity and scalability in Appendix D.)
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] (Provided in the supplementary materials; implementation details and dependencies referenced in Appendix E, Reproducibility.)

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] (Assumptions listed explicitly in Appendix A, "Assumptions".)
   (b) Complete proofs of all theoretical results. [Yes] (Proofs for Lemma 4.2, Theorems 4.3 and 4.4 in Appendix A.)
   (c) Clear explanations of any assumptions. [Yes] (Explained in Section 4 and clarified in Appendix A.)

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] (Code in supplementary; PDE setups and real-data details in Appendices C, F.1, F.2.)
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] (Training protocol in Appendix B; hyperparameters and splits in Appendix E.1.)
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] (rRMSE defined in Eq. 18; results reported as mean±std over 10 runs; significance noted in Appendix E.4.)
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] (see Implementation Details in Section 5, Runtime Analysis (FLOPS and wall-clock) in Appendix E.3, and Computational Complexity discussed in Appendix D.)

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] (Baselines and datasets cited, e.g., DDB (DDB, 2023), UNIFAC (Fredenslund et al., 1975), prior PINN variants.)
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes] (New implementation/code released in the supplementary materials.)
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

A Detailed Theoretical Analysis

This appendix provides complete proofs for the results in Section 4, showing that Element-Aware Sampling (EAS) yields unbiased and lower-variance estimators than Global Uniform Sampling (GUS) for the total squared PDE residual, and that its stochastic gradients have reduced variance.

A.1 Assumptions

We work under the following assumptions regarding the domain, sampling, and moments. Items (A1)–(A5) mirror the main text; (A6) is used only for the gradient analysis.

Assumption A.1. ***Domain Partition.*** *$\Omega \subset \mathbb{R}^d$ is partitioned into K measurable, pairwise disjoint elements $\{E_k\}_{k=1}^K$ with $\bigcup_{k=1}^K E_k = \Omega$.*

Assumption A.2. ***Element-Aware Sampling (EAS).*** *For each k, sample m_k points $\{x_{k,i}\}_{i=1}^{m_k}$ i.i.d. from $U(E_k)$.*

Assumption A.3. ***Global Uniform Sampling (GUS).*** *Sample M points $\{x_j\}_{j=1}^M$ i.i.d. from $U(\Omega)$.*

Assumption A.4. ***Proportional Allocation.*** *Unless stated otherwise, $m_k = M\,|E_k|/|\Omega|$ with $M = \sum_k m_k$.*

Assumption A.5. ***Residual Integrability.*** *Let $\phi(x) = |R_\theta(x)|^2$. Then $\int_\Omega \phi(x)\,dx < \infty$ and $\int_\Omega \phi(x)^2\,dx < \infty$ (equivalently, $|R_\theta|^4$ integrable).*

Assumption A.6. ***Gradient Moments and Interchange.*** *Let $g(x;\theta) = \nabla_\theta \phi(x;\theta)$. Then $\mathbb{E}\|g(x;\theta)\|_2^2 < \infty$ under the relevant sampling measure(s), and differentiation commutes with expectation: $\nabla_\theta \mathbb{E}[\phi(x;\theta)] = \mathbb{E}[g(x;\theta)]$.*

Notation. Let

$$I \;=\; \int_\Omega \phi(x)\,dx, \qquad I_k \;=\; \int_{E_k} \phi(x)\,dx, \qquad \mu_k \;=\; \frac{I_k}{|E_k|}.$$

The Monte Carlo estimators are

$$\hat{I}_{\mathrm{GUS}} \;=\; \frac{|\Omega|}{M}\sum_{j=1}^{M}\phi(x_j), \qquad \hat{I}_{\mathrm{EAS}} \;=\; \sum_{k=1}^{K}\frac{|E_k|}{m_k}\sum_{i=1}^{m_k}\phi(x_{k,i}). \tag{7}$$

Under Assumption A.4, $\hat{I}_{\mathrm{EAS}} = \frac{|\Omega|}{M}\sum_{k=1}^K\sum_{i=1}^{m_k}\phi(x_{k,i})$.

A.2 Unbiasedness and Coverage

Lemma A.7 (Unbiased Estimation)**.** *(Lemma 4.2) Under Assumptions A.1 to A.3 and A.5, both estimators in Eq. (7) are unbiased: $\mathbb{E}[\hat{I}_{\mathrm{GUS}}] = I$ and $\mathbb{E}[\hat{I}_{\mathrm{EAS}}] = I$.*

Proof. For GUS, $x \sim U(\Omega)$ yields $\mathbb{E}[\phi(x)] = \frac{1}{|\Omega|}\int_\Omega \phi(x)\,dx = I/|\Omega|$ by Assumption A.5. Linearity gives

$$\mathbb{E}[\hat{I}_{\mathrm{GUS}}] = \frac{|\Omega|}{M}\sum_{j=1}^{M}\mathbb{E}[\phi(x_j)] = \frac{|\Omega|}{M}\cdot M\cdot\frac{I}{|\Omega|} = I.$$

For EAS, $x \sim U(E_k)$ yields $\mathbb{E}[\phi(x)] = I_k/|E_k| = \mu_k$. Independence and linearity give

$$\mathbb{E}[\hat{I}_{\mathrm{EAS}}] = \sum_{k=1}^{K}\frac{|E_k|}{m_k}\sum_{i=1}^{m_k}\mathbb{E}[\phi(x_{k,i})] = \sum_{k=1}^{K}\frac{|E_k|}{m_k}\cdot m_k\cdot\mu_k = \sum_{k=1}^{K} I_k = I,$$

using Assumption A.1 in the last equality. No proportionality (Assumption A.4) is required for unbiasedness. □

Theorem A.8 (Variance Reduction)**.** *(Theorem 4.3) Under Assumptions A.1 to A.5,*

$$\mathrm{Var}(\hat{I}_{\mathrm{GUS}}) = \frac{|\Omega|}{M}\int_\Omega \phi(x)^2\,dx \;-\; \frac{1}{M}\,I^2, \tag{8}$$

$$\mathrm{Var}(\hat{I}_{\mathrm{EAS}}) = \frac{|\Omega|}{M}\int_\Omega \phi(x)^2\,dx \;-\; \frac{|\Omega|}{M}\sum_{k=1}^{K}\frac{I_k^2}{|E_k|} \;\le\; \mathrm{Var}(\hat{I}_{\mathrm{GUS}}). \tag{9}$$

Equality holds iff all elementwise averages are equal, i.e., $\mu_k = \mu_\ell$ for all k, ℓ.

Proof. **GUS:** By independence,

$$\mathrm{Var}(\hat{I}_{\mathrm{GUS}}) = \left(\frac{|\Omega|}{M}\right)^2 \sum_{j=1}^{M}\mathrm{Var}(\phi(x_j)) = \frac{|\Omega|^2}{M}\left(\mathbb{E}[\phi(x)^2] - \mathbb{E}[\phi(x)]^2\right),$$

where $x \sim U(\Omega)$. Using $\mathbb{E}[\phi(x)^2] = \frac{1}{|\Omega|}\int_\Omega \phi^2$ and $\mathbb{E}[\phi(x)] = I/|\Omega|$ (by Assumption A.5) yields Eq. (8).

EAS: Let $Y_k = \sum_{i=1}^{m_k}\phi(x_{k,i})$. Samples across elements are independent, so

$$\mathrm{Var}(\hat{I}_{\mathrm{EAS}}) = \mathrm{Var}\left(\frac{|\Omega|}{M}\sum_{k=1}^{K} Y_k\right) = \left(\frac{|\Omega|}{M}\right)^2 \sum_{k=1}^{K}\mathrm{Var}(Y_k).$$

Within E_k, the $x_{k,i}$ are i.i.d. with variance $\mathrm{Var}(\phi(x_{k,1})) = \frac{1}{|E_k|}\int_{E_k}\phi^2 - \mu_k^2$, hence $\mathrm{Var}(Y_k) = m_k\,\mathrm{Var}(\phi(x_{k,1})) = m_k\Big(\frac{1}{|E_k|}\int_{E_k}\phi^2 - \mu_k^2\Big)$. With Assumption A.4, $m_k = M|E_k|/|\Omega|$, whence

$$\begin{aligned}\mathrm{Var}(\hat{I}_{\mathrm{EAS}}) &= \left(\frac{|\Omega|}{M}\right)^2 \sum_{k=1}^{K}\left(M\frac{|E_k|}{|\Omega|}\right)\left(\frac{1}{|E_k|}\int_{E_k}\phi^2 - \mu_k^2\right)\\ &= \frac{|\Omega|}{M}\sum_{k=1}^{K}\left(\int_{E_k}\phi^2 - |E_k|\mu_k^2\right) = \frac{|\Omega|}{M}\int_\Omega\phi^2 - \frac{|\Omega|}{M}\sum_{k=1}^{K}\frac{I_k^2}{|E_k|},\end{aligned}$$

which is Eq. (9).

Comparison and equality condition: Subtract Eq. (9) from Eq. (8); the common $\frac{|\Omega|}{M}\int_\Omega\phi^2$ cancels, yielding the equivalent inequality

$$\sum_{k=1}^{K}\frac{I_k^2}{|E_k|} \;\ge\; \frac{I^2}{|\Omega|}.$$

Let $w_k = |E_k|$. By Cauchy–Schwarz with vectors $a_k = \sqrt{w_k}$ and $b_k = I_k/\sqrt{w_k}$,

$$\Big(\sum_k I_k\Big)^2 = \Big(\sum_k a_k b_k\Big)^2 \le \Big(\sum_k a_k^2\Big)\Big(\sum_k b_k^2\Big) = \Big(\sum_k w_k\Big)\Big(\sum_k \tfrac{I_k^2}{w_k}\Big),$$

which rearranges to the desired inequality. Equality in Cauchy–Schwarz occurs iff $b_k = c\,a_k$, i.e., $I_k/w_k = c$ for a constant c, equivalently μ_k is constant across k. □

Corollary A.9 (Reliability of Coverage)**.** *Let $C_{\mathrm{GUS}} = I/\hat{I}_{\mathrm{GUS}}$ and $C_{\mathrm{EAS}} = I/\hat{I}_{\mathrm{EAS}}$ (Def. 4.1). Then C_{EAS} has smaller (asymptotic) variance and smaller first-order MSE than C_{GUS}:*

$$\mathrm{Var}(C_{\mathrm{EAS}}) \;\approx\; \frac{\mathrm{Var}(\hat{I}_{\mathrm{EAS}})}{I^2} \;\le\; \frac{\mathrm{Var}(\hat{I}_{\mathrm{GUS}})}{I^2} \;\approx\; \mathrm{Var}(C_{\mathrm{GUS}}),$$

and $\mathrm{MSE}(C_{\mathrm{EAS}}) \le \mathrm{MSE}(C_{\mathrm{GUS}})$ at first order. Moreover, by Jensen, $\mathbb{E}[C_{\mathrm{GUS}}] \ge 1$ and $\mathbb{E}[C_{\mathrm{EAS}}] \ge 1$.

Proof. By Lemma 4.2, $\mathbb{E}[\hat{I}_{\mathrm{EAS}}] = \mathbb{E}[\hat{I}_{\mathrm{GUS}}] = I$. Applying the delta method to $h(z) = I/z$ at $z = I$ gives $\mathrm{Var}(h(\hat{I})) \approx h'(I)^2\mathrm{Var}(\hat{I}) = \mathrm{Var}(\hat{I})/I^2$, and $\mathrm{Bias}(h(\hat{I})) \approx \frac{1}{2}h''(I)\mathrm{Var}(\hat{I}) = \mathrm{Var}(\hat{I})/I^2$. Since $\mathrm{Var}(\hat{I}_{\mathrm{EAS}}) \le \mathrm{Var}(\hat{I}_{\mathrm{GUS}})$ by Thm. 4.3, both the variance and the first-order MSE of C_{EAS} are no larger. Jensen's inequality for the convex map $z \mapsto I/z$ yields $\mathbb{E}[C] \ge I/\mathbb{E}[\hat{I}] = 1$ for both sampling schemes. □

A.3 Comparative Statistical Analysis of SetPINN (EAS) and PINN (GUS) Losses

Let u_θ be the network, $\phi(x;\theta) = \|\mathcal{O}_\Omega(u_\theta(x))\|_2^2$, and $I(\theta) = \int_\Omega \phi(x;\theta)\,dx$.

A.3.1 Loss Estimators

Define

$$L_{\mathrm{GUS}}(\theta) = \hat{I}_{\mathrm{GUS}}(\theta) = \frac{|\Omega|}{M}\sum_{j=1}^{M}\phi(x_j;\theta), \tag{10}$$

$$L_{\mathrm{SetPINN}}(\theta) = \hat{I}_{\mathrm{EAS}}(\theta) = \sum_{k=1}^{K}\frac{|E_k|}{m_k}\sum_{i=1}^{m_k}\phi(x_{k,i};\theta). \tag{11}$$

By Lemma 4.2, $\mathbb{E}[L_{\mathrm{GUS}}(\theta)] = \mathbb{E}[L_{\mathrm{SetPINN}}(\theta)] = I(\theta)$. Under Assumption A.4, the variance identities Eqs. (8) and (9) hold verbatim with $\phi(\cdot)$ replaced by $\phi(\cdot;\theta)$ and I, I_k by $I(\theta), I_k(\theta)$.

A.3.2 Stochastic Gradient Estimators

Let $g(x;\theta) = \nabla_\theta\phi(x;\theta)$. Under Assumption A.6,

$$G_{\mathrm{GUS}}(\theta) = \nabla_\theta L_{\mathrm{GUS}}(\theta) = \frac{|\Omega|}{M}\sum_{j=1}^{M} g(x_j;\theta), \qquad G_{\mathrm{SetPINN}}(\theta) = \nabla_\theta L_{\mathrm{SetPINN}}(\theta) = \sum_{k=1}^{K}\frac{|E_k|}{m_k}\sum_{i=1}^{m_k} g(x_{k,i};\theta),$$

and both are unbiased for $\nabla_\theta I(\theta)$.

Theorem A.10 (Gradient Variance Reduction)**.** *(Theorem 4.4) Under Assumptions A.4 and A.6,*

$$\mathrm{Tr}\big(\mathrm{Cov}\big(G_{\mathrm{SetPINN}}(\theta)\big)\big) \;\le\; \mathrm{Tr}\big(\mathrm{Cov}\big(G_{\mathrm{GUS}}(\theta)\big)\big)\,.$$

Proof. Write $g = (g_1, \ldots, g_P)$ for $P = \dim(\theta)$. By independence and linearity of covariance for sums of i.i.d. terms,

$$\mathrm{Cov}\big(G_{\mathrm{GUS}}(\theta)\big) = \frac{|\Omega|^2}{M^2}\sum_{j=1}^{M}\mathrm{Cov}_{x\sim U(\Omega)}\big(g(x;\theta)\big) = \frac{|\Omega|^2}{M}\,\mathrm{Cov}_{x\sim U(\Omega)}\big(g(x;\theta)\big), \tag{12}$$

$$\mathrm{Cov}\big(G_{\mathrm{SetPINN}}(\theta)\big) = \sum_{k=1}^{K}\left(\frac{|E_k|}{m_k}\right)^2\sum_{i=1}^{m_k}\mathrm{Cov}_{x\sim U(E_k)}\big(g(x;\theta)\big) = \sum_{k=1}^{K}\frac{|E_k|^2}{m_k}\,\mathrm{Cov}_{x\sim U(E_k)}\big(g(x;\theta)\big). \tag{13}$$

With Assumption A.4, $|E_k|^2/m_k = |E_k|\,|\Omega|/M$, thus

$$\mathrm{Tr}\big(\mathrm{Cov}\big(G_{\mathrm{SetPINN}}\big)\big) = \frac{|\Omega|}{M}\sum_{k=1}^{K}|E_k|\,\mathrm{Tr}\Big(\mathrm{Cov}_{x\sim U(E_k)}(g)\Big).$$

Let $p \in \{1, \ldots, P\}$ denote a coordinate. Then

$$\mathrm{Tr}\Big(\mathrm{Cov}_{x\sim U(E_k)}(g)\Big) \;=\; \sum_{p=1}^{P}\mathrm{Var}_{x\sim U(E_k)}\big(g_p(x;\theta)\big).$$

Therefore,

$$\mathrm{Tr}\big(\mathrm{Cov}\big(G_{\mathrm{SetPINN}}\big)\big) = \frac{|\Omega|}{M}\sum_{p=1}^{P}\sum_{k=1}^{K}|E_k|\,\mathrm{Var}_{x\sim U(E_k)}\big(g_p(x;\theta)\big).$$

By the law of total variance with the element index Z taking value k with weight $\pi_k = |E_k|/|\Omega|$,

$$\mathrm{Var}_{x\sim U(\Omega)}\big(g_p(x;\theta)\big) = \sum_{k=1}^{K}\pi_k\,\mathrm{Var}_{x\sim U(E_k)}\big(g_p(x;\theta)\big) \;+\; \mathrm{Var}_{k\sim\{\pi\}}\Big(\mathbb{E}[g_p(x;\theta)\mid x\in E_k]\Big) \;\ge\; \sum_{k=1}^{K}\pi_k\,\mathrm{Var}_{x\sim U(E_k)}\big(g_p\big).$$

Multiplying both sides by $|\Omega|$ and summing over p yields

$$\frac{|\Omega|}{M}\sum_{p=1}^{P}\sum_{k=1}^{K}|E_k|\,\mathrm{Var}_{x\sim U(E_k)}(g_p) \;\le\; \frac{|\Omega|^2}{M}\sum_{p=1}^{P}\mathrm{Var}_{x\sim U(\Omega)}(g_p) \;=\; \mathrm{Tr}\big(\mathrm{Cov}\big(G_{\mathrm{GUS}}(\theta)\big)\big)\,,$$

using Eq. (12). This proves the claim. □

A.3.3 Implications

Theorems 4.3 and 4.4 formally justify the empirical findings in Section 5: EAS reduces the variance of both the loss estimator and its stochastic gradients, producing steadier optimization, improved coverage (via Corollary A.9), and higher accuracy. Equality in Theorem 4.3 characterizes the (rare) case where all elements have identical average residuals, in which case EAS and GUS coincide in variance.

Remark (beyond proportional allocation). Unbiasedness in Lemma 4.2 holds for any $m_k > 0$. Adaptive allocations (e.g., importance-style m_k) may further reduce variance; our results above benchmark EAS under the neutral proportional scheme of Assumption A.4.

B Training and Evaluation

B.1 Physics-Informed Neural Networks (PINNs)

Let $\Omega \subset \mathbb{R}^d$ and $x = (x_1, \ldots, x_{d-1}, t)$. We encode the PDE, initial, and boundary conditions via operators $\mathcal{O}_\Omega$, $\mathcal{O}_{\Omega_0}$, and $\mathcal{O}_{\partial\Omega}$:

$$\begin{aligned}\mathcal{O}_\Omega(u)(x) &= 0 && x \in \Omega,\\ \mathcal{O}_{\Omega_0}(u)(x) &= g(x) && x \in \Omega_0,\\ \mathcal{O}_{\partial\Omega}(u)(x) &= h(x) && x \in \partial\Omega.\end{aligned} \tag{14}$$

A PINN uses u_θ and minimizes a region-weighted Monte Carlo objective. For each region $X \in \{\Omega, \Omega_0, \partial\Omega\}$, let μ_X be the appropriate measure (Lebesgue for Ω, surface/initial-time for $\partial\Omega, \Omega_0$), $|X| := \mu_X(X)$, and draw N_X i.i.d. samples $\{x_i^X\}_{i=1}^{N_X} \sim U(X)$ (w.r.t. μ_X). Writing the targets as

$$b_X(x) = \begin{cases} 0, & X = \Omega,\\ g(x), & X = \Omega_0,\\ h(x), & X = \partial\Omega,\end{cases}$$

the PINN loss is

$$\mathcal{L}_{\mathrm{PINN}}(u_\theta) = \sum_{X\in\{\Omega,\Omega_0,\partial\Omega\}} \lambda_X \frac{|X|}{N_X}\sum_{i=1}^{N_X}\big\|\mathcal{O}_X(u_\theta)(x_i^X) - b_X(x_i^X)\big\|_2^2 \;+\; \lambda_{\mathrm{data}}\frac{1}{N_{\mathrm{data}}}\sum_{i=1}^{N_{\mathrm{data}}}\big\|u_\theta(x_i^{\mathrm{data}}) - y_i\big\|_2^2, \tag{15}$$

omitting the last term when no labeled data are used.

B.2 SetPINNs Training

Let the domain be partitioned into elements $\{E_k\}_{k=1}^K$ (Def. 3.1). For each region X, define the induced pieces $E_k^X := E_k \cap X$ with measure $|E_k^X| := \mu_X(E_k^X)$. In each training step:

1. For every k and region X with $|E_k^X| > 0$, draw m_k^X i.i.d. samples $\{x_{k,i}^X\}_{i=1}^{m_k^X} \sim U(E_k^X)$; let $M_X = \sum_k m_k^X$.
2. Map each element's set through the Mixer p_ψ, Set Processor f_ϕ, and PDE Probe q_η to obtain u_θ at the sampled points.
3. Compute localized energies

$$\mathcal{E}_X(E_k, u_\theta) := \int_{E_k^X}\big\|\mathcal{O}_X(u_\theta)(x) - b_X(x)\big\|_2^2\, d\mu_X(x) \;\approx\; \frac{|E_k^X|}{m_k^X}\sum_{i=1}^{m_k^X}\big\|\mathcal{O}_X(u_\theta)(x_{k,i}^X) - b_X(x_{k,i}^X)\big\|_2^2. \tag{16}$$

4. Minimize the measure-correct global objective

$$\mathcal{L}_{\text{SetPINNs}} = \sum_{X\in\{\Omega,\Omega_0,\partial\Omega\}} \lambda_X \frac{1}{|X|}\sum_{k=1}^{K}\mathcal{E}_X(E_k,u_\theta) \;=\; \sum_X \lambda_X \frac{|X|}{M_X}\sum_{k=1}^{K}\sum_{i=1}^{m_k^X}\big\|\mathcal{O}_X(u_\theta)(x_{k,i}^X)-b_X(x_{k,i}^X)\big\|_2^2. \quad (17)$$

We use Adam for warmup and LBFGS for fine-tuning. Proportional per-region allocation $m_k^X \propto |E_k^X|$ is used unless stated otherwise.

Evaluation. We report relative RMSE (rRMSE):

$$\text{rRMSE} = \sqrt{\frac{\sum_{n=1}^{N}(\hat{y}_n - y_n)^2}{\sum_{n=1}^{N} y_n^2}}. \quad (18)$$

C PDE Benchmarks and Typical Failure Modes

We follow standard benchmark configurations from Zhao et al. (2024); Wu et al. (2024). Below we list the PDEs, domains, BC/IC, and parameter choices used in our experiments.

1D Convection

$$\partial_t u + \beta\,\partial_x u = 0, \quad x\in[0,2\pi],\ t\in[0,1], \qquad u(x,0)=\sin x, \quad u(0,t)=u(2\pi,t), \quad (19)$$

with $\beta = 50$.

1D Reaction (logistic)

$$\partial_t u - \rho\,u(1-u) = 0, \quad x\in[0,2\pi],\ t\in[0,1], \qquad u(x,0)=\exp\Big(-\tfrac{(x-\pi)^2}{2(\pi/4)^2}\Big), \quad u(0,t)=u(2\pi,t), \quad (20)$$

with $\rho = 5$. The analytic solution is $u(x,t) = \dfrac{h(x)e^{\rho t}}{h(x)e^{\rho t}+1-h(x)}$ where $h(x)=u(x,0)$.

1D Wave

$$\partial_{tt}u - \beta\,\partial_{xx}u = 0, \quad x\in[0,1],\ t\in[0,1], \qquad u(x,0)=\sin(\pi x)+\tfrac{1}{2}\sin(\beta\pi x),\ \partial_t u(x,0)=0,\ u(0,t)=u(1,t)=0, \quad (21)$$

with $\beta = 4$. The solution is $u(x,t)=\sin(\pi x)\cos(2\pi t)+\frac{1}{2}\sin(\beta\pi x)\cos(2\beta\pi t)$.

2D Incompressible Navier–Stokes (vorticity form via velocity–pressure)

$$\partial_t u + \lambda_1(u\,\partial_x u + v\,\partial_y u) = -\partial_x p + \lambda_2(\partial_{xx}u+\partial_{yy}u), \quad (22)$$

$$\partial_t v + \lambda_1(u\,\partial_x v + v\,\partial_y v) = -\partial_y p + \lambda_2(\partial_{xx}v+\partial_{yy}v), \quad (23)$$

with $\lambda_1 = 1$, $\lambda_2 = 0.01$; domain, BC/IC as in Raissi et al. (2019).

Poisson with Localized Load (plate surrogate)

$$-\Delta u = f \quad \text{in } [0,1]^2, \qquad u=0 \text{ on } \partial\Omega, \quad (24)$$

with a rectangular load patch $f(x,y) = Q\,\mathbf{1}_{[x_0,x_1]\times[y_0,y_1]}(x,y)$, $Q=20$, $(x_0,x_1,y_0,y_1)=(0.25,0.30,0.70,0.75)$.

Harmonic Poisson

$$-\Delta u = A\sin(k_x\pi x)\sin(k_y\pi y) \quad \text{in } [0,1]^2, \qquad u=0 \text{ on } \partial\Omega, \quad (25)$$

with $A=500$, $k_x=5$, $k_y=3$.

3D Helmholtz

$$-\Delta u - \kappa^2 u = f \quad \text{in } [0,1]^3, \qquad u = 0 \text{ on } \partial\Omega. \tag{26}$$

Let $u^*(x,y,z) = \sin(k_x\pi x)\sin(k_y\pi y)\sin(k_z\pi z)$. We set

$$f(x,y,z) = \left(\pi^2(k_x^2+k_y^2+k_z^2) - \kappa^2\right) u^*(x,y,z),$$

so u^* is the exact solution for any $\kappa \geq 0$ (if $\kappa^2 = \pi^2(k_x^2 + k_y^2 + k_z^2)$ then $f \equiv 0$).

Typical Failure Modes

Table 2: Benchmarks, principal challenges, and common PINN failure modes.

Benchmark	**Primary challenge(s)**	**Typical PINN failure mode(s)**
1D Reaction	Sharp moving fronts; stiffness	Over-smoothing; poor gradient resolution; slow/unstable convergence
1D Wave	Wavefront propagation; high frequencies	Spectral bias; dispersion/dissipation; poor sharp-feature transport
Convection	Advective transport; Gibbs oscillations	False diffusion; directional bias; oscillations near discontinuities
2D Navier–Stokes	Coupled nonlinearities; multi-scale vortical features	Instability; loss-imbalance; failure to capture fine scales
Harmonic Poisson	BC sensitivity; smooth oscillatory forcing	Imprecise BC satisfaction; slow convergence on larger domains
Poisson (localized load)	Highly localized response; boundary-layer behavior	Missed peak response; poor local resolution near load-/boundary
3D Helmholtz	High dimensionality; oscillatory eigenmodes; pollution error	Severe spectral bias; phase/amplitude inaccuracy; high compute

D Computational Complexity and Scalability

Notation. Let M be the total number of collocation points, the domain be partitioned into K_{elm} elements, and let S be the (average) set size per element, so $M = K_{\text{elm}}\, S$. Let D denote the embedding width and L_{att} the number of attention blocks in the set processor. We write $\tilde{\mathcal{O}}(\cdot)$ to hide factors that are constant across methods (e.g., fixed D, number of residual terms, PDE order); explicit constants are stated where informative.

D.1 Per-iteration time complexity

Standard PINNs (pointwise MLP). A pointwise network evaluates u_θ and its residuals independently at M points:

$$\text{Time} \;=\; \mathcal{O}\big(M\, C_{\text{MLP}}\big) \;=\; \tilde{\mathcal{O}}(M),$$

where C_{MLP} is the per-point MLP+autodiff cost (constant for fixed width/depth and fixed-order PDE derivatives).

SetPINNs (per-element self-attention over sets of size S). Each iteration consists of (i) a pointwise Mixer p_ψ, (ii) an attention Set Processor f_ϕ applied *within* each element, and (iii) a pointwise Probe q_η:

$$\text{Time} \;=\; \underbrace{\mathcal{O}\big(M\,C_{\text{mix}}\big)}_{\text{Mixer}} \;+\; \underbrace{\mathcal{O}\big(K_{\text{elm}}\,L_{\text{att}}(S^2D + SD^2)\big)}_{\text{self-attn per element}} \;+\; \underbrace{\mathcal{O}\big(M\,C_{\text{probe}}\big)}_{\text{Probe}}.$$

For fixed D and L_{att}, the attention term dominates and simplifies to

$$\text{Time} \;=\; \tilde{\mathcal{O}}\big((M/S)\cdot S^2\big) \;=\; \tilde{\mathcal{O}}(M\,S).$$

Thus, with small S (e.g., $S \in \{4, 8, 16\}$), SetPINNs scale linearly in M with a mild multiplicative factor S.

PINNsFormer-style global/pseudo-sequence attention. If each of the M tokens attends to a window/pseudo-sequence of length S using self-attention, the dominant cost is

$$\text{Time} \;=\; \tilde{\mathcal{O}}(M\,S^2),$$

which is a factor $\approx S$ higher than SetPINNs' per-element attention for the same S.

D.2 Activation/memory complexity

Standard PINNs. Activations scale linearly with M: $\tilde{\mathcal{O}}(M)$ for forward, the same order (times a small constant) for backward/autodiff.

SetPINNs. Self-attention within each element stores $S \times S$ attention maps per block. Summed over elements and L_{att} blocks:

$$\text{Memory} \;=\; \tilde{\mathcal{O}}\big(L_{\text{att}}\,(M/S)\cdot S^2\big) \;=\; \tilde{\mathcal{O}}(M\,S),$$

plus linear terms for embeddings and MLPs. In practice this is managed by batching elements so that only $B_{\text{elm}}\,S$ tokens are resident at once.

PINNsFormer-style attention. Global/pseudo-sequence attention of width S over M tokens yields

$$\text{Memory} \;=\; \tilde{\mathcal{O}}(M\,S^2),$$

again a factor $\approx S$ larger than SetPINNs for the same S.

D.3 Autodiff overhead for PDE operators

Let the residual require up to r-th order derivatives w.r.t. inputs. Reverse/forward-mode AD introduces a multiplicative constant α_r to all methods (e.g., second derivatives typically incur a small constant multiple of the forward cost). This factor is independent of M and does not change the *relative* scalings above.

D.4 Scalability remarks

- **Element locality vs. global context.** By constraining attention to sets of size S, SetPINNs reduce time/memory from $\tilde{\mathcal{O}}(M\,S^2)$ to $\tilde{\mathcal{O}}(M\,S)$ while preserving the local interactions we care about; empirically, small S suffices to capture locality.
- **Parallelism.** Elements are independent at each step: both time and memory parallelize across devices/streams. With batch-of-elements B_{elm}, the effective token count per step is $B_{\text{elm}}S$.
- **Dimensionality.** The *per-point* cost of MLPs/attention is dimension-agnostic once the feature map is formed; however, the required M to resolve a d-dimensional domain at a target resolution typically scales with d. Hence all methods grow with M, but SetPINNs retain the S-limited overhead.
- **Once-only partitioning.** Unlike classical FEM pipelines, domain partitioning and set formation are done once at initialization; no repeated meshing/basis construction is needed during training.

Table 3: Asymptotic per-iteration complexity (dominant terms). M: total collocation points; S: set or pseudo-sequence length; D, L_{att} treated as fixed.

Method	Time	Memory	Notes
PINNs (MLP)	$\tilde{\mathcal{O}}(M)$	$\tilde{\mathcal{O}}(M)$	Pointwise; no attention
SetPINNs (local attn)	$\tilde{\mathcal{O}}(M\,S)$	$\tilde{\mathcal{O}}(M\,S)$	$M = K_{\text{elm}}S$; small S
PINNsFormer-style (global/pseudo)	$\tilde{\mathcal{O}}(M\,S^2)$	$\tilde{\mathcal{O}}(M\,S^2)$	Attention over windows of size S

Takeaway. For fixed local context size S, SetPINNs scale linearly with the collocation budget M with only a mild S factor in both time and memory, while global/pseudo-sequence attention incurs an extra factor $\approx S$. In practice we use small S (e.g., 4–16), which keeps SetPINNs' wall-clock comparable to state-of-the art baselines such as RoPINNs and far better than transformer-based baselines such as PINNsFormers (Appendix E.3) while enabling locality-aware learning.

E Experimental Details

E.1 Hyperparameters

We use the authors' official implementations where available and follow the PINNsFormer pipeline for baselines.[1] All methods use the same collocation budget M and comparable parameter counts.

Table 4: Model configurations (core settings) and parameter counts.

Model	Core settings	Params
PINNs	MLP, 4 layers, hidden 512	527K
QRes	MLP, 4 layers, hidden 256	397K
FLS	MLP, 4 layers, hidden 512 (sine first layer)	527K
RoPINNs	MLP, 4 layers, hidden 512	527K
PINNsFormer	enc=1, dec=1, embed 32, heads 2, hidden 512, k=5, Δt=10^{-4}	454K
SetPINNs (ours)	set size S=4, enc=1, embed 32, heads 2, hidden 512	366K

E.2 Training Setup

We train in two stages: Adam warmup then LBFGS fine-tuning (strong Wolfe). Physics terms use unit weights unless noted.

Table 5: Training hyperparameters (shared unless specified per-task).

Adam iterations	50K
LBFGS iterations	500
LBFGS line search	Strong Wolfe
$\lambda_\Omega, \lambda_{\Omega_0}, \lambda_{\partial\Omega}$	1, 1, 1
Train grid : Test grid	50×50 : 101×101
Points (train : test)	2500 : 10201

E.3 Runtime and Memory

All models are implemented in PyTorch and trained on a single NVIDIA A100. We report wall-clock over 10 runs on the convection task.

GPU memory. Parameter counts are comparable across models (Table 4); peak memory is similar under identical batch sizes. SetPINNs' per-element processing keeps resident tokens to $B_{\text{elm}}S$, aiding memory control.

[1] `https://github.com/AdityaLab/pinnsformer`

Table 6: Wall-clock timings. Runtime (s) of different models trained on convection equation averaged over 10 runs.

PINNs	314.43
QRes	266.95
FLS	369.03
RoPINNs	619.04
PINNsFormer	1015.64
SetPINNs (ours)	614.99

Table 7: Statistical comparison of SetPINNs with the second-best model on each benchmark. Mean ± standard deviation of rRMSE is reported. Best (lower) scores are in **bold**.

Task	**SetPINNs (mean ± std)**	**Second-Best (Model / mean ± std)**
White-box PDE Benchmarks		
1D-Wave	**0.078 ± 0.02**	FLS / 0.480 ± 0.19
1D-Reaction	0.061 ± 0.00	**RoPINNs / 0.030 ± 0.00**
Convection	**0.031 ± 0.00**	FLS / 0.897 ± 0.28
3D-Helmholtz	**0.072 ± 0.02**	RoPINNs / 0.132 ± 0.03
Navier-Stokes	**0.218 ± 0.08**	PINNsFormer / 0.841 ± 0.29
Harmonic	**0.025 ± 0.00**	RoPINNs / 0.092 ± 0.04
Plate	**0.324 ± 0.05**	QRes / 1.003 ± 0.00
Grey-box Benchmarks		
Act. Coeff.	**0.090 ± 0.00**	FLS / 0.152 ± 0.00
Agg. Break.	**0.220 ± 0.00**	PINNs / 0.451 ± 0.00

E.4 Significance Analysis

For each task we report mean ± std across 10 seeds in Table 7. When comparing two methods, we use two-sided t-tests. SetPINNs significantly outperforms baselines in 8/9 tasks. Note that some scores here might differ from Table 1 as we report mean and variance over 10 runs for significance analysis.

F Real-World Process Engineering Tasks

We evaluate SetPINNs on two grey-box, permutation-invariant chemical engineering problems: predicting activity coefficients (AC) in liquid mixtures and agglomerate breakage (AB) via population balances. Both tasks feature sparse/irregular data and physics constraints that are naturally expressed as set-level relations.

F.1 Predicting Activity Coefficients

Background. As introduced in Section 5.2, activity coefficients quantify deviations from ideal mixing, constrained by the Gibbs–Duhem relation (Eq. 5). Here we provide additional details on data preparation and modeling.

Setup. Experimental AC data are obtained from the Dortmund Data Bank (DDB). We: (i) filter low-quality records flagged by DDB; (ii) retrieve SMILES; (iii) canonicalize with RDKit (dropping unconvertible entries); and (iv) featurize each component using count-based Morgan fingerprints (radius 0, 128 bits). A *system*-wise split (mixture identity) uses 90% for training and 10% for testing, ensuring no mixture leakage across splits.

Modeling and constraint. We train grey-box SetPINNs with a physics term enforcing (5) via a residual penalty added to the data loss (weights as in §B). The permutation-equivariant set encoder ingests unordered component descriptors and compositions; predictions $\{\ln \gamma_1, \ln \gamma_2\}$ respect the coupling implied by (5).

F.2 Predicting Agglomerate Breakage

Background. As introduced in Section 5.2, the population balance equation (Eq. 6) governs the evolution of the particle-size distribution. Here we detail the analytic test case and experimental protocol. We use a standard analytic test: $r(x) = x$, $f(x, y) = 2/y$, and the mono-disperse initial condition $u(x, 0) = \delta(x - L)$. The solution is

$$u(x,t) = e^{-tx}\big(\delta(x-L) + \big[\,2t + t^2(L-x)\,\big]\Theta(L-x)\big)\,, \tag{27}$$

with $\delta(\cdot)$ the Dirac delta and $\Theta(\cdot)$ the Heaviside step function. Although real experiments require estimating unknown kernels (inverse PBE), this analytic case isolates approximation accuracy under known physics and sparse supervision.

F.3 Discussion for Chemical Tasks

- **Set structure & invariances.** Mixture properties and particulate systems are invariant to component/-particle ordering. SetPINNs' permutation-equivariant encoder matches this symmetry and handles variable-cardinality inputs naturally.

- **Physics-augmented learning.** Adding the Gibbs–Duhem residual (AC) or the PBE residual (AB) stabilizes training and improves generalization by ruling out physically inconsistent predictions.

- **Sparse, irregular data.** The element-/set-wise processing and EAS improve coverage and reduce estimator/gradient variance (see §4), aiding learning from scarce or unevenly distributed measurements.

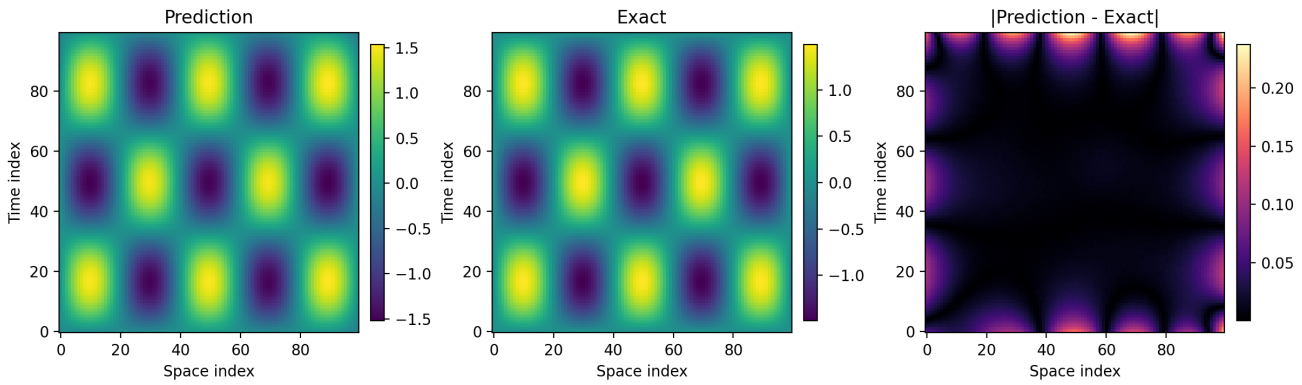

Figure 6: Qualitative comparison on the harmonic PDE. SetPINNs closely match the reference.

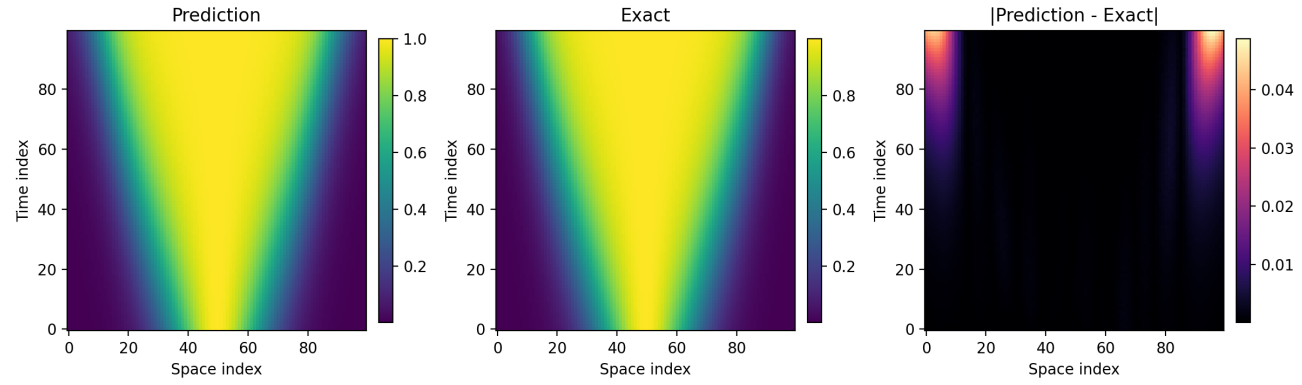

Figure 7: Qualitative comparison on the reaction PDE. SetPINNs closely match the reference.

G Ablation Studies

We isolate key design choices and sampling strategies. Unless stated, we use the reaction equation, 100 residual/BC/IC points, and a 100×100 grid; other hyperparameters follow Table 4. We report rRMSE (mean±std) over multiple seeds.

Qualitative sanity check. Fig. 6 and Fig. 7 shows SetPINNs preserve sharp fronts in reaction and oscillations in harmonic PDE. EAS with set-processing is able to accurately capture the oscillations in harmonic PDE.

Effect of sampling on vanilla PINNs. EAS improves vanilla PINNs without architectural changes, outperforming uniform, Latin Hypercube (LHS), and residual-adaptive distribution (RAD) on average.

Table 8: rRMSE (↓) for vanilla PINNs with different samplers (mean ± std).

PDE	Uniform	+ EAS	+ LHS	+ RAD
1D Wave	0.148 ± 0.01	0.119 ± 0.02	0.135 ± 0.02	0.132 ± 0.03
1D Reaction	0.801 ± 0.12	0.725 ± 0.16	0.766 ± 0.15	0.752 ± 0.14
Convection	1.136 ± 0.13	0.905 ± 0.07	1.020 ± 0.10	0.985 ± 0.09
Harmonic	0.342 ± 0.15	0.270 ± 0.04	0.295 ± 0.06	0.310 ± 0.08
Plate	1.467 ± 0.58	1.275 ± 0.31	1.380 ± 0.34	1.198 ± 0.29

Element size. Partitioning into smaller elements (hence smaller intra-set distances) consistently reduces error; see Fig. 8.

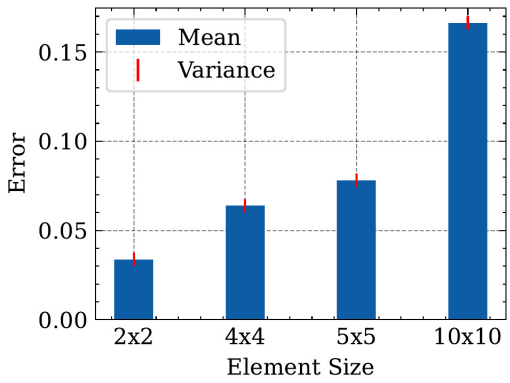

Figure 8: Error vs. element size $n \times n$. Larger sets (wider spatial extent) weaken local interactions and increase error.

Transformer depth. Increasing encoder blocks from 1 to 2 improves accuracy (Fig. 9), reflecting additional capacity for local interaction modeling.

Attention heads. Varying heads (with larger embedding to keep per-head width reasonable) has little effect (Fig. 10), indicating capacity is not head-limited in our regimes.

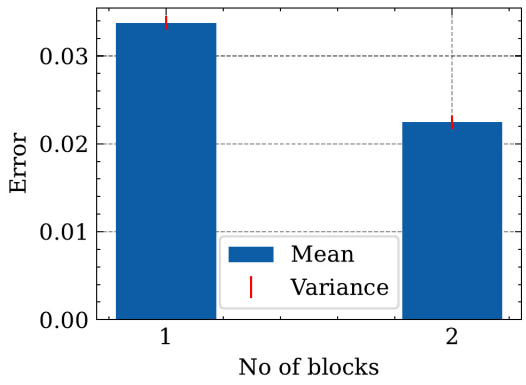

Figure 9: Effect of encoder depth on error. Two blocks > one block.

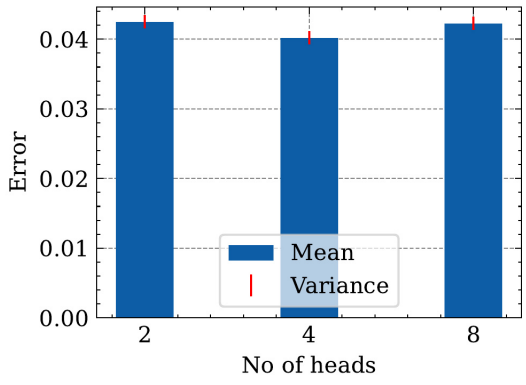

Figure 10: Error is largely insensitive to the number of attention heads.