Tensor Gaussian Processes: Efficient Solvers for Nonlinear PDEs

Qiwei Yuan[1] **Zhitong Xu**[1] **Yinghao Chen**[1] **Yiming Xu**[2] **Houman Owhadi**[3] **Shandian Zhe**[1]
[1] Kahlert School of Computing, University of Utah
[2] Department of Mathematics, University of Kentucky
[3] Department of Computing and Mathematical Sciences, California Institute of Technology

Abstract

Machine learning solvers for partial differential equations (PDEs) have attracted growing interest. However, most existing approaches, such as neural network solvers, rely on stochastic training, which is inefficient and typically requires a great many training epochs. Gaussian process (GP)/kernel-based solvers, while mathematical principled, suffer from scalability issues when handling large numbers of collocation points often needed for challenging or higher-dimensional PDEs. To overcome these limitations, we propose TGPS, a tensor-GP-based solver that introduces factor functions along each input dimension using one-dimensional GPs and combines them via tensor decomposition to approximate the full solution. This design reduces the task to learning a collection of one-dimensional GPs, substantially lowering computational complexity, and enabling scalability to massive collocation sets. For efficient nonlinear PDE solving, we use a partial freezing strategy and Newton's method to linerize the nonlinear terms. We then develop an alternating least squares (ALS) approach that admits closed-form updates, thereby substantially enhancing the training efficiency. We establish theoretical guarantees on the expressivity of our model, together with convergence proof and error analysis under standard regularity assumptions. Experiments on several benchmark PDEs demonstrate that our method achieves superior accuracy and efficiency compared to existing approaches. The code is released at `https://github.com/BayesianAIGroup/TGPSolve-NonLinear-PDEs`

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

1 Introduction

Machine learning (ML) solvers for partial differential equations (PDEs) have been receiving increasing attention due to their ease of implementation and competitive accuracy. These approaches approximate the solution with a machine learning model, such as deep neural networks (Raissi et al., 2019a), trained by minimizing a composite objective function that combines boundary and residual losses evaluated at a set of collocation points, thereby enforcing the boundary conditions and equation. Unlike traditional numerical solvers, ML approaches avoid complex, problem-specific discretization schemes and numerical routines, making them simpler and more convenient to implement and verify.

Despite these advantages, most existing ML solvers — including physics-informed neural networks (Raissi et al., 2019b) and recent Gaussian process (GP) and kernel-based methods (Fang et al., 2023; Xu et al., 2024) — rely on stochastic optimization to effectively learn the model parameters, which often requires tens of thousands to even millions of iterations, making solving procedure quite inefficient. GP and kernel-based solvers, though mathematically principled, also face scalability challenges as the number of collocation points grows. For example, Chen et al. (2021a); Long et al. (2022) place a GP prior over the solution and its derivatives, yielding block-structured covariance matrices whose time and memory costs exceed the standard $\mathcal{O}(M^3)$ and $\mathcal{O}(M^2)$ scaling, where M is the number of collocation points. To mitigate this, Fang et al. (2023); Xu et al. (2025) proposed using product kernels on Cartesian grids, exploiting Kronecker algebra for efficient matrix operations. However, this approach requires estimating the solution values at *all* grid points, leading to exponential growth in parameters with dimension, and its reliance on structured grids limits applicability to irregular domains.

To overcome these limitations, we propose TGPS, a tensor Gaussian process solver for nonlinear PDEs. Our main contributions are as follows.

Model: We introduce a set of one-dimensional factor functions, each modeled as a GP, along every input dimension. These factors are combined via multilinear tensor decompo-

sitions — such as CANDECOMP/PARAFAC (CP) (Harshman et al., 1970) or tensor-ring decomposition (Zhao et al., 2016) — to approximate the full solution. For each dimension, we place inducing points and represent the factor function by its GP conditional mean, with the inducing values serving as trainable parameters. Collocation points are freely sampled to form the training objective. This design allows our model to scale linearly with both the PDE dimension and the number of collocation points — not only in covariance matrix computation and storage but also in the number of trainable parameters.

Algorithm: To efficiently solve nonlinear PDEs, we linearize the nonlinear terms using two complementary strategies. The first is a partial freezing strategy, which fixes part of the nonlinear terms using results from the previous iteration, leaving only a linear component. The second is Newton's method, which approximates nonlinear terms via their first-order Taylor expansion. We then exploit two key properties of our model: (i) the solution approximation is multilinear in the inducing values, and (ii) derivatives of the solution preserve the same multilinear structure. Based on these insights, we design an alternating least squares (ALS) scheme that cyclically updates the inducing values along each dimension in closed form. This eliminates the need for stochastic optimization, yielding far greater efficiency and achieving accurate approximations with only a small number of iterations.

Theorem: We present a rigorous theoretical analysis of our framework. We show that, despite relying on a multilinear functional decomposition, our model can approximate the true solution arbitrarily well when provided with a sufficient number of factor functions (*i.e.,* rank). Under CP decomposition, we further prove that not only do such approximations exist within our modeling space, but also that, as the number of collocation points increases, the training optimum converges to these approximations. These results theoretically guarantee the effectiveness of our method in recovering high-quality solutions.

Experiments: We evaluate our method on a range of benchmark PDEs. In less challenging settings, where only a modest number of collocation points (*e.g.*, 1,000) suffices, our method consistently achieves lower or comparable errors than existing approaches. In more challenging cases — such as Burgers' equation with viscosity 0.001 or a 6D Allen-Cahn equation — our method seamlessly scales to tens of thousands of collocation points, achieving errors on the order of 10^{-3} to 10^{-6}. Across all benchmarks, it runs orders of magnitude faster than PINNs and recent GP solvers, while delivering comparable or superior accuracy with drastically reduced runtime.

2 Background

Consider solving a PDE of the general form:

$$\mathcal{P}(u) = a(\mathbf{x}) \ \ (\mathbf{x} \in \Omega), \quad \mathcal{B}(u) = b(\mathbf{x}) \ \ (\mathbf{x} \in \partial\Omega), \quad (1)$$

where $\mathbf{x} = (x_1, \ldots, x_d)^\top$, Ω and $\partial\Omega$ denote the interior and boundary domains respectively, $\mathcal{P}$ and $\mathcal{B}$ are (possibly nonlinear) differential operators applied to u.

Physics-Informed Neural Networks (PINNs). To solve (1), PINNs approximate the PDE solution using a (deep) neural network. A set of collocation points $\mathcal{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{M_\Omega} \in \Omega, \mathbf{x}_{M_\Omega+1}, \ldots, \mathbf{x}_M \in \partial\Omega\}$ is sampled, and the network is trained by minimizing the loss,

$$\Theta^* = \operatorname*{argmin}_{\Theta} \lambda_b \mathcal{F}_b(\Theta) + \mathcal{F}_r(\Theta) \quad (2)$$

where $\mathcal{F}_r = \frac{1}{M_\Omega} \sum_{m=1}^{M_\Omega} \left(\mathcal{P}(\mathrm{NN}_\Theta)(\mathbf{x}_m) - a(\mathbf{x}_m)\right)^2$ is the PDE residual loss, and $\mathcal{F}_b = \frac{1}{M-M_\Omega} \sum_{j=1}^{M-M_\Omega} \left(\mathcal{B}\left(\mathrm{NN}_\Theta\right)(\mathbf{x}_{M_\Omega+j}) - b(\mathbf{x}_{M_\Omega+j})\right)^2$ is the boundary loss. Here NN denotes the neural network, Θ its parameters, and $\lambda_b > 0$ a weighting coefficient. Training typically relies on stochastic optimization (*e.g.*, ADAM (Kingma, 2014)), as in standard neural network applications like image classification. Achieving good accuracy generally requires tens of thousands of iterations, and a second-order optimizer (*e.g.*, L-BFGS) is often used for refinement and stabilization (Shin, 2020; Li et al., 2023; Penwarden et al., 2023). As a result, while this framework is straightforward and convenient to implement, the training (solving) process is often lengthy and inefficient.

Gaussian Process and Kernel-Based Solvers. An alternative class of solvers is based on GP/kernel methods, which rest on strong mathematical foundations. In (Chen et al., 2021a; Long et al., 2022) , a GP prior is placed over the solution u and all its derivatives (or more generally, linear operators) $D_j(u)$ appearing in the PDE. The goal is to estimate the values of u and all $D_j(u)$ evaluated at the collocation points, which leads to a multi-variate Gaussian prior distribution with a block covariance matrix, $\mathbf{C} = \{\mathbf{C}_{ij}\}$, where each block $\mathbf{C}_{ij}$ is associated with a pair of linear operators (*e.g.*, derivatives and u itself). Since each collocation point can contribute to multiple values (*e.g.*, different $D_j(u)$), the covariance matrix $\mathbf{C}$ is typically larger than $M \times M$, where M is the number of collocation points. As M increases, the time and space complexity exceed $\mathcal{O}(M^3)$ and $\mathcal{O}(M^2)$, respectively, making computation prohibitively expensive or even infeasible for large M.

To mitigate this issue, recent work by Xu et al. (2025) approximates the solution using standard GP/kernel interpolation: $u(\mathbf{x}; \mathbf{u}_\mathcal{M}) = \kappa(\mathbf{x}, \mathcal{M})\mathbf{K}_{MM}^{-1}\mathbf{u}_\mathcal{M}$, where $\mathbf{K}_{MM} = \kappa(\mathcal{M}, \mathcal{M})$ is the $M \times M$ covariance matrix at the collocation points, and $\mathbf{u}_\mathcal{M}$ denotes the solution values at these points. Differential operators D_j are applied directly to this interpolation to approximate $D_j(u)$, enabling the use

of a training loss similar to (2). To reduce computation cost, collocation points are placed on a Cartesian grid: $\mathcal{M} = \mathbf{s}^1 \times \ldots \times \mathbf{s}^d$ where each $\mathbf{s}^j$ is a set of input locations along dimension j. By adopting a product kernel of the form $\kappa(\mathbf{x}, \mathbf{x}') = \prod_j \kappa_j(x_j, x_j')$, the covariance matrix admits a Kronecker product structure: $\kappa(\mathcal{M}, \mathcal{M}) = \mathbf{K}_1 \otimes \ldots \otimes \mathbf{K}_d$, where each $\mathbf{K}_j = \kappa_j(\mathbf{s}^j, \mathbf{s}^j)$ is the kernel matrix along input dimension j. Kronecker algebra (Kolda, 2006) allows one to avoid computing the full $\mathbf{K}_{MM}$; instead, only the smaller matrices $\mathbf{K}_j$ are computed and inverted, greatly reducing the cost. However, this method still requires estimating $\mathcal{U}_\mathcal{M}$ — the solution values over the entire grid $\mathcal{M}$. As the input dimension increases, the size of $\mathcal{U}_\mathcal{M}$ grows exponentially, making storage and estimation infeasible. Furthermore, effective training still relies on stochastic optimization, often requiring up to one million iterations (Xu et al., 2025), rendering the solving procedure inefficient in practice.

3 Our Method

3.1 Model

To leverage the principled mathematical framework of GPs/kernel methods while overcoming their scalability and efficiency bottlenecks, we propose TGPS, a tensor-GP based PDE solver. Specifically, we introduce a set of one-dimensional factor (component) functions for each input dimension i,

$$f_r^i : \Omega_0 \subset \mathbb{R} \to \mathbb{R} \in \mathcal{G}^i \ (1 \le r \le R_i), \tag{3}$$

where $1 \le i \le d$, and $\mathcal{G}^i$ is a Reproducing Kernel Hilbert Space (RKHS) induced by a Mercer kernel function $\kappa_i(\cdot, \cdot)$. In other words, we model each factor function as a GP, $f_r^i \sim \mathcal{GP}(0, \kappa_i(\cdot, \cdot))$. For simplicity of discussion, we assume the PDE domain $\Omega \cup \partial\Omega \subseteq \bigtimes_{i=1}^d \Omega_0$. We then combine these factor functions via mutilinear tensor decomposition to construct the solution approximation. We consider two representative tensor decomposition models.

CANDECOMP/PARAFAC(CP) Decomposition (Harshman et al., 1970). We set $R_1 = \ldots = R_d = R$, and model the solution function as

$$\begin{aligned} u(x_1, \ldots, x_d) &= \sum\nolimits_{r=1}^R \prod\nolimits_{i=1}^d f_r^i(x_i) \\ &= \left(\mathbf{f}^1(x_1) \circ \ldots \circ \mathbf{f}^d(x_d)\right)^\top \mathbf{1}, \end{aligned} \tag{4}$$

where each $\mathbf{f}^i(x_i) = \left(f_1^i(x_i), \ldots, f_R^i(x_i)\right)^\top$, and $\circ$ is Hadamard (element-wise) product.

Tensor-Ring (TR) Decomposition (Zhao et al., 2016). In each dimension i, we view the factor functions together as a single function with a matrix output, $\mathbf{F}^i = \{f_r^i(\cdot)\} : \mathbb{R} \to \mathbb{R}^{R_{i-1} \times R_i}$ where $R_0 = R_d$. The solution is modeled as

$$u(x_1, \ldots, x_d) = \text{Trace}(\mathbf{F}^1(x_1)\mathbf{F}^2(x_2) \cdots \mathbf{F}^d(x_d)). \tag{5}$$

When $d = 2$, TR decomposition reduces to CP decomposition: $\text{Trace}\left(\mathbf{F}^1(x_1)\mathbf{F}^2(x_2)\right) = \mathbf{f}^1(x_1)^\top \mathbf{f}^2(x_2) = \left(\mathbf{f}^1(x_1) \circ \mathbf{f}^2(x_2)\right)^\top \mathbf{1}$ where $\mathbf{f}^1(x_1) = \text{vec}\left(\left(\mathbf{F}^1(x_1)\right)^\top\right)$, $\mathbf{f}^2(x_2) = \text{vec}(\mathbf{F}^2(x_2))$, and vec denotes vectorization.

Our formulation can be interpreted as a multilinear decomposition in functional space. To learn these factor functions, we introduce a set of inducing locations $\boldsymbol{\gamma}_i = (\gamma_1, \ldots, \gamma_{N_i})^\top$ in each dimension i, and represent each factor function as kernel interpolation (GP conditional mean),

$$f_r^i(x_i) = \kappa_i(x_i, \boldsymbol{\gamma}_i)\mathbf{K}_i^{-1}\boldsymbol{\eta}_r^i, \tag{6}$$

where $\mathbf{K}_i = \kappa_i(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i)$, and $\boldsymbol{\eta}_r^i = \left(f_r^i(\gamma_1), \ldots, f_r^i(\gamma_{N_i})\right)^\top$ denotes the values of the factor function at the inducing locations, *i.e.,*inducing values. The learning is conducted by solving the following constrained optimization problem,

$$\begin{cases} \underset{\{f_r^i \in \mathcal{G}^i\}}{\text{minimize}} \ \sum_{i=1}^d \sum_{r=1}^R \|f_r^i\|_{\mathcal{G}^i}^2 \\ \text{s.t. } \frac{1}{M_\Omega} \sum_{m=1}^{M_\Omega} \left(\mathcal{P}(u)(\mathbf{x}_m) - a(\mathbf{x}_m)\right)^2 \\ + \frac{1}{M - M_\Omega} \sum_{m=M_\Omega+1}^{M} \left(\mathcal{B}(u)(\mathbf{x}_m) - b(\mathbf{x}_m)\right)^2 \le \delta^2, \\ f_r^i \text{ takes kernel interpolation form (6)}, \\ u \text{ takes the tensor decomposition form (4) or (5)}, \end{cases} \tag{7}$$

where $\|\cdot\|_{\mathcal{G}^i}$ is the RKHS norm under $\mathcal{G}^i$, and δ is a relaxation parameter. Since we approximate the full solution function with a multilinear (low-rank) decomposition, we introduce $\delta^2 > 0$ to guarantee the feasibility of optimization, and to establish convergence. See Section 4 for our theoretical analysis. Directly solving (7) can be unwieldy in practice. We may choose to solve an unconstrained optimization with soft regularization instead,

$$\begin{aligned} &\underset{\{\boldsymbol{\eta}_r^i\}}{\text{minimize}} \ \mathcal{F}(u(\mathbf{x}; \{\boldsymbol{\eta}_r^i\}); \alpha_1, \alpha_2) := \sum_{i=1}^d \sum_{r=1}^R \|f_r^i\|_{\mathcal{G}^i}^2 \\ &+ \alpha_1 \left[\frac{1}{M_\Omega} \sum\nolimits_{m=1}^{M_\Omega} \left(\mathcal{P}(u)(\mathbf{x}_m) - a(\mathbf{x}_m)\right)^2 - \delta^2/2\right] \\ &+ \alpha_2 \left[\frac{1}{M - M_\Omega} \sum\nolimits_{m=M_\Omega+1}^{M} \left(\mathcal{B}(u)(\mathbf{x}_m) - b(\mathbf{x}_m)\right)^2 - \delta^2/2\right] \end{aligned} \tag{8}$$

where $\alpha_1, \alpha_2 > 0$ represent regularization strength, and δ can be simply set to zero.

Minimizing (8) is equivalent to maximizing the log joint probability of our tensor GP model, *i.e.,* performing probabilistic training. Specifically, from the interpolation form (6), each squared RKHS norm is $\|f_r^i\|_{\mathcal{G}^i}^2 = \left(\boldsymbol{\eta}_r^i\right)^\top \mathbf{K}_i^{-1} \boldsymbol{\eta}_r^i$, which corresponds to the negative log prior probability of $\boldsymbol{\eta}_r^i$ under the GP prior over f_r^i, namely, $p(\boldsymbol{\eta}_r^i) = \mathcal{N}(\boldsymbol{\eta}_r^i | \mathbf{0}, \mathbf{K}_i)$. Meanwhile, the residual and boundary loss terms at each collocation point correspond to negative log Gaussian likelihoods, given by $N\left(a(\mathbf{x}_m) | \mathcal{P}(u)(\mathbf{x}_m), \alpha_1^{-1}\right)$ and $\mathcal{N}\left(b(\mathbf{x}_m) | \mathcal{B}(u)(\mathbf{x}_m), \alpha_2^{-1}\right)$.

3.2 Algorithm

While applying stochastic optimization to (8) is straightforward, it typically requires many iterations and is therefore in-

efficient. To improve learning efficiency and achieve higher accuracy, we propose an alternating least squares (ALS) approach that performs closed-form updates across input dimensions. For clarity, we focus on the CP decomposition in (4), noting that the TR decomposition extends naturally. To illustrate the idea, we present a nonlinear 2D Allen-Cahn equation as a concrete example,

$$u_{x_1x_1} + u_{x_2x_2} + u(u^2 - 1) = a(x_1, x_2), \tag{9}$$

where $a(x_1, x_2)$ is the source function.

First, we observe two key properties of our tensor-GP model: (i) the solution approximation is mutilinear in the inducing values in each dimension i, denoted as $\mathbf{H}_i = [\boldsymbol{\eta}_1^i, \ldots, \boldsymbol{\eta}_R^i] \in \mathbb{R}^{N_i \times R}$. Specifically, according to (4), we have $\mathbf{f}^i(x_i) = \left(\kappa_i(x_i, \boldsymbol{\gamma}_i)\mathbf{K}_i^{-1}\mathbf{H}_i\right)^\top = \mathbf{H}_i^\top \mathbf{w}_i(x_i)$ where $\mathbf{w}_i(x_i) = \mathbf{K}_i^{-1}\kappa_i(\boldsymbol{\gamma}_i, x_i)$, and

$$u(x_1, \ldots, x_d) = \left\langle \circ_{i=1}^d \mathbf{H}_i^\top \mathbf{w}_i(x_i), \mathbf{1} \right\rangle, \tag{10}$$

where $\langle \cdot, \cdot \rangle$ is the dot product. Hence, we have u is linear in each $\mathbf{H}_i$ when fixing inducing values in other dimensions. For example, the solution of (9) is modeled as $u(x_1, x_2) = \mathbf{w}_1(x_1)^\top \mathbf{H}_1 \mathbf{H}_2^\top \mathbf{w}_2(x_2)$. (ii) Due to the mutlilinear combination of 1D functions, any partial derivative over our solution approximation maintains the same mutilinear structure in (10): $\partial_{x_{j_1} \ldots x_{j_K}} u = \left\langle \circ_{i=1}^d \mathbf{H}_i^\top \widehat{\mathbf{w}}_i(x_i), \mathbf{1} \right\rangle$ where $\widehat{\mathbf{w}}_i(x_i)$ is the derivative of $\mathbf{w}_i(x_i)$ if $x_i \in \{x_{j_1}, \ldots, x_{j_K}\}$ and otherwise $\widehat{\mathbf{w}}_i(x_i) = \mathbf{w}_i(x_i)$. Therefore, the partial derivatives are still multilinear in $\mathbf{H}_i$'s. For instance, for (9), we have

$$\begin{aligned} u_{x_1x_1} &= \left(\mathbf{w}_1''(x_1)\right)^\top \mathbf{H}_1 \mathbf{H}_2^\top \mathbf{w}_2(x_2), \\ u_{x_2x_2} &= \mathbf{w}_1(x_1)^\top \mathbf{H}_1 \mathbf{H}_2^\top \mathbf{w}_2''(x_2), \end{aligned} \tag{11}$$

where $\mathbf{w}_1''(x_1) = \partial^2 \mathbf{w}_1 / \partial x_1^2$ and $\mathbf{w}_2''(x_2) = \partial^2 \mathbf{w}_2 / \partial x_2^2$.

Therefore, if the operators $\mathcal{P}$ and $\mathcal{B}$ in (1) are linear in u and its derivatives, the squared residual and boundary loss at each collocation point is quadratic to each $\mathbf{H}_i$ (see (7) and (8)). For example, suppose $d = 2$ and $\mathcal{P}[u] = u_{x_1x_1} + u_{x_2x_2}$. At collocation point $\mathbf{x}_m = (x_{m1}, x_{m2})$, the residual loss w.r.t $\mathbf{H}_1$ takes the form,

$$\left(\mathcal{P}(u)(\mathbf{x}_m) - a(x_m)\right)^2 = \left(\mathrm{tr}(\mathbf{B}_m^\top \mathbf{H}_1) - a(\mathbf{x}_m)\right)^2 \tag{12}$$

where $\mathbf{B}_m = \left(\mathbf{w}_1''(x_{m1})\mathbf{w}_2^\top(x_{m2}) + \mathbf{w}_1(x_{m1})\mathbf{w}_2''(x_{m2})^\top\right)$. $\mathbf{H}_2$, according to (11). The residual loss w.r.t $\mathbf{H}_2$ takes a similar form.

Furthermore, we observe that the sum of squared RKHS norms in (7) and (8) is also quadratic to each $\mathbf{H}_i$,

$$\sum_{r=1}^R \|f_r^i\|_{\mathcal{G}^i}^2 = \sum\nolimits_{r=1}^R \left(\boldsymbol{\eta}_r^i\right)^\top \mathbf{K}_i^{-1} \boldsymbol{\eta}_r^i = \mathrm{Trace}(\mathbf{K}_i^{-1} \mathbf{H}_i \mathbf{H}_i^\top).$$

Combining this with (12), we see that *optimizing any* $\mathbf{H}_i$ *while holding the other inducing values fixed reduces to a least-squares problem with a closed-form solution*. This naturally suggests an alternating least squares (ALS) scheme, where we cyclically update each $\mathbf{H}^i$ while keeping the others fixed. Unlike stochastic gradient descent, which is noisy, inaccurate, and requires carefully adjusted small stepsizes to prevent divergence, ALS updates are more direct and aggressive, leading to substantially higher efficiency.

However, a critical bottleneck arises when $\mathcal{P}$ and $\mathcal{B}$ are nonlinear. The nonlinear terms disrupt the mutilinear structure, making ALS infeasible. To address this issue, we use two complementary strategies to linearize the nonlinear terms.

Partial Freezing. The first strategy is to freeze part of the nonlinear terms from the previous iteration, leaving only a linear component. For example, consider the nonlinear term $u(u^2 - 1)$ in (9) as an example. We freeze $u^2 - 1$ and approximate $u(u^2 - 1) \approx u \cdot \left((u^{\text{prev}})^2 - 1\right)$, where u^{prev} is computed from the factor functions estimated in the previous iteration. At the collocation points, the values of $(u^{\text{prev}})^2 - 1$ are treated as constants, making the entire term mutilinear in the $\mathbf{H}_i$'s. As the updates proceed, the discrepency between u^{prev} and u gradually diminishes, and vanishes upon convergence.

Newton's method (Tangent Linearization). Consider solving the PDE as a root finding problem: $\mathcal{R}(u) = 0$, where $\mathcal{R}$ is the PDE operator. We apply Newton's method by linearizing $\mathcal{R}(u)$ around the previous iteration u^{prev} via a first-order Taylor expansion: $\mathcal{R}(u) \approx \mathcal{R}(u^{\text{prev}}) + J(u^{\text{prev}})(u - u^{\text{prev}}) = 0$, where $J(u^{\text{prev}})$ is the Fréchet derivative, $J(u^{\text{prev}}) = \left.\frac{\mathrm{d}\mathcal{R}(u^{\text{prev}} + \epsilon v)}{\mathrm{d}\epsilon}\right|_{\epsilon=0}$, and v is an arbitrarily small perturbation. By construction, this first-order approximation is always linear in u. In practice, the procedure amounts to replacing the nonlinear terms in the PDE and boundary conditions with their first-order Taylor expansions. For example, in (9), the cubic term u^3 is replaced by $(u^{\text{prev}})^3 + 3(u^{\text{prev}})^2(u - u^{\text{prev}})$.

Computational Complexity. At each iteration, we linearize the nonlinear terms and perform ALS updates on each $\mathbf{H}_i$ while holding the others fixed. The overall time complexity is thereby $\mathcal{O}(M \sum_{i=1}^d (N_i R)^2 + N_i^3 R^3)$, where M is the number of collocation points, N_i the number of inducing points in dimension i, and R the number of factor functions per dimension. Thus, the time complexity scales linearly with both the number of collocation points and the input dimension. The space complexity is $\mathcal{O}(\sum_{i=1}^d N_i R + N_i^2)$, which accounts for storing the inducing values $\mathbf{H}_i$ and the kernel matrices $\mathbf{K}_i$ in each dimension.

4 Theoretical Analysis

We first show that, while our model adopts a multilinear functional decomposition as in (4) and (5), this decomposition remains sufficiently expressive to accurately approximate the true solution u^* under Sobolev regularity.

Lemma 4.1. *Suppose that $u^* \in H^k(\Omega)$ for some $k \in \mathbb{N}$, where $H^k(\Omega)$ denotes the Sobolev space consisting of functions whose weak derivatives up to the order of k have finite L^2 norms. Given an arbitrary error $0 < \varepsilon < 1$, if we model u as the CP format in* (4) *and let $\overline{R} = (\frac{\sqrt{d}}{\varepsilon})^{\frac{d-1}{k}}$, then*

$$\min_{R \le \overline{R},\ u_r^i \in L^2(\Omega_0)} \|u^* - u\|_{L^2(\Omega)} \lesssim \varepsilon. \tag{13}$$

Moreover, if we further assume $u^* \in H^k_{\mathbf{v}}(\Omega)$ for some $\mathbf{v} \in \mathbb{R}^D_+$ satisfying $v_j \lesssim j^{-(1+\delta')/k}$ for some $\delta' > \delta + k$ where $\delta > 0$, then (13) *holds with $\overline{R} \lesssim (\frac{1}{\varepsilon})^{\frac{1}{k}}$, where the implicit constants depend on δ. Here $H^k_{\mathbf{v}}(\Omega)$ is a weighted Sobolev spaces defined in Appendix Definition C.1.*

Lemma 4.2. *Suppose that $u^* \in H^{k+1}(\Omega)$ for some $k \in \mathbb{N}$. Given an arbitrary error $0 < \varepsilon < 1$, if we model u as the TR format in* (5) *and let $\overline{R} = (\frac{\sqrt{d}}{\varepsilon})^{\frac{d}{k}}$, then*

$$\min_{\sum_{i=1}^d R_{i-1}R_i \le \overline{R},\ u_r^i \in L^2(\Omega_0)} \|u^* - u\|_{L^2(\Omega)} \lesssim \varepsilon. \tag{14}$$

If we further assume $u^* \in H^{k+1}_{\mathbf{v}}(\Omega)$ for some $\mathbf{v} \in \mathbb{R}^d_+$ satisfying $v_j \lesssim j^{-(1+\delta')/k}$ for some $\delta' > \delta + k$ where $\delta > 0$, then (14) *holds with $\overline{R} \lesssim \exp\{2(\frac{1}{\varepsilon})^{\frac{1}{k}} + \log(\frac{1}{\varepsilon})\}$, where the implicit constants depend on δ.*

Next, we establish the convergence result under the CP format. Our analysis follows the roadmap of Batlle et al. (2023); Xu et al. (2025), adopting standard assumptions on PDE stability and on the regularity of the domain and boundary conditions (Xu et al., 2025, Assumption 4.1).

Assumption 4.3. The following conditions hold:

- *(C1) (Regularity of the domain and its boundary)* $\Omega \subset \mathbb{R}^d$ with $d > 1$ is a compact set and $\partial\Omega$ is a smooth connected Riemannian manifold of dimension $d-1$ endowed with a geodesic distance $\rho_{\partial\omega}$.
- *(C2) (Stability of the PDE)* $\exists k, t \in \mathbb{N}$ with $k > d/2$ and $t > (d-1)/2$, and $\exists s, l \in \mathbb{R}$ such that for any $r > 0$, it holds that $\forall u_1, u_2 \in B_r(H^l(\Omega))$,

$$\begin{aligned}\|u_1 - u_2\|_{H^l(\Omega)} \le C\big(&\|\mathcal{P}(u_1) - \mathcal{P}(u_2)\|_{H^0(\Omega)} \\ &+ \|\mathcal{B}(u_1) - \mathcal{B}(u_2)\|_{H^0(\partial\Omega)}\big),\end{aligned} \tag{15}$$

  and $\forall u_1, u_2 \in B_r(H^s(\Omega))$,

$$\begin{aligned}&\|\mathcal{P}(u_1) - \mathcal{P}(u_2)\|_{H^k(\Omega)} + \|\mathcal{B}(u_1) - \mathcal{B}(u_2)\|_{H^t(\partial\Omega)} \\ &\le C\|u_1 - u_2\|_{H^s(\Omega)},\end{aligned} \tag{16}$$

  where $C = C(r) > 0$ is a constant independent of u_1 and u_2, $B(r)$ is an open ball with radius r, $H^j = W^{j,2}$ is a Sobolev space.
- *(C3)* The RKHS $\mathcal{U}$ is continuously embedded in $H^{s+\tau}(\Omega)$ where $\tau > 0$.

Lemma 4.4. *Let $u^* \in \mathcal{U}$ denote the unique strong solution of* (1)*, and suppose Assumption 4.3 holds. Let $\mathcal{M} = \mathcal{M}_\Omega \cup \mathcal{M}_{\partial\Omega}$ be a set of collocation points, with $\mathcal{M}_\Omega \subset \Omega$ and $\mathcal{M}_{\partial\Omega} \subset \partial\Omega$. Assume the Voronoi diagram induced by $\mathcal{M}$ has a uniformly bounded aspect ratio across all cells. Define the fill-distances $h_\Omega := \sup_{\mathbf{x}\in\Omega} \inf_{\mathbf{x}'\in\mathcal{M}_\Omega} |\mathbf{x} - \mathbf{x}'|$ and $h_{\partial\Omega} := \sup_{\mathbf{x}\in\partial\Omega} \inf_{\mathbf{x}'\in\mathcal{M}_{\partial\Omega}} \rho_{\partial\Omega}(\mathbf{x}, \mathbf{x}')$, where $|\cdot|$ is the Euclidean distance, and $\rho_{\partial\Omega}$ is a geodesic distance defined on $\partial\Omega$. Set $h = max(h_\Omega, h_{\partial\Omega})$. Suppose each RKHS $\mathcal{G}^i$, to which the factor functions in dimension i belong, is associated with a universal kernel, and that $\mathcal{U} = \mathcal{G}^1 \otimes \cdots \otimes \mathcal{G}^d$[1]. Then, for any arbitrarily small $\varepsilon > 0$, with a sufficiently large R and an appropriate δ, the optimization (learning) problem* (7) *under the CP format* (4)*, always has a minimizer $u^\dagger$, and this minimizer satisfies $\|u^\dagger - u^*\|_{H^l(\Omega)} \lesssim \varepsilon$ as $h \to 0$.*

Proposition 4.5. *Given the same set of collocation points $\mathcal{M}$ and δ, there exist constants $\alpha_{1\mathcal{M}}, \alpha_{2\mathcal{M}} > 0$ such that the minimizer of* (8) *with $\alpha_1 = \alpha_{1\mathcal{M}}$ and $\alpha_2 = \alpha_{2\mathcal{M}}$ coincides with the minimizer of* (7)*. In other words, with appropriately chosen regularization strengths, the minimizer of* (8) *inherits the same convergence guarantee.*

This result highlights that, as long as the model space is sufficiently expressive to contain a good approximation of the true solution (up to an arbitrarily small error level ε), our training formulation is able to recover such an approximation. In particular, as the number of collocation points increases, the optimization (learning) is guided toward identifying this accurate solution candidate. The proofs of these theorems are provided in Appendix Section C, E and F.

5 Related Work

While PINNs have achieved many success stories, *e.g.*, (Raissi et al., 2020; Jin et al., 2021; Sahli Costabal et al., 2020; Li et al., 2023), the training of PINN is often lengthy and challenging, partly because applying differential operators over NNs can complicate the loss landscape (Krishnapriyan et al., 2021). Alternatively, early works such as (Graepel, 2003; Raissi et al., 2017) developed GP models to solve linear PDEs from noisy observations of the source terms. Chen et al. (2021b); Long et al. (2022) further extended this direction by developing a kernel method capable of addressing both linear and nonlinear PDEs. Batlle et al. (2023) developed a rigorous convergence framework, establishing both convergence guarantees and rates of (Chen et al., 2021b). To mitigate the scalability issue for massive collocation sets, Chen et al. (2023) proposed a sparse approximation technique based on the sparse inverse Cholesky factorization (Schafer et al., 2021).

Xu et al. (2025) proposed using a standard kernel interpolation framework to approximate the PDE solution, and

[1] Here $\mathcal{U}$ is a completed tensor product space, and $\otimes$ is the topological tensor product, *i.e.,* the completion of the algebraic tensor product with respect to the product RKHS norm rather than the algebraic tensor product.

induced a Kronecker product structure to simply the computation. The efficiency of Kronecker-structured GP models has been studied in prior works (Saatcci, 2012; Wilson and Nickisch, 2015; Wilson et al., 2015; Izmailov et al., 2018). Fang et al. (2023) leveraged a similar idea for solving high-frequency and multiscale PDEs, using a spectral mixture kernel along each input dimension to capture dominant frequencies in the kernel function. Broader discussions on Bayesian approaches to PDEs are given in (Owhadi, 2015).

The idea of using separable function approximations for solving PDEs was introduced by Beylkin and Mohlenkamp (2002) and has seen rapid development in recent years. Notable works include Richter et al. (2021), Oster et al. (2022), and Fackeldey et al. (2022), which apply tensor decomposition techniques — often with random sampling — to solve PDEs and/or stochastic differential equations efficiently. A comprehensive review of deterministic methods in this context can be found in Bachmayr (2023).

6 Numerical Experiments

We evaluated our method on five commonly-used benchmark PDE families from the literature on machine learning PDE solvers (Raissi et al., 2019a; Chen et al., 2021c; Xu et al., 2024): viscous Burger's equations, nonlinear elliptic PDEs, Eikonal PDEs, Allen-Cahn equations, and nonlinear Darcy Flow equations. The details are provided in Appendix Section A. We denote our method using partial freezing as TGPS-PF and Newton's method as TGPS-NT. We compared against two state-of-the-art GP/kernel-based solvers: (1) DAKS (Derivative-Augmented Kernel Solver) Chen et al. (2021c), which augments the GP covariance matrix with derivative information to estimate solution values and their derivatives at the collocation points. (2) SKS (Simple Kernel-based Solver) (Xu et al., 2025), which employs standard GP interpolation. We also compared with (3) PINN (Raissi et al., 2019a). SKS, DAKS and TGPS were implemented with JAX (Frostig et al., 2018), while PINN with PyTorch (Paszke et al., 2019). Hyperparameters and settings are detailed in Appendix Section B.

6.1 Solution Accuracy

Simpler Cases. We first evaluated all methods on relatively simple benchmarks, where only a modest number of collocation points is required. Specifically, we considered Burgers' equation (17) with viscosity $\nu = 0.02$, the nonlinear elliptic PDE (18), and the Eikonal PDE (19) — the same test cases as in (Chen et al., 2021b). Following the setups in (Chen et al., 2021b; Xu et al., 2025), the number of collocation points was varied as 600, 1200, 2400, 4800 for Burgers' equation and 300, 600, 1200, 2400 for the nonlinear elliptic and Eikonal PDEs. DAKS employed randomly sampled collocation points, whereas SKS used a regularly spaced square grid with a comparable number of points. Notably,

Table 1: Relative L^2 error of solving *simpler* PDEs, with a small number of collocation points. Inside the parenthesis of each top row indicates the grid used by SKS, which takes approximately the same number of collocation points used by other methods. The top two results are shown in bold face.

(a) Burgers' equation (17) with viscosity $\nu = 0.02$

Method	600 (25×25)	1200 (35×35)	2400 (49×49)	4800 (70×70)
DAKS	3.05E-02	1.38E-02	1.51E-03	1.70E-04
PINN	**4.67E-03**	1.17E-03	6.27E-04	6.50E-04
SKS	2.51E-02	9.41E-03	1.36E-03	5.59E-04
TGPS-PF	**3.56E-03**	**5.77E-04**	**1.35E-04**	**8.83E-05**
TGPS-NT	8.42E-03	**8.50E-04**	**1.46E-04**	**5.71E-05**

(b) Nonlinear elliptic PDE (18)

Method	300 (18×18)	600 (25×25)	1200 (35×35)	2400 (49×49)
DAKS	1.03E-01	1.03E-04	7.78E-04	1.51E-07
PINN	3.05E-01	1.73E-02	1.15E-03	2.88E-04
SKS	1.13E-02	6.23E-05	6.11E-06	1.65E-06
TGPS-PF	**1.97E-06**	**2.82E-07**	**1.28E-07**	**4.04E-08**
TGPS-NT	**1.78E-06**	**3.52E-07**	**1.74E-07**	**4.06E-08**

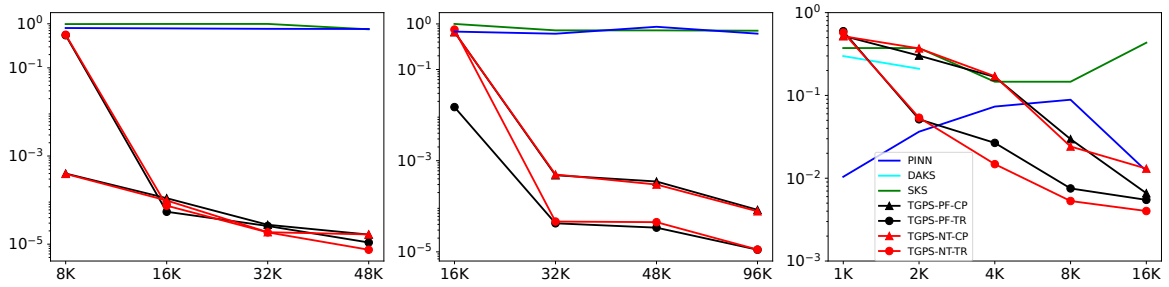
(c) Eikonal PDE (19)

Method	300	600	1200	2400
DAKS	5.65E-01	9.18E-02	1.27E-03	4.35E-04
PINN	1.65E-01	7.05E-02	2.54E-02	1.96E-02
SKS	3.49E-03	1.50E-03	1.07E-03	1.40E-04
TGPS-PF	**2.20E-04**	**1.56E-04**	**6.99E-05**	**9.98E-05**
TGPS-NT	**3.60E-04**	**9.06E-05**	**5.95E-05**	**7.23E-05**

DAKS performed worse under gridded collocation, as also observed in (Xu et al., 2025). We ran PINN and our method on both randomly sampled points and regularly spaced grids, and report the best outcomes. For methods using random sampling, each experiment was repeated five times and the average relative L^2 error was recorded. As shown in Table 1, our method — both TGPS-PF and TGPS-NT — consistently achieves the *highest solution accuracy*. The only exception is when solving Burgers' equation ($\nu = 0.02$) with 600 collocation points, where TGPS-NT performs slightly worse than PINN but still ranks third. Note that for all these PDEs the input dimension is $d = 2$, under which the TR and CP decomposition forms are equivalent (see Section 3.1). Accordingly, we do not introduce additional notation to distinguish between them.

Difficult Cases. Next, we evaluated the methods on more challenging problems: Burgers' equation with $\nu = 0.001$, the 2D Allen-Cahn equation with $a = 15$ and $a = 20$, higher-dimensional Allen-Cahn equations with $a = 15$, $d = 4$ and $d = 6$, and the 6D nonlinear Darcy flow.

To assess the necessity of massive collocation, we first used the same scale as in the simpler PDEs and tested on these PDEs with input dimension $d = 2$. As shown in Appendix Table 3, the performance of all the methods deteriorates noticeably while TGPS still consistently outperforms the



(a) 4D Allen-Cahn ($a = 15$) (b) 6D Allen-Cahn ($a = 15$) (c) 6D Nonlinear Darcy Flow

Figure 1: Relative L^2 error *vs.* the number of collocation points in solving higher-dimensional PDEs.

competing methods. Notably, with 2400 and 4800 collocation points, TGPS achieves relative L^2 errors of 10^{-5} and 10^{-6}, respectively, on the 2D Allen-Cahn equation with $a = 15$, and 10^{-3} and 10^{-5} with $a = 20$.

We then increased the number of collocation points to the range of 6.4K – 40K. In this regime, DAKS became prohibitively expensive, so no results are reported. As shown in Table 2, TGPS substantially reduce errors, achieving 10^{-4} for Burgers' equation with 28K–32K collocation points, and 10^{-6} for the 2D Allen-Cahn equation (both $a = 15$ and $a = 20$) across all the cases. While SKS also improves significantly, its accuracy is consistently worse than TGPS (except for Burgers' equation with 8K collocation points, where SKS slightly outperforms TGPS-PF), often by one order of magnitude. PINN, by contrast, only reaches a relative L^2 error of 10^{-2} on Burgers' equation and remains highly inaccurate on Allen-Cahn (relative L^2 error exceeding one), with little benefit from additional collocation points. This poor performance is likely due to the relatively high-frequency components in the Allen-Cahn solution (see (20)), which neural networks struggle to capture because of their known spectral bias (Rahaman et al., 2019).

We next evaluated our method on higher-dimensional PDEs. As shown in Figure 1, with only a few thousand collocation points, all methods exhibit large relative L^2 errors, indicating that these collocation points are insufficient. As the collocation number increases, the performance of TGPS improves substantially (*e.g.*, achieving 7.5×10^{-6} and 1.1×10^{-5} errors in the 4D and 6D Allen-Cahn equations). By contrast, SKS and PINN show little improvement. This is likely because they require far more collocation points to realize significant gains. For instance, when applying SKS to the 6D Allen-Cahn equation, even 96K collocation points correspond to a $7 \times 7 \times 7 \times 7 \times 7 \times 7$ grid — far too coarse to achieve meaningful accuracy. However, increasing the grid to a reasonably dense level — say, 100 points per dimension — causes the number of model parameters in SKS to explode. PINNs may still suffer from spectral bias in solving higher-dimensional Allen-Cahn equations. Moreover, DAKS can only accommodate a few thousand collocation points (see Figure 1c), which is inadequate for higher-dimensional problems. These results not only show the superiority of TGPS in solution accuracy, but also highlight its ability to leverage collocation points more efficiently than competing methods.

Table 2: Relative L^2 error of solving *more difficult* PDEs.

(a) The Burgers' equation (17) with viscosity $\nu = 0.001$.

Method	8000 (200×40)	16000 (400×40)	28000 (700×40)	32000 (800×40)
PINN	5.58E-01	2.65E-01	2.30E-02	1.07E-02
SKS	**2.65E-02**	9.41E-03	4.20E-03	3.88E-03
TGPS-PF	3.41E-02	**5.32E-03**	**4.75E-04**	**5.05E-04**
TGPS-NT	**3.30E-02**	**4.85E-03**	**5.44E-04**	**6.52E-04**

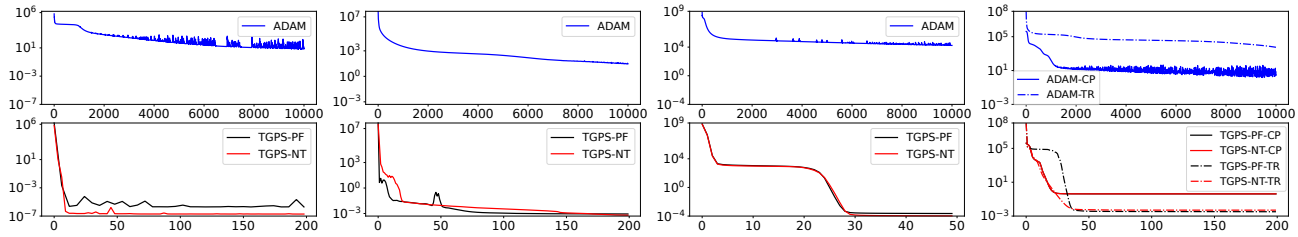
(b) The 2D Allen-Cahn equation (20) with $a = 15$

Method	6400 (80×80)	8100 (90×90)	22500 (150×150)	40000 (200×200)
PINN	7.11E0	7.50E0	5.95E0	8.29E0
SKS	1.17E-04	4.82E-05	6.14E-06	6.28E-06
TGPS-PF	**6.00E-06**	**1.21E-06**	**4.87E-06**	**1.43E-06**
TGPS-NT	**3.99E-06**	**1.28E-06**	**1.70E-06**	**1.76E-06**

(c) The 2D Allen-Cahn equation (20) with $a = 20$

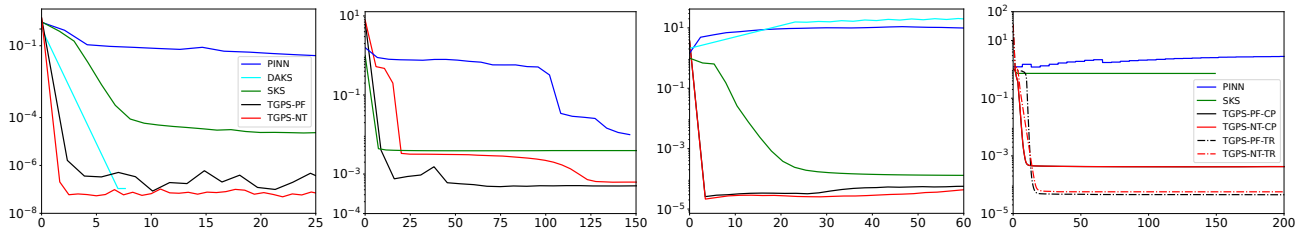
Method	6400	8100	22500	40000
PINN	5.91E0	6.29E0	8.29E0	8.39E0
SKS	5.63E-04	2.57E-04	5.66E-05	4.21E-05
TGPS-PF	**8.50E-06**	**8.47E-06**	**5.90E-06**	**5.14E-06**
TGPS-NT	**9.03E-06**	**7.42E-06**	**5.79E-06**	**4.80E-06**

Comparison with Conventional Numerical Methods. We compared against two established numerical methods. The first is the P2 Galerkin Finite Element Method (FEM) (Barrett and Liu, 1993; Brenner and Scott, 2008), and the second is a robust finite difference (FD) scheme. Details are provided in Appendix Section G. We conducted experiments on the nonlinear elliptic PDE (18) and the 2D Allen–Cahn equation (20) with $a = 15$ and $a = 20$, where ground-truth solutions are available for fair comparison. For FEM, we set the mesh spacing to match the collocation grids used in SKS. The FD scheme used the same grids as SKS. As reported in Appendix Table 4, our method (TGPS) consistently outperforms the conventional numerical solvers, often



(a) Nonlinear Elliptic (b) Burgers ($\nu = 0.001$) (c) 2D Allen-Cahn ($a = 15$) (d) 6D Allen-Cahn ($a = 15$)

Figure 2: Training curves: Training loss *vs.* Number of iterations.



(a) Nonlinear Elliptic (b) Burgers ($\nu = 0.001$) (c) 2D Allen-Cahn ($a = 15$) (d) 6D Allen-Cahn ($a = 15$)

Figure 3: Running time (in seconds) *vs.* Relative L^2 error.

by several orders of magnitude in error.

Point-Wise Error. For a more fine-grained comparison, we present the pointwise error of each method when solving Burgers' equation ($\nu = 0.001$) and 2D Allen-Cahn equations. Details are provided in Appendix Section H.

Irregular Domains. We further tested our approach by solving the nonlinear elliptic PDE and 2D Allen-Cahn equation on two irregular domains: one is circular and the other triangular. In all the cases, TGPS attained errors at the same level as those on regular domains, thereby confirming the advantage of TGPS as a mesh-free method. Detailed results and discussion are provided in Appendix Section I.

6.2 Running Efficiency

We next evaluated the computational efficiency of our method. Specifically, we tested four PDEs of increasing difficulty — nonlinear elliptic, Burgers' equation with $\nu = 0.001$, 2D Allen-Cahn ($a = 15$), and 6D Allen-Cahn ($a = 15$) — using 2.5K, 28K, 4.8K, and 48K collocation points, respectively. All experiments were conducted on a Linux workstation equipped with an NVIDIA A100 GPU.

Our first objective was to assess whether our ALS training is more efficient than the widely used stochastic gradient descent (SGD) methods. To this end, we trained our model with ADAM using exactly the same training loss and initial learning rate 10^{-2}. For a fair comparison, both the ADAM and ALS started from identical model initialization and hyperparameters. Moreover, no mini-batch sampling was applied: all collocation points were used to compute the full gradient of the training loss at each step, with ADAM adjusting the gradient through momentum and per-element step sizes. We then compared the learning curves of ADAM and ALS. As shown in Figure 2, ALS training converges hundreds to thousands of times faster than ADAM in all the cases. With our ALS, the training loss saturates after 25, 150, 30, and 50 iterations for nonlinear elliptic, Burgers', 2D Allen-Cahn, and 6D Allen-Cahn, respectively. In contrast, after 10K iterations, ADAM still yields training losses several orders of magnitude larger. These results confirm that our ALS training, with its closed-form updates at each iteration, dramatically improves efficiency compared to standard SGD-based training.

Our second objective was to evaluate efficiency relative to competing approaches. For this purpose, we examined how the relative L^2 error of each method evolves with training time. The results, shown in Figure 3, indicate that within the same runtime, TGPS almost always achieves the smallest solution error, underscoring its efficiency advantage. Note that DAKS applies only to nonlinear elliptic and 2D Allen-Cahn due to the use of smaller numbers of collocation points. When solving the nonlinear elliptic equation, although DAKS employs a Gauss–Newton approach that converges quickly, each iteration is significantly more expensive (due to computing inverse Hessian approximations). Consequently, TGPS still requires roughly half the runtime to achieve comparable or better accuracy. For Burgers' equation, SKS converges faster — likely due to its efficient Kronecker product algebra — but ultimately saturates at a relative L^2 error much larger than both TGPS-PF and TGPS-NT. Across all cases, PINN exhibits both slower

convergence and substantially larger errors.

Overall, the results demonstrate that our method not only achieves superior solution accuracy compared to competing ML solvers, but also requires much less runtime to do so.

7 Conclusion

We have introduced TGPS, a new machine learning solver for nonlinear PDEs. By combining one-dimensional Gaussian processes with tensor decomposition, our method alleviates the computational bottleneck of covariance matrices, controls the number of model parameters, and scales efficiently to massive collocation sets and higher-dimensional PDEs. The alternating least-squares (ALS) updates, coupled with partial freezing and Newton's method, yield substantial efficiency gains over standard stochastic gradient descent training. We also established theoretical guarantees on the expressivity of the model, as well as its accuracy and convergence as the number of collocation points increases. Experimental results on a variety of benchmark PDEs demonstrate not only high solution accuracy but also significant improvements in runtime efficiency.

Acknowledgments

HO and SZ acknowledge support from the Air Force Office of Scientific Research under MURI award number FA9550-20-1-0358 (Machine Learning and Physics-Based Modeling and Simulation). HO acknowledges support from the Air Force Office of Scientific Research under MURI award number FOA-AFRL-AFOSR-2023-0004 (Mathematics of Digital Twins), the Department of Energy under award number DE-SC0023163 (SEA-CROGS: Scalable, Efficient, and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems), the National Science Foundations under award number 2425909 (Discovering the Law of Stress Transfer and Earthquake Dynamics in a Fault Network using a Computational Graph Discovery Approach), and from the DoD Vannevar Bush Faculty Fellowship Program under ONR award number N00014-18-1-2363. SZ acknolwedges support from NSF CAREER Award IIS-2046295, NSF OAC-2311685 (Elements: A Convergent Physics-based and Data-driven Computing Platform for Building Modeling), NSF DMS-2529112 (Collaborative Research: MATH-DT: Computationally efficient hypercomplex variable-based sensitivity methods for rapid Digital Twin model updating), and DARPA SURGE HR0011-25-C-0036 (Rapid certification of additive manufactured components using ML/AI and physics-based modeling).

References

Bachmayr, M. (2023). Low-rank tensor methods for partial differential equations. Acta Numerica, 32:1–121.

Barrett, J. W. and Liu, W. B. (1993). Finite element approximation of the p-Laplacian. Mathematics of computation, 61(204):523–537.

Batlle, P., Chen, Y., Hosseini, B., Owhadi, H., and Stuart, A. M. (2023). Error analysis of kernel/gp methods for nonlinear and parametric pdes. arXiv preprint arXiv:2305.04962.

Batlle, P., Chen, Y., Hosseini, B., Owhadi, H., and Stuart, A. M. (2025). Error analysis of kernel/gp methods for nonlinear and parametric pdes. Journal of Computational Physics, 520:113488.

Beylkin, G. and Mohlenkamp, M. J. (2002). Numerical operator calculus in higher dimensions. Proceedings of the National Academy of Sciences, 99(16):10246–10251.

Brenner, S. C. and Scott, L. R. (2008). The mathematical theory of finite element methods. Springer.

Chen, Y., Hosseini, B., Owhadi, H., and Stuart, A. M. (2021a). Solving and learning nonlinear pdes with gaussian processes. Journal of Computational Physics, 447:110668.

Chen, Y., Hosseini, B., Owhadi, H., and Stuart, A. M. (2021b). Solving and learning nonlinear PDEs with Gaussian processes. arXiv preprint arXiv:2103.12959.

Chen, Y., Owhadi, H., and Schäfer, F. (2023). Sparse cholesky factorization for solving nonlinear pdes via gaussian processes. arXiv preprint arXiv:2304.01294.

Chen, Z., Liu, Y., and Sun, H. (2021c). Physics-informed learning of governing equations from scarce data. Nature communications, 12(1):1–13.

Fackeldey, K., Oster, M., Sallandt, L., and Schneider, R. (2022). Approximative policy iteration for exit time feedback control problems driven by stochastic differential equations using tensor train format. Multiscale Modeling & Simulation, 20(1):379–403.

Fang, S., Cooley, M., Long, D., Li, S., Kirby, R., and Zhe, S. (2023). Solving high frequency and multi-scale pdes with gaussian processes. arXiv preprint arXiv:2311.04465.

Frostig, R., Johnson, M. J., and Leary, C. (2018). Compiling machine learning programs via high-level tracing. Systems for Machine Learning, 4(9).

Graepel, T. (2003). Solving noisy linear operator equations by Gaussian processes: Application to ordinary and partial differential equations. In ICML, pages 234–241.

Griebel, M. and Harbrecht, H. (2023). Analysis of tensor approximation schemes for continuous functions. Foundations of Computational Mathematics, pages 1–22.

Harshman, R. A. et al. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. UCLA working papers in phonetics, 16(1):84.

Izmailov, P., Novikov, A., and Kropotov, D. (2018). Scalable gaussian processes with billions of inducing inputs via tensor train decomposition. In International Conference on Artificial Intelligence and Statistics, pages 726–735.

Jin, X., Cai, S., Li, H., and Karniadakis, G. E. (2021). Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. Journal of Computational Physics, 426:109951.

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

Kolda, T. G. (2006). Multilinear operators for higher-order decompositions, volume 2. United States. Department of Energy.

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.

Li, S., Penwarden, M., Xu, Y., Tillinghast, C., Narayan, A., Kirby, R., and Zhe, S. (2023). Meta learning of interface conditions for multi-domain physics-informed neural networks. In Proceedings of the 40th International Conference on Machine Learning, pages 19855–19881.

Long, D., Wang, Z., Krishnapriyan, A., Kirby, R., Zhe, S., and Mahoney, M. (2022). Autoip: A united framework to integrate physics into gaussian processes. In International Conference on Machine Learning, pages 14210–14222. PMLR.

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

Oster, M., Sallandt, L., and Schneider, R. (2022). Approximating optimal feedback controllers of finite horizon control problems using hierarchical tensor formats. SIAM Journal on Scientific Computing, 44(3):B746–B770.

Owhadi, H. (2015). Bayesian numerical homogenization. Multiscale Modeling & Simulation, 13(3):812–828.

Owhadi, H. and Scovel, C. (2019). Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design, volume 35. Cambridge University Press.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32.

Penwarden, M., Jagtap, A. D., Zhe, S., Karniadakis, G. E., and Kirby, R. M. (2023). A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions. Journal of Computational Physics, 493:112464.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. (2019). On the spectral bias of neural networks. In International conference on machine learning, pages 5301–5310. PMLR.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019a). 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.

Raissi, M., Wang, Z., Triantafyllou, M. S., and Karniadakis, G. E. (2019b). Deep learning of vortex-induced vibrations. Journal of Fluid Mechanics, 861:119–137.

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.

Richter, L., Sallandt, L., and Nüsken, N. (2021). Solving high-dimensional parabolic pdes using the tensor train format. In International Conference on Machine Learning, pages 8998–9009. PMLR.

Saatcci, Y. (2012). Scalable inference for structured Gaussian process models. PhD thesis, Citeseer.

Sahli Costabal, F., Yang, Y., Perdikaris, P., Hurtado, D. E., and Kuhl, E. (2020). Physics-informed neural networks for cardiac activation mapping. Frontiers in Physics, 8:42.

Schafer, F., Katzfuss, M., and Owhadi, H. (2021). Sparse cholesky factorization by kullback–leibler minimization. SIAM Journal on scientific computing, 43(3):A2019–A2046.

Shin, Y. (2020). On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. Communications in Computational Physics, 28(5):2042–2074.

Wilson, A. and Nickisch, H. (2015). Kernel interpolation for scalable structured gaussian processes (kiss-gp). In International Conference on Machine Learning, pages 1775–1784.

Wilson, A. G., Dann, C., and Nickisch, H. (2015). Thoughts on massively scalable gaussian processes. arXiv preprint arXiv:1511.01870.

Xu, Z., Long, D., Xu, Y., Yang, G., Zhe, S., and Owhadi, H. (2024). Toward efficient kernel-based solvers for nonlinear pdes.

Xu, Z., Long, D., Xu, Y., Yang, G., Zhe, S., and Owhadi, H. (2025). Toward efficient kernel-based solvers for nonlinear pdes. In Forty-second International Conference on Machine Learning. PMLR.

Zhao, Q., Zhou, G., Xie, S., Zhang, L., and Cichocki, A. (2016). Tensor ring decomposition. arXiv preprint arXiv:1606.05535.

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]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

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]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (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]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

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]
   (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]
   (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 PDE Benchmarks

We employed the following PDE benchmarks.

Burger's Equation. The first benchmark is a viscous Burger's equation:

$$\begin{aligned} &u_t + uu_x - \nu \cdot u_{xx} = 0, \qquad \forall (t,x) \in (0,1] \times (-1,1), \\ &u(t,-1) = u(t,1) = 0, \quad u(0,x) = -\sin(\pi x), \end{aligned} \tag{17}$$

where ν is the viscosity. The solution is computed via the Cole-Hopf transformation with quadrature (Chen et al., 2021a). We considered two cases: $\nu = 0.02$ and $\nu = 0.001$.

Nonlinear Elliptic PDE. We then tested on a nonlinear elliptic equation (Chen et al., 2021a),

$$\begin{aligned} &-\Delta u(x_1,x_2) + u^3 = a(x_1,x_2), \quad \forall (x_1,x_2) \in \Omega, \\ &u(x_1,x_2) = 0, \quad \forall (x_1,x_2) \in \partial\Omega, \end{aligned} \tag{18}$$

where $\Omega \in [0,1]^2$, the solution is defined as $u(x_1,x_2) = \sin(\pi x_1)\sin(\pi x_2) + 4\sin(4\pi x_1)\sin(4\pi x_2)$, and the source term a is computed accordingly based on the PDE.

Eikonal PDE. The third is a regularized Eikonal equation (Chen et al., 2021b; Xu et al., 2025),

$$\begin{aligned} &|\nabla u(\mathbf{x})|^2 = g(\mathbf{x})^2 + \epsilon \Delta u(\mathbf{x}), \ \ \forall \mathbf{x} \in \Omega, \\ &u(\mathbf{x}) = 0, \ \ \forall \mathbf{x} \in \partial\Omega, \end{aligned} \tag{19}$$

where $\Omega = [0,1]^2$, $g(\mathbf{x}) = 1$, and $\epsilon = 0.1$. The solution was calculated from a highly-resolved finite difference solver provided by (Chen et al., 2021b).

Allen-Cahn Equation. Fourth, we considered a stationary Allen-Cahn equation with Dirichlet boundary conditions, generalized from the benchmark used in (Xu et al., 2025),

$$\sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} + \gamma(u^m - u) = a(x_1, \ldots, x_d), \tag{20}$$

where $(x_1, \ldots, x_d) \in [0,1]^d$, $\gamma = 1$, $m = 3$, the solution is defined as

$$u = \sum\nolimits_{i=1}^{d} \left(\sin(2\pi\beta x_i)\cos(2\pi\beta x_{(i+1) \bmod d}) + \sin(2\pi x_i)\cos(2\pi x_{(i+1) \bmod d}) \right) \tag{21}$$

and the corresponding source term a is obtained through the equation. Here β controls the frequency of the solution, and we varied $\beta = 15, 20$, and the PDE dimension $d = 2, 4, 6$.

Nonlinear Darcy Flow. Fifth, we employed a 6D nonlinear Darcy flow equation with Dirichlet boundary conditions (Batlle et al., 2025):

$$-\nabla \cdot (c \cdot \nabla u) + u^3 = a(x_1, \cdots, x_6), \tag{22}$$

where each $x_i \in [0,1]$, $c(x_1, \cdots, x_6) = \exp\left(\sin(\sum_{i=1}^{6} \cos(x_i))\right)$, the solution is crafted as $u = \exp\left(\sin\left(\beta \sum_{i=1}^{6} \cos(x_j)\right)\right)$, and a is computed based on the PDE. We set $\beta = 6$, which is more challenging than the case used in (Batlle et al., 2025).

B Method Details

- **SKS**. We used the original JAX implementation[2]. The training is conduced via ADAM optimization with initial learning rate 10^{-3}. The maximum number of iterations was set to 1M. In the training process, SKS does not sample mini-batches of collocation points to compute stochastic gradients. Instead, the full gradient is computed from the training objective at each step, and then fed into ADAM optimizer to update the momentum online and to adjust element-wise step-size. The optimization is stopped if the training objective does not improve for 1K updates. SKS employed Square Exponential (SE) kernel with different length-scales across the input dimensions. Alternative kernels, such as the Matérn kernel, led to inferior performance. The length-scale hyperparameters were selected from a grid search, as detailed in the original paper (Xu et al., 2025). The nugget term was selected from $\{5 \times 10^{-5}, 10^{-5}, 5 \times 10^{-6}, 10^{-6}, \ldots, 10^{-13}\}$.

- **DAKS**. We used the original JAX implementation provided by the authors[3]. The training is performed using relaxed Gauss-Newton optimization. Note that applying the same method to train SKS almost always led to divergence. The kernel was selected from among the squared exponential (SE) kernel and the Matérn kernel with degrees of freedom 3/2 or 5/2. The nugget term was selected from the set $\{5 \times 10^{-5}, 10^{-5}, 5 \times 10^{-6}, 10^{-6}, \ldots, 10^{-13}\}$. Hyperparameters were chosen following the same procedure as for SKS. For solving the nonlinear elliptic PDE with DAKS, however, we employed its default strategy (Chen et al., 2021b), which adaptively assigns nugget values to the two sub-blocks of the Gram matrix; this yielded the best performance for DAKS.

- **PINN**. The network architecture was selected by varying the width and depth over $\{10, 20, \ldots, 100\}$ and $\{2, 3, 5, 8, 10\}$, respectively. The `tanh` activation function was used. The weight of the boundary loss, λ_b, was selected from $\{1, 100, 500, 1000\}$. Training PINNs involved two stages: the first consisted of 10K ADAM epochs with an initial learning rate of 10^{-3}, followed by L-BFGS optimization until convergence, with the tolerance 10^{-9} and the maximum number of iterations as 50K.

- **TGPS**. For our method with CP decomposition, we varied the rank R in each dimension over $\{5, 10, 12, 15, 18, 20, 25\}$. For the TR decomposition, we set $R_0 = \cdots = R_d = R$ and selected R from $\{3, 4, 5, 6, 7\}$. The number of inducing points for the GP components in each dimension was tuned within the range 20–720. Specifically, we first performed a random search to identify a promising configuration, followed by a grid search for refinement. The inducing points were equally spaced and kept fixed during training. For the factor function in each dimension, we chose kernel functions from the Squared Exponential (SE) and Matérn families with degrees of freedom 3/2 or 5/2. The length-scale parameters were selected from {0.005:0.001:0.009, 0.01:0.01:0.1, 0.1:0.1:1.0, 1:1:8}, while nugget values were drawn from $\{10^{-11}, 10^{-10}, 10^{-9}, 10^{-6}\}$ to ensure numerical stability. The regularization parameters α_1 and α_2 in (8) were chosen from $\{10^0, 10^1, 10^2, \ldots, 10^9, 10^{10}\}$.

C Proof of Lemma 4.1 and 4.2

Definition C.1 (Sobolev Space and Weighed Sobolev Space)**.** Let $\Omega \subset \mathbb{R}^d$ be an open subset, and $k \in \mathbb{N}$. The Sobolev space $H^k(\Omega)$ is defined as:

$$H^k(\Omega) := \left\{ g \in L^2(\Omega) \,\middle|\, \partial^{\boldsymbol{\alpha}} g \in L^2(\Omega),\ \forall\, |\boldsymbol{\alpha}| \le k \right\},$$

where $L^2(\Omega)$ is the space of square-integrable functions and $\partial^{\boldsymbol{\alpha}} g$ denotes the weak derivative of g of multi-index $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$, with total order $|\boldsymbol{\alpha}| = \sum_i \alpha_i \le k$. For $\mathbf{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d_+$, the weighted Sobolev space $H^k_{\mathbf{v}}(\Omega) \subseteq H^k(\Omega)$ is defined as

$$H^k_{\mathbf{v}}(\Omega) = \left\{ g \in H^k(\Omega) : \|\partial^{\boldsymbol{\alpha}} g\|_{L^2(\Omega)} \lesssim (v_j)^k \, \|g\|_{H^k(\Omega)} \text{ for } |\boldsymbol{\alpha}| = k \text{ and } j = 1, \ldots, d \right\}. \tag{23}$$

Proof. The proof of Lemma 4.1 is based on the existing results for the Tucker decomposition in (Griebel and Harbrecht, 2023). In particular, (Griebel and Harbrecht, 2023, Theorem 2) states that for the Tucker format,

$$u(x_1, \ldots, x_d) = \sum_{r_1=1}^{R_1} \cdots \sum_{r_d=1}^{R_d} w_{r_1 \ldots r_d} \cdot f^1_{r_1}(x_1) \ldots f^d_{r_d}(x_d), \tag{24}$$

[2] `https://github.com/BayesianAIGroup/Efficient-Kernel-PDE-Solver`
[3] `https://github.com/yifanc96/NonLinPDEs-GPsolver`

Table 3: Relative L^2 error of solving *more difficult* PDEs with *a small number* of collocation points. The grids used in SKS are the same as in Table 1 of the main paper.

(a) Burgers' equation (17) with viscosity $\nu = 0.001$.

Method	600	1200	2400	4800
DAKS	6.30E-01	5.08E-01	5.86E-01	3.86E-01
PINN	**2.07E-01**	4.22E-01	5.18E-01	4.31E-01
SKS	2.18E-01	1.81E-01	**1.31E-01**	**3.08E-02**
TGPS-PF	**1.03E-01**	**7.30E-02**	**8.52E-02**	**5.18E-02**
TGPS-NT	3.53E-01	**1.78E-01**	1.70E-01	2.26E-01

(b) 2D Allen-Cahn equation (20): $a = 15, d = 2$.

Method	600	1200	2400	4800
DAKS	9.67E-01	9.36E-01	8.88E-01	8.12E-01
PINN	5.69E0	8.77E0	6.03E0	7.62E0
SKS	9.62E-01	2.97E-01	7.28E-03	1.30E-04
TGPS-PF	**6.43E-01**	**2.66E-04**	**8.39E-05**	**4.92E-06**
TGPS-NT	**6.42E-01**	**3.79E-04**	**4.36E-05**	**8.64E-06**

(c) 2D Allen-Cahn equation (20): $a = 20, d = 2$.

Method	600	1200	2400	4800
DAKS	9.63E-01	9.29E-01	8.76E-01	7.98E-01
PINN	7.04E0	8.18E0	8.30E0	4.30E0
SKS	1.00E0	9.77E-01	2.56E-01	1.39E-03
TGPS-PF	**6.74E-01**	**1.53E-01**	**2.73E-03**	**5.39E-05**
TGPS-NT	**6.51E-01**	**9.55E-02**	**1.32E-03**	**6.75E-05**

if we choose $R_1 = \cdots = R_d = (\frac{\sqrt{d}}{\varepsilon})^{1/k}$, then the error is $\lesssim \varepsilon$. Here $\{f^i_{r_i}(\cdot)\}_{1\le r_i\le R_i, 1\le i\le d}$ are a collection of one-dimensional functions. We can rewrite the Tucker format as follows,

$$\begin{aligned}
&\sum_{r_1=1}^{R_1}\cdots\sum_{r_d=1}^{R_d} w_{r_1\ldots r_d}\cdot f^1_{r_1}(x_1)\ldots f^d_{r_d}(x_d)\\
&=\sum_{r_1=1}^{R_1} f^1_{r_1}(x_1)\ldots\sum_{r_{d-1}=1}^{R_{d-1}} u^{d-1}_{r_{d-1}}(x_{d-1})\left(\sum_{r_d=1}^{R_d} w_{r_1\ldots r_d}\cdot f^{R_d}_{r_d}(x_d)\right),
\end{aligned}$$

which can be viewed as CP format that includes a summation of $\overline{R} = \prod_{i=1}^{d-1} R_i = (\frac{\sqrt{d}}{\varepsilon})^{\frac{(d-1)}{k}}$ products of rank-one functions. This establishes the first part of Lemma 4.1.

The proof of Lemma 4.2 builds on (Griebel and Harbrecht, 2023, Theorem 3). In particular, the approximation results apply to the TT format (Oseledets, 2011) by invoking (Griebel and Harbrecht, 2023, Theorems 4-5 and Remark 2), where we specialize to the case in which each factor function has input dimension one. Since the TT format is a special case of the TR format with $R_0 = R_d = 1$, these approximation results carry over to the TR format as well. □

D Bound of RKHS Norm

To prove Lemma 4.4, we first prove the following RKHS norm bound.

Lemma D.1. *Let $\mathcal{G}^1(\Omega_0), \ldots, \mathcal{G}^d(\Omega_0)$ be a collection of RKHS's defined on $\Omega_0 \subset \mathbb{R}$, and let $\mathcal{U} = \mathcal{G}^1 \otimes \ldots \otimes \mathcal{G}^d$ denote their tensor-product RKHS. For any function of the form*

$$u(x_1, \ldots, x_d) = \sum_{r=1}^{R}\prod_{i=1}^{d} f^i_r(x_i), \quad f^i_r \in \mathcal{G}^i,$$

we have

$$\|u\|_{\mathcal{U}} \le \left[\frac{1}{d}\sum_{i=1}^{d}\sum_{r=1}^{R}\|f^i_r\|^2_{\mathcal{G}^i}\right]^{d/2}. \tag{25}$$

Proof. We first obtain the inner product in $\mathcal{U}$. Since $\mathcal{U}$ is a tensor-product RKHS, the kernel associated with $\mathcal{U}$ is the product of the kernels associated with each $\mathcal{G}^i$. Therefore, given arbitrary two functions $q^1 \otimes \ldots \otimes q^d$ and $g^1 \otimes \ldots \otimes g^d$ where each $q^i, g^i \in \mathcal{G}^i$, their inner product under $\mathcal{U}$ is defined as

$$\langle q^1 \otimes \ldots \otimes q^d, g^1 \otimes \ldots \otimes g^d \rangle_{\mathcal{G}} = \langle q^1, g^1 \rangle_{\mathcal{G}^1} \cdots \langle q^d, g^d \rangle_{\mathcal{G}^d}. \tag{26}$$

This inner product further extends to the sum of tensor products (*i.e.,* the CP format):

$$\begin{aligned} u = \sum_r \bigotimes_{i=1}^d f_r^i, \quad q = \sum_l \bigotimes_{i=1}^d g_l^i \\ \langle u, q \rangle_{\mathcal{U}} = \sum_r \sum_l \prod_{i=1}^d \left\langle f_r^i, g_l^i \right\rangle_{\mathcal{G}^i}. \end{aligned} \tag{27}$$

According to (27), we have

$$\|u\|_{\mathcal{U}}^2 = \sum_{r=1}^R \sum_{l=1}^R \prod_{i=1}^d \left\langle f_r^i, f_l^i \right\rangle_{\mathcal{G}^i}. \tag{28}$$

Let us define $R \times R$ matrices $\mathbf{A}^i$ where each element $A_{rl}^i = \langle f_r^i, f_l^i \rangle_{\mathcal{G}^i}$. Then

$$\|u\|_{\mathcal{U}}^2 = \sum_r \sum_l \prod_{i=1}^d A_{rl}^i = \left\langle \circ_{i=1}^{d-1} \mathbf{A}^i, \mathbf{A}^d \right\rangle_F \tag{29}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product, $\circ$ is the Hadamard (element-wise) product. Leveraging Cauchy-Schwarz inequality under Frobenius inner product, we have

$$\|u\|_{\mathcal{U}}^2 = \left\langle \circ_{i=1}^{d-1} \mathbf{A}^i, \mathbf{A}^d \right\rangle_F \leq \| \circ_{i=1}^{d-1} \mathbf{A}^i \|_F \cdot \|\mathbf{A}^d\|_F. \tag{30}$$

Since in general $\|\mathbf{A} \circ \mathbf{B}\|_F \leq \|\mathbf{A}\|_F \cdot \|\mathbf{B}\|_F$, we have

$$\| \circ_{i=1}^{d-1} \mathbf{A}^i \|_F \leq \prod_{i=1}^{d-1} \|\mathbf{A}^i\|_F, \tag{31}$$

and therefore

$$\|u\|_{\mathcal{U}}^2 \leq \prod_{i=1}^d \|\mathbf{A}^i\|_F. \tag{32}$$

For each $\mathbf{A}^i$, we have

$$\begin{aligned} \|\mathbf{A}^i\|_F^2 &= \sum_{r=1}^R \sum_{l=1}^R \left\langle f_r^i, f_l^i \right\rangle_{G^i}^2 \\ &\leq \sum_{r=1}^R \sum_{l=1}^R \|f_r^i\|_{\mathcal{G}^i}^2 \cdot \|f_l^i\|_{\mathcal{G}^i}^2 \qquad \text{(Cauchy-Schwarz Inequality)} \\ &= \left[\sum_{r=1}^R \|f_r^i\|_{\mathcal{G}^i}^2 \right]^2. \end{aligned} \tag{33}$$

Combining with (32), we obtain

$$\|u\|_{\mathcal{U}}^2 \leq \prod_{i=1}^d \left(\sum_{r=1}^R \|f_r^i\|_{\mathcal{G}^i}^2 \right) \tag{34}$$

We then leverage the AM–GM inequality (Arithmetic Mean-Geometric Mean inequality): for any $a_1, \ldots, a_n \geq 0$, $(a_1 \cdots a_n)^{1/n} \leq \frac{1}{n}(a_1 + \ldots + a_n)$, and so $a_1 \cdots a_n \leq \left[\frac{1}{n}(a_1 + \ldots + a_n)\right]^n$. Therefore, we obtain

$$\|u\|_{\mathcal{U}} \leq \left[\frac{1}{d}\sum_{i=1}^{d}\sum_{r=1}^{R}\|f_r^i\|_{\mathcal{G}^i}^2\right]^{d/2}. \tag{35}$$

□

E Proof of Lemma 4.4

Proof. The proof consists of the following steps.

Step 1. First, we show that given an arbitrarily small $\varepsilon > 0$, there exists a rank R and a set of one-dimensional factor functions $\{f_r^i \in \mathcal{G}^i\}_{1\leq r\leq R}$ in each dimension i ($1 \leq i \leq d$), such that their combination via CP decomposition (4), denoted as

$$\widehat{u} = \sum_{r=1}^{R}\prod_{i=1}^{d}\hat{f}_r^i(x_i), \tag{36}$$

satisfies

$$\|\widehat{u} - u^*\|_{\mathcal{U}} \leq \varepsilon. \tag{37}$$

To show this, denote the associated kernel with each $\mathcal{G}^i$ as κ_i. Since $\mathcal{U} = \mathcal{G}^1 \otimes \ldots \otimes \mathcal{G}^d$, the kernel inducing $\mathcal{U}$ is therefore $\kappa(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{d}\kappa_i(x_i, x_i')$. Since each κ_i is universal, the product kernel κ is also universal on the product domain. As a result, if we denote the eigenfunctions of k_i as $\{\phi_j^i(\cdot)\}_{j=1}^{\infty}$ and the eigenvalues as $\{\lambda_j^i\}_{j=1}^{\infty}$ (note that all $\lambda_j^i > 0$), then we have $\Phi_{j_1\ldots j_d}(x_1, \ldots, x_d) \coloneqq \phi_{j_1}^1(x_1)\cdots\phi_{j_d}^d(x_d)$ constitute orthonormal bases in $L^2(\Omega)$. We can represent the true PDE solution as

$$u^*(x_1, \ldots, x_d) = \sum_{j_1,\ldots,j_d=1}^{\infty}\langle u^*, \Phi_{j_1\ldots j_d}\rangle \cdot \Phi_{j_1\ldots j_d}(x_1, \ldots, x_d),$$

where the dot product $\langle\cdot,\cdot\rangle$ is defined in $L^2(\Omega)$. Since $\kappa = \otimes_{i=1}^{d}\kappa_i$, $\{\Phi_{j_1\ldots j_d}\}$ and $\{\prod_{i=1}^{d}\lambda_{j_i}^i\}$ form the eigenfunctions and eigenvalues of κ, respectively. Because $u^* \in \mathcal{U}$, we have

$$\|u^*\|_{\mathcal{U}}^2 \coloneqq \sum_{j_1,\ldots,j_d=1}^{\infty}\frac{\langle u^*, \Phi_{j_1\ldots j_d}\rangle^2}{\lambda_{j_1}^1\cdots\lambda_{j_d}^d} < \infty.$$

Consequently, for any $\epsilon > 0$, there exists a sufficiently large $I(\epsilon)$, such that

$$\|u^* - \sum_{j_1,\ldots,j_d\leq I(\varepsilon)}\langle u, \Phi_{j_1\ldots j_d}\rangle\Phi_{j_1\ldots j_d}\|_{\mathcal{U}} < \epsilon.$$

This truncation can be expressed as

$$\sum_{j_1=1}^{I(\varepsilon)}\phi_{j_1}^1(x_1)\sum_{j_2=1}^{I(\varepsilon)}\phi_{j_2}^2(x_2)\ldots\sum_{j_{d-1}=1}^{I(\varepsilon)}\phi_{j_{d-1}}^{d-1}(x_d-1)\left(\sum_{j_d=1}^{I(\varepsilon)}\phi_{j_d}^d(x_d)\langle u^*, \Phi_{j_1\ldots j_d}\rangle\right), \tag{38}$$

which can be viewed as a CP decomposition form in (36), with rank $R = I(\varepsilon)^{d-1}$. We map each multi-index $(j_1, \cdots, j_{d-1})$ with $j_i \leq I(\varepsilon)$, to an index $r \in \{1, \cdots, R\}$. For each such r, denote the corresponding tuple by $(j_{r_1}, \ldots, j_{r_{d-1}})$. Then we set $\hat{f}_r^i = \phi_{j_{r_i}}^i$ for $i < d$, and $\hat{f}_r^d = \sum_{j_d=1}^{I(\varepsilon)}\phi_{j_d}^d(x_d)\langle u^*, \Phi_{j_{r_1}\ldots j_{r_{d-1}}j_d}\rangle$. Clearly, each $\hat{f}_r^i \in \mathcal{G}^i$, and the approximation $\widehat{u} = \sum_{r=1}^{R}\prod_{i=1}^{d}\hat{f}_r^i(x_i)$ satisfies that $\|\widehat{u} - u^*\|_{\mathcal{U}} \leq \varepsilon$.

Step 2. Next, we show that with rank R and an appropriate choice of δ, the optimization problem (7) using the CP form is feasible; that is, a solution exists.

Denote by $\mathcal{M}^i$ the projection of the collocation set $\mathcal{M}$ onto the i-th coordinate axis: $\mathcal{M}^i = \{[\mathbf{x}_m]_i : \mathbf{x}_m \in \mathcal{M}\}$, *i.e.,* the set of all distinct coordinates of the collocation points along dimension i. we first construct a set of intermediate optimization problems. Each problem $\mathcal{Z}_r^i$ $(1 \le i \le d, 1 \le r \le R)$ is defined as:

$$\begin{cases} \underset{f_r^i \in \mathcal{G}^i}{\text{minimize}} \ \ \|f_r^i\|_{\mathcal{G}^i} \\ \text{s.t. } f_r^i(x_m) = \hat{f}_r^i(x_m), \ \ x_m \in \mathcal{M}^i, \end{cases} \tag{39}$$

where $\hat{f}_r^i$ is from the approximation $\widehat{u}$ in (36). This is a standard kernel regression problem. Let us denote the minimizer of (39) by $\hat{f}_{r\mathcal{M}}^i$. According to the optimal recovery theorem (Owhadi and Scovel, 2019), we have $\hat{f}_{r\mathcal{M}}^i$ takes the kernel interpolation form (6), and

$$\|\hat{f}_{r\mathcal{M}}^i\|_{\mathcal{G}^i} \le \|\hat{f}_r^i\|_{\mathcal{G}^i}. \tag{40}$$

Let us define

$$\widehat{u}_{\mathcal{M}} = \sum_{r=1}^{R} \prod_{i=1}^{d} \hat{f}_{r\mathcal{M}}^i(x_i). \tag{41}$$

Obviously, $\widehat{u}_{\mathcal{M}}(\mathbf{x}_m) - \widehat{u}(\mathbf{x}_m) = 0$ for every $\mathbf{x}_m \in \mathcal{M}_\Omega$. According to the sampling inequality (Proposition A.1 of (Batlle et al., 2023)), when h is sufficiently small (note that the fill-in distance $h_\Omega \le h$),

$$\|\widehat{u}_{\mathcal{M}} - \widehat{u}\|_{H^s(\Omega)} \lesssim h^\tau \|\widehat{u}_{\mathcal{M}} - \widehat{u}\|_{H^{s+\tau}(\Omega)}. \tag{42}$$

We now consider bounding

$$\mathcal{L} = \|\mathcal{P}(\widehat{u}_{\mathcal{M}}) - \mathcal{P}(u^*)\|_{H^k(\Omega)} + \|\mathcal{B}(\widehat{u}_{\mathcal{M}}) - \mathcal{B}(u^*)\|_{H^t(\partial\Omega)}. \tag{43}$$

Using the norm triangle inequality, we have

$$\begin{aligned} \mathcal{L} \le & \|\mathcal{P}(\widehat{u}_{\mathcal{M}}) - \mathcal{P}(\widehat{u})\|_{H^k(\Omega)} + \|\mathcal{P}(\widehat{u}) - \mathcal{P}(u^*)\|_{H^k(\Omega)} \\ & + \|\mathcal{B}(\widehat{u}_{\mathcal{M}}) - \mathcal{B}(\widehat{u})\|_{H^t(\partial\Omega)} + \|\mathcal{B}(\widehat{u}) - \mathcal{B}(u^*)\|_{H^t(\partial\Omega)} \end{aligned} \tag{44}$$

Combining (44) with the PDE stability (16) in Assumption (4.3) and the result (42), we obtain

$$\mathcal{L} \lesssim h^\tau \cdot \|\widehat{u}_{\mathcal{M}} - \widehat{u}\|_{H^{s+\tau}(\Omega)} + \|\widehat{u} - u^*\|_{H^s(\Omega)}. \tag{45}$$

Since $\mathcal{U} \hookrightarrow H^{s+\tau}(\Omega)$ (C3 of Assumption 4.3) and $H^{s+\tau}(\Omega) \hookrightarrow H^s(\Omega)$, we have

$$\|\widehat{u}_{\mathcal{M}} - \widehat{u}\|_{H^{s+\tau}(\Omega)} \lesssim \|\widehat{u}_{\mathcal{M}} - \widehat{u}\|_{\mathcal{U}}, \quad \|\widehat{u} - u^*\|_{H^s(\Omega)} \lesssim \|\widehat{u} - u^*\|_{\mathcal{U}}. \tag{46}$$

Combining with (45), we further derive that

$$\begin{aligned} \mathcal{L} & \lesssim h^\tau \cdot \|\widehat{u}_{\mathcal{M}} - \widehat{u}\|_{\mathcal{U}} + \|\widehat{u} - u^*\|_{\mathcal{U}} \\ & \lesssim h^\tau \cdot \|\widehat{u}_{\mathcal{M}}\|_{\mathcal{U}} + h^\tau \cdot \|\widehat{u}\|_{\mathcal{U}} + \varepsilon. \qquad \text{(see (37))} \end{aligned} \tag{47}$$

Leveraging the RKHS norm bound (25) in Lemma D.1, we obtain that

$$\begin{aligned} \mathcal{L} & \le C \left(h^\tau \left(\frac{1}{d} \sum_{i=1}^{d} \sum_{r=1}^{R} \|\hat{f}_{r\mathcal{M}}^i\|_{\mathcal{G}^i}^2 \right)^{d/2} + h^\tau \left(\frac{1}{d} \sum_{i=1}^{d} \sum_{r=1}^{R} \|\hat{f}_r^i\|_{\mathcal{G}^i}^2 \right)^{d/2} + \varepsilon \right) \\ & \le C \left(h^\tau \cdot 2 \left(\frac{1}{d} \sum_{i=1}^{d} \sum_{r=1}^{R} \|\hat{f}_r^i\|_{\mathcal{G}^i}^2 \right)^{d/2} + \varepsilon \right), \qquad \text{(according to (40))} \end{aligned} \tag{48}$$

where C is a constant independent of terms on both sides of the inequality. Note that each $\hat{f}_r^i \in \mathcal{G}^i$ and can be constructed from the eigenfunctions of κ_i that is universal —see (38) , therefore $\|\hat{f}_r^i\|_{\mathcal{G}^i}$ is a bounded constant partly determined by ε.

Meanwhile, because $H^k(\Omega) \hookrightarrow C^0(\Omega)$ and $H^t(\partial\Omega) \hookrightarrow C^0(\partial\Omega)$, we have

$$\begin{aligned}\|\mathcal{P}(\widehat{u}_{\mathcal{M}}) - \mathcal{P}(u^*)\|_{C^0(\Omega)} &\lesssim \|\mathcal{P}(\widehat{u}_{\mathcal{M}}) - \mathcal{P}(u^*)\|_{H^k(\Omega)},\\ \|\mathcal{B}(\widehat{u}_{\mathcal{M}}) - \mathcal{B}(u^*)\|_{C^0(\partial\Omega)} &\lesssim \|\mathcal{B}(\widehat{u}_{\mathcal{M}}) - \mathcal{B}(u^*)\|_{H^t(\partial\Omega)}.\end{aligned} \tag{49}$$

In addition, at any collocation point $\mathbf{x}_m$,

$$\begin{aligned}\left(\mathcal{P}(\widehat{u}_{\mathcal{M}})(\mathbf{x}_m) - \mathcal{P}(u^*)(\mathbf{x}_m)\right)^2 &\le \|\mathcal{P}(\widehat{u}_{\mathcal{M}}) - \mathcal{P}(u^*)\|^2_{C^0(\Omega)},\\ \left(\mathcal{B}(\widehat{u}_{\mathcal{M}})(\mathbf{x}_m) - \mathcal{B}(u^*)(\mathbf{x}_m)\right)^2 &\le \|\mathcal{B}(\widehat{u}_{\mathcal{M}}) - \mathcal{B}(u^*)\|^2_{C^0(\partial\Omega)}.\end{aligned} \tag{50}$$

Combining (49), (50) and (48), we therefore obtain that

$$\begin{aligned}&\frac{1}{M_\Omega}\sum_{m=1}^{M_\Omega}\left(\mathcal{P}(\widehat{u}_{\mathcal{M}})(\mathbf{x}_m) - \mathcal{P}(u^*)(\mathbf{x}_m)\right)^2 + \frac{1}{M - M_\Omega}\sum_{m=M_\Omega+1}^{M}\left(\mathcal{B}(\widehat{u}_{\mathcal{M}})(\mathbf{x}_m) - \mathcal{B}(u^*)(\mathbf{x}_m)\right)^2\\ &\le \mathcal{L}^2 \le C^2\left(h^\tau \cdot 2\left(\frac{1}{d}\sum_{i=1}^{d}\sum_{r=1}^{R}\|\hat{f}_r^i\|^2_{\mathcal{G}^i}\right)^{d/2} + \varepsilon\right)^2.\end{aligned} \tag{51}$$

Therefore, if we set

$$\delta = C\left(h^\tau \cdot 2\left(\frac{1}{d}\sum_{i=1}^{d}\sum_{r=1}^{R}\|\hat{f}_r^i\|^2_{\mathcal{G}^i}\right)^{d/2} + \varepsilon\right) \tag{52}$$

in (7), $\widehat{u}_{\mathcal{M}}$ is at least a feasible solution, and the optimization problem (7) is feasible.

Step 3. Let us denote by $u^\dagger$ the solution of problem (7). We then analyze the error of $u^\dagger$. Using an idea similar to (Xu et al., 2025), we define two error functions,

$$\begin{aligned}\xi_P(\mathbf{x}) &= \mathcal{P}(u^\dagger)(\mathbf{x}) - \mathcal{P}(u^*)(\mathbf{x}), \quad \mathbf{x} \in \Omega\\ \xi_B(\mathbf{x}) &= \mathcal{B}(u^\dagger)(\mathbf{x}) - \mathcal{B}(u^*)(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega.\end{aligned} \tag{53}$$

Our goal is to bound the L^2 norm of the error functions: $\|\xi_P\|_{H^0(\Omega)}$ and $\|\xi_B\|_{H^0(\partial\Omega)}$. Let us first consider ξ_P. To bound $\|\xi_P\|_{H^0(\Omega)}$, we decompose Ω into a Voronoi diagram according to the collocation points, which results in M_Ω regular non-overlapping regions, $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_{M_\Omega} = \Omega$, where each region $\mathcal{T}_i$ only contains one collocation point $\mathbf{x}_i$, and its filled-distance $h_i \lesssim h$ ($1 \le i \le M_\Omega$). Accordingly, we can decompose the squared L^2 norm as

$$\|\xi_P\|^2_{H^0(\Omega)} = \sum_{i=1}^{M_\Omega}\int_{\mathcal{T}_i}\xi_P(\mathbf{x})^2 \mathrm{d}\mathbf{x} = \sum_{i=1}^{M_\Omega}\|\xi_P\|^2_{H^0(\mathcal{T}_i)}. \tag{54}$$

Leveraging the fact that

$$\xi_P(\mathbf{x})^2 = \left(\xi_P(\mathbf{x}) - \xi_P(\mathbf{x}_i) + \xi_P(\mathbf{x}_i)\right)^2 \le 2\left(\xi_P(\mathbf{x}) - \xi_P(\mathbf{x}_i)\right)^2 + 2\xi_P(\mathbf{x}_i)^2,$$

we obtain

$$\|\xi_P\|^2_{H^0(\mathcal{T}_i)} \lesssim \|\xi_P - \xi_P(\mathbf{x}_i)\|^2_{H^0(\mathcal{T}_i)} + \lambda(\mathcal{T}_i)\xi_P(\mathbf{x}_i)^2, \tag{55}$$

where $\lambda(\mathcal{T}_i)$ is the volume of $\mathcal{T}_i$.

The function $\xi_P - \xi_P(\mathbf{x}_i)$ is zero at $\mathbf{x}_i$. Since the aspect ratio of $\mathcal{T}_i$ is bounded, we can apply the sampling inequality — a.k.a Poincaré inequality,

$$\|\xi_P - \xi_P(\mathbf{x}_i)\|_{H^0(\mathcal{T}_i)} \lesssim h_i^k\|\xi_P - \xi_P(\mathbf{x}_i)\|_{H^k(\mathcal{T}_i)} \lesssim h^k\|\xi_P - \xi_P(\mathbf{x}_i)\|_{H^k(\mathcal{T}_i)}. \tag{56}$$

Applying the mean inequality,

$$\|\xi_P - \xi_P(\mathbf{x}_i)\|^2_{H^0(\mathcal{T}_i)} \lesssim h^{2k}\left(\|\xi_P\|^2_{H^k(\mathcal{T}_i)} + \|\xi_P(\mathbf{x}_i)\|^2_{H^k(\mathcal{T}_i)}\right) = h^{2k}\left(\|\xi_P\|^2_{H^k(\mathcal{T}_i)} + \lambda(\mathcal{T}_i)\xi_P(\mathbf{x}_i)^2\right). \tag{57}$$

Since $\lambda(\mathcal{T}_i) \lesssim h^d$, combining (54), (55) and (57), we can obtain

$$\begin{aligned}\|\xi_P\|^2_{H^0(\Omega)} &\lesssim h^{2k} \sum_i \|\xi_P\|^2_{H^k(\mathcal{T}_i)} + (h^d + h^{2k+d}) \sum_i \xi_P(\mathbf{x}_i)^2 \\ &\lesssim h^{2k} \|\xi_P\|^2_{H^k(\Omega)} + (h^d + h^{2k+d}) \cdot M_\Omega \cdot \delta^2, \end{aligned} \tag{58}$$

where δ^2 comes from the constraint of (7). Using a similar approach, we can show that

$$\|\xi_B\|^2_{H^0(\partial\Omega)} \lesssim h^{2t} \|\xi_B\|^2_{H^t(\partial\Omega)} + (h^d + h^{2t+d})(M - M_\Omega)\delta^2. \tag{59}$$

Combining (58) and (59),

$$\left(\|\xi_P\|_{H^0(\Omega)} + \|\xi_B\|_{H^0(\partial\Omega)}\right)^2 \lesssim h^{2\cdot\rho} \left(\|\xi_P\|_{H^k(\Omega)} + \|\xi_B\|_{H^t(\partial\Omega)}\right)^2 + (h^d + h^{2\rho+d}) M\delta^2, \tag{60}$$

where $\rho = \min(k, t)$. When $h \lesssim M^{-\frac{1}{d}}$ and is sufficiently small, we have $(h^d + h^{2\rho+d})M \le 1 + h^{2\rho} \le 2$, and

$$\left(\|\xi_P\|_{H^0(\Omega)} + \|\xi_B\|_{H^0(\partial\Omega)}\right)^2 \lesssim h^{2\cdot\rho} \left(\|\xi_P\|_{H^k(\Omega)} + \|\xi_B\|_{H^t(\partial\Omega)}\right)^2 + 2\delta^2. \tag{61}$$

Using the PDE stability (15) and (16), we obtain

$$\|u^\dagger - u^*\|_{H^l(\Omega)} \lesssim h^\rho \|u^\dagger - u^*\|_{H^s(\Omega)} + \delta. \tag{62}$$

Since $\mathcal{U} \hookrightarrow H^{s+\tau}(\Omega) \hookrightarrow H^s(\Omega)$, we further have

$$\begin{aligned}\|u^\dagger - u^*\|_{H^l(\Omega)} &\lesssim h^\rho \|u^\dagger - u^*\|_{\mathcal{U}} + \delta \\ &\lesssim h^\rho \|u^\dagger - \widehat{u}\|_{\mathcal{U}} + h^\rho \|\widehat{u} - u^*\|_{\mathcal{U}} + \delta \\ &\lesssim h^\rho \|u^\dagger - \widehat{u}\|_{\mathcal{U}} + h^\rho \varepsilon + \delta \\ &\lesssim h^\rho \|u^\dagger\|_{\mathcal{U}} + h^\rho \|\widehat{u}\|_{\mathcal{U}} + h^\rho \varepsilon + \delta \end{aligned} \tag{63}$$

Denote each factor function in $u^\dagger$ as $f_r^{i\dagger}$. According to the RKHS norm bound (25) in Lemma D.1, we have $\|u^\dagger\|_{\mathcal{U}} \le \left(\frac{1}{d}\sum_{i=1}^d \sum_{r=1}^R \|f_r^{i\dagger}\|^2_{\mathcal{G}^i}\right)^{d/2}$. Since $\widehat{u}_{\mathcal{M}}$ is a feasible solution to (7), we must have

$$\begin{aligned}\sum_{i=1}^d \sum_{r=1}^R \|f_r^{i\dagger}\|^2_{\mathcal{G}^i} &\le \sum_{i=1}^d \sum_{r=1}^R \|\hat{f}^i_{r\mathcal{M}}\|^2_{\mathcal{G}^i} \\ &\le \sum_{i=1}^d \sum_{r=1}^R \|\hat{f}^i_r\|^2_{\mathcal{G}^i} \qquad \text{(according to (40))}. \end{aligned} \tag{64}$$

Combining (63), (52) and (64), we obtain

$$\begin{aligned}\|u^\dagger - u^*\|_{H^l(\Omega)} &\lesssim (h^\rho + h^\tau) C_0 + (h^\rho + 1)\varepsilon \\ &\lesssim h^\nu C_0 + (h^\rho + 1)\varepsilon, \end{aligned} \tag{65}$$

where $\nu = \min(\rho, \tau) = \min(k, t, \tau)$, and $C_0 = \left(\frac{1}{d}\sum_{i=1}^d \sum_{r=1}^R \|\hat{f}^i_r\|^2_{\mathcal{G}^i}\right)^{d/2}$. Therefore, when $h \to 0$,

$$\|u^\dagger - u^*\|_{H^l(\Omega)} \lesssim \varepsilon.$$

□

F Proof of Proposition 4.5

We first construct the Lagrange function. The constraint optimization problem (7) is equivalent to the mini-max optimization problem over the Lagrange function,

$$\begin{aligned}\min_{\{f_r^i \in \mathcal{G}^i\}} \max_{\beta \ge 0} \sum_{i=1}^d \sum_{r=1}^R \|f_r^i\|^2 + \beta \Bigg[&\frac{1}{M_\Omega} \sum_{m=1}^{M_\Omega} \left(\mathcal{P}(u)(\mathbf{x}_m) - a(\mathbf{x}_m)\right)^2 - \frac{\delta^2}{2} \\ &+ \frac{1}{M - M_\Omega} \sum_{m=M_\Omega+1}^{M} \left(\mathcal{B}(u)(\mathbf{x}_m) - b(\mathbf{x}_m)\right)^2 - \frac{\delta^2}{2} \Bigg]. \end{aligned} \tag{66}$$

Table 4: Relative L^2 error of conventional numerical solvers and TGPS according to the ground-truth solution.

(a) Nonlinear elliptic PDE (18)

Method	18×18	25×25	35×35	49×49
FEM	1.68E-02	8.61E-03	4.35E-03	2.20E-03
FD	3.02E-02	1.60E-02	8.32E-03	4.30E-03
TGPS-PF	**1.97E-06**	**2.82E-07**	**1.28E-07**	**4.04E-08**
TGPS-NT	**1.78E-06**	**3.52E-07**	**1.74E-07**	**4.06E-08**

(b) The 2D Allen-Cahn equation (20) with $a = 15$.

Method	80×80	90×90	150×150	200×200
FEM	3.32E-02	2.62E-02	9.65E-03	5.42E-03
FD	1.21E-01	9.45E-02	3.30E-02	1.84E-02
TGPS-PF	**6.00E-06**	**1.21E-06**	**4.87E-06**	**1.43E-06**
TGPS-NT	**3.99E-06**	**1.28E-06**	**1.70E-06**	**1.76E-06**

(c) The 2D Allen-Cahn equation (20) with $a = 20$.

Method	80×80	90×90	150×150	200×200
FEM	5.94E-02	4.66E-02	1.71E-02	9.62E-03
FD	2.29E-01	1.75E-01	5.97E-02	3.31E-02
TGPS-PF	**8.50E-06**	**8.47E-06**	**5.90E-06**	**5.14E-06**
TGPS-NT	**9.03E-06**	**7.42E-06**	**5.79E-06**	**4.80E-06**

Suppose the feasible region is non-empty. Let us denote the optimum of (66) as $(\{f_r^{i\dagger}\}, \beta^\dagger)$. Then $\{f_r^{i\dagger}\}$ is a minimizer of (7). If we now set $\alpha_1 = \alpha_2 = \beta^\dagger$ in (8), then optimizing (8) will recover the minimizer $\{f_r^{i\dagger}\}$.

G Numerical Solvers

The P2 Galerkin finite element method (FEM) is implemented using the MATLAB PDE Toolbox[4] with high-order quadratic Lagrange elements and a multi-level mesh refinement strategy. Specifically, we utilized quadratic (P2) Lagrange elements on triangular meshes, which provide third-order convergence rate for smooth solutions. Each element contains nodes at vertices and edge midpoints. The multi-level mesh hierarchy is generated via progressive refinement. For the nonlinear elliptic PDE, we have $h_{\max} \in \{0.055, 0.04, 0.0286, 0.0204, 0.01\}$, and for the Allen-Cahn equations ($a = 15$ and $a = 20$), we have $h_{\max} \in \{0.04, 0.0286, 0.0204, 0.0143\}$. We used the weak formulation to construct and assemble the stiffness matrix, mass matrix, and loading vectors. The resulting nonlinear discretized system was solved using Newton iterations combined with an Armijo line search to guarantee convergence.

The finite difference (FD) scheme discretized each PDE using centered second-order finite differences. The resulting nonlinear system was solved with a Newton-Krylov method, where the inverse of the Jacobian was computed using iterative Krylov subspace techniques.

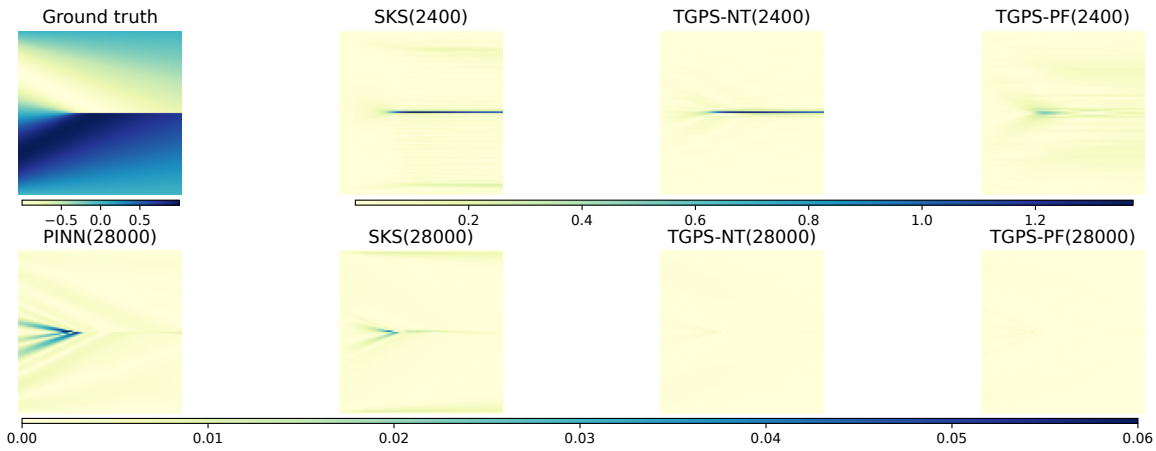
H Point-Wise Error

For a fine-grained evaluation, we examined the point-wise errors of TGPS, SKS, and PINN when solving Burgers' equation (17) with $\nu = 0.001$ and the 2D Allen-Cahn equation (20) with $a = 20$ and $d = 2$. The number of collocation points was varied from $2400, 28$K for Burgers' equation and from $600, 2400, 4800, 8100$ for the Allen-Cahn equation. The point-wise absolute errors are shown in Figure 4.

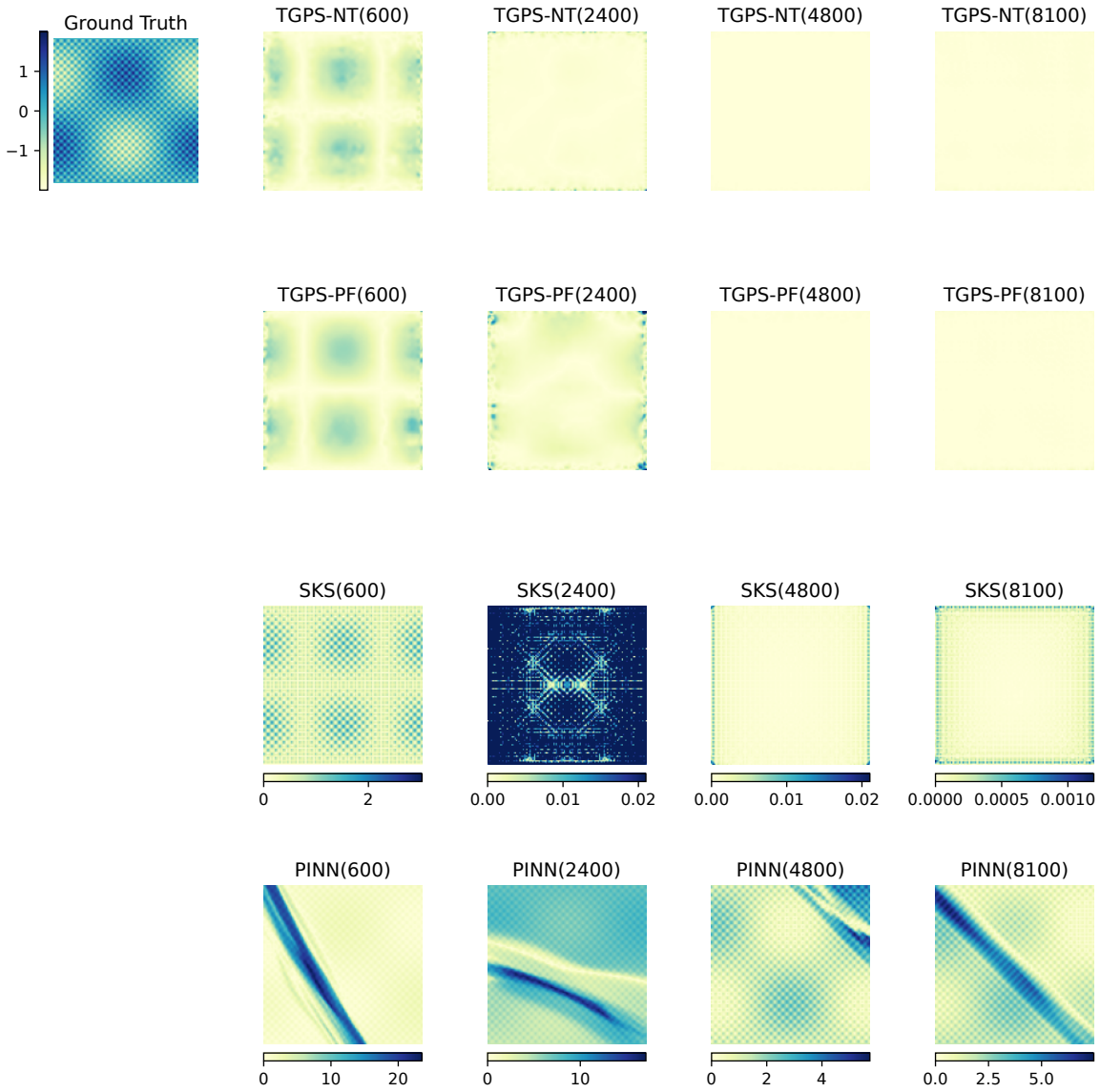
As expected, when the number of collocation points is small, all methods incur larger errors across the domain — for instance, using 2400 points for Burgers' equation or 600 points for the Allen-Cahn equation. Notably, the error of PINN with 2400 collocation points on Burgers' equation was so large that we excluded its error plot in this case. Increasing the number of collocation points substantially improves performance for all methods. Nevertheless, our approach consistently produces smaller errors across the domain. For example, in Figure 4b, when solving the 2D Allen-Cahn equation with 2400 collocation points, SKS exhibits large errors throughout the domain, while both TGPS-PF and TGPS-NT confine relatively larger errors only near the boundary.

These results demonstrate that our method not only improves global solution accuracy but also achieves lower local error.

[4] `https://www.mathworks.com/products/pde.html`



(a) Burger's equation (17) with $\nu = 0.001$.



(b) 2D Allen-Cahn (20) ($a = 20, d = 2$).

Figure 4: Point-wise error. Inside each parenthesis is the number of collocation points.

Table 5: Relative L^2 error of solving PDEs on *irregularly shaped* domains.

(a) 2D Allen-Cahn equation (20): $a = 15, d = 2$.

Shape	PINN	SKS	TGPS-PF	TGPS-NT
Triangle	4.97E0	4.08E-03	2.85E-06	4.80E-06
Circle	7.48E0	4.50E-03	1.72E-06	1.57E-06

(b) Nonlinear elliptic PDE (18).

Shape	PINN	SKS	TGPS-PF	TGPS-NT
Triangle	4.10E-4	3.27E-04	3.98E-08	4.29E-08
Circle	2.66E-04	4.47E-04	3.68E-08	3.31E-08

I Irregular Domains

We evaluated performance on the nonlinear elliptic PDE (18) and the 2D Allen-Cahn equation (20) ($a = 15, d = 2$) over two irregular domains: (i) an inscribed circle within $[0, 1] \times [0, 1]$, and (ii) a triangle with vertices at $(0, 0)$, $(1, 0)$, and $(0.5, 1)$. The reference solutions follow Section A, with boundary conditions derived accordingly. Competing baselines included SKS and PINN, all tested with 10K collocation points. For fairness, each method also used the same 396 uniformly sampled boundary points. Since SKS is restricted to regular domains, we embedded each irregular domain into a 100×100 regularly-spaced virtual grid over $[0, 1]^2$. In contrast, TGPS and PINN directly operated on (the same set of) 10K randomly sampled collocation points from the irregular domains (including the 396 boundary points).

The relative L^2 errors are summarized in Table 5. As shown, PINN again failed on the Allen-Cahn equation, yielding a relative L^2 error larger than one, likely due to the spectral bias. While SKS remained reasonably accurate, its errors deteriorated by several orders of magnitude compared to regular domains. For example, when solving the nonlinear elliptic PDE, SKS achieved errors on the order of 10^{-6} with a 49×49 grid on $[0, 1]^2$ (see Table 1b in the main paper), but errors increased to 10^{-4} on the circle and triangle domains even with a denser 100×100 (virtual) grid. Similarly, for the 2D Allen-Cahn equation, SKS reached 10^{-6} error on the rectangular domain (see Table 2b in the main paper), but only 10^{-3} on the irregular domains.

By contrast, TGPS consistently attained errors on the order of 10^{-8} (elliptic PDE) and 10^{-6} (Allen–Cahn) on both irregular domains — matching its performance on regular domains. Notably, PINN also maintained its error level on the elliptic PDE. Overall, these results highlight the robustness of our mesh-free solver: its accuracy remains stable regardless of domain geometry. The pointwise error plots in Figure 5 and 6 further corroborate this conclusion.

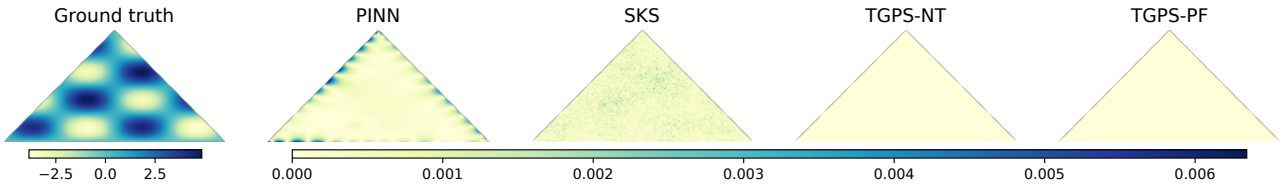
J Ablation Studies

Furthermore, we conducted ablation studies to evaluate the influence of two important types of hyperparameters in our model: kernel parameters and the number of factor functions (i.e., rank). For this purpose, we employed Burgers' equation (17) with $\nu = 0.02$ and the 2D Allen-Cahn equation (20) with $a = 15$.

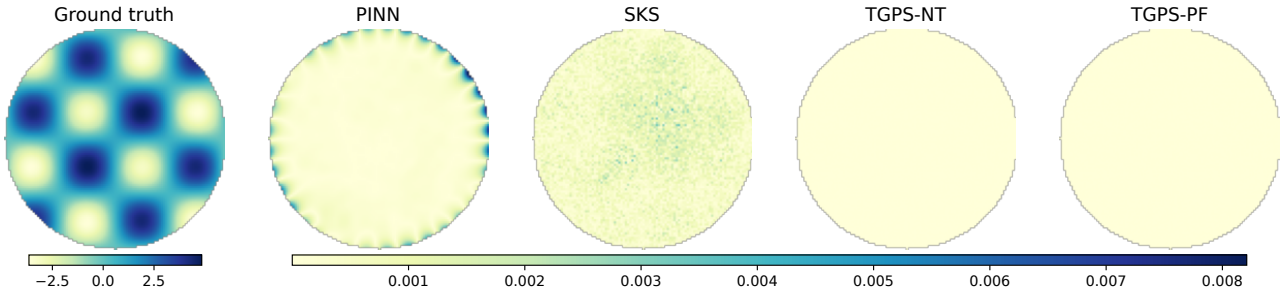
Kernel Hyperparameters. We first examined the effect of kernel length-scale parameters. For Burgers' equation ($\nu = 0.02$), we fixed the spatial length-scale to 0.04 and varied the temporal length-scale over $\{0.001, 0.01, 0.5, 1.0, 2.0\}$, keeping the number of factor functions consistent with our main experiments (Table 1). As shown in Table 6a, both TGPS-PF and TGPS-NT are highly sensitive to the spatial length-scale, achieving the lowest relative L^2 error when it is set to 0.5. Deviations in either direction caused orders-of-magnitude error growth.

Next, we fixed the temporal length-scale at 0.2 and varied the spatial length-scale over $\{0.001, 0.01, 0.1, 0.5, 1.0\}$. The results (Table 6b) reveal the same pattern: optimal performance occurs for intermediate values, while smaller or larger scales lead to substantial degradation. Finally, we tested our method on the 2D Allen-Cahn equation with identical length-scales across both spatial dimensions, varying the parameter over $\{0.001, 0.01, 0.05, 0.1, 0.2\}$. As shown in Table 6c, the smallest error arises at 0.05, with larger or smaller values again producing error increases by orders of magnitude. Collectively, these results underscore the critical role of length-scale parameters in determining model performance.

From a theoretical standpoint, the kernel and its hyperparameters determine the GP prior and thus the associated RKHS. The closer this RKHS matches the regularity class and characteristic scales of the true PDE solution, the better the approximation quality and stability. Hyperparameter tuning can therefore be viewed as selecting an RKHS whose inductive bias is well

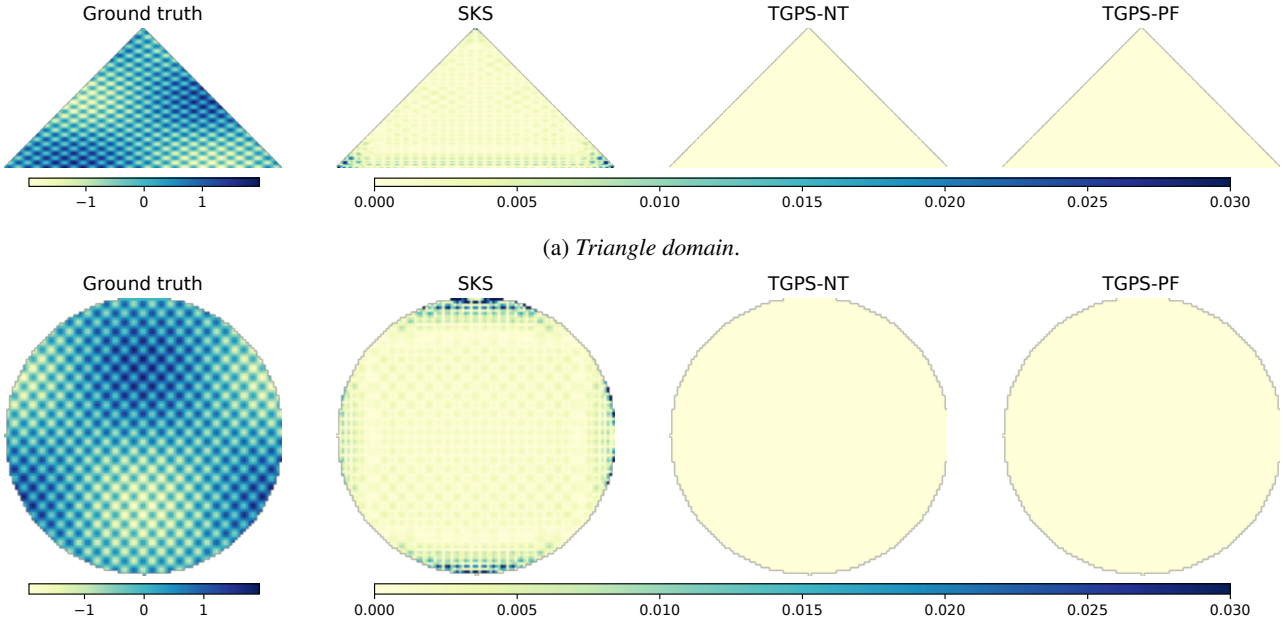


(a) *Triangle domain*.



(b) *Circle domain*.

Figure 5: Solving the nonlinear elliptic PDE on irregular domains. The first column shows the ground-truth while the remaining columns the point-wise error of each method.



(a) *Triangle domain*.

(b) *Circle domain*.

Figure 6: Solving 2D Allen-Cahn equation ($a = 15$) on irregular domains. The first column shows the ground-truth while the remaining columns the point-wise error of each method.

Table 6: Relative L^2 error of TGPS with different length-scales.

(a) Solving Burgers' equation (17) with viscosity $\nu = 0.02$. The number of collocation point is 2400 and the spatial length-scale is fixed to 0.04.

Temporal length-scale	0.001	0.01	0.5	1.0	2.0
TGPS-PF	9.98E-01	8.60E-01	3.93E-03	1.17E-02	2.17E-02
TGPS-NT	1.01E-0	9.54E-01	3.80E-03	1.25E-02	2.42E-02

(b) Solving Burgers' equation with viscosity $\nu = 0.02$. The number of collocation point is 2400 and the temporal length-scale is fixed to 0.2.

Spatial length-scale	0.001	0.01	0.1	0.5	1.0
TGPS-PF	1.00E0	9.66E-01	1.32E-02	1.86E-02	2.73E-01
TGPS-NT	1.44E0	9.23E-01	4.07E-02	2.56E-01	3.23E-01

(c) Solving 2D Allen-Cahn equation (20) with $a = 15$. The number of collocation points is 4800. The length-scales are the same for both spatial dimensions.

Length-scale	0.001	0.01	0.05	0.1	0.2
TGPS-PF	1.00E0	1.04E0	8.69E-04	9.99E-01	1.25E0
TGPS-NT	1.00E0	1.04E0	5.79E-04	4.01E0	7.70E0

aligned with the target solution. While there is no universal rule for kernel and hyperparameter selection, we have found several useful heuristics that guide kernel and length-scale choices:

- Smooth solutions (*e.g.*, moderate-viscosity Burgers', elliptic PDE): Gaussian kernels generally perform well.
- Solutions with sharp gradients or low regularity (*e.g.*, near-shocks in Burgers' with small viscosity): less smooth kernels such as Matérn-2/3 are often more appropriate.
- Higher-frequency structure: Smaller length-scales help capture oscillatory behavior.

For instance, in Burgers' equation, viscosity 0.02 was well modeled using a Gaussian kernel in space, whereas viscosity 0.001 required switching to Matérn-2/3. For the nonlinear elliptic PDE and Allen-Cahn equation, Gaussian kernels remained effective but required different length-scales (*e.g.*, 0.1 for the elliptic PDE *vs.* 0.04 for Allen–Cahn with $a = 15$), consistent with their different effective frequency content.

Number of Factor Functions. We next evaluated the effect of the number of factor functions (rank). With length-scale parameters fixed as in Table 1, we varied the rank over $\{3, 5, 10, 20\}$. Experiments were conducted with 600 and 2400 collocation points for the Burgers' equation, and with 4800 and 22.5K points for the 2D Allen-Cahn equation. As reported in Tables 7a and 7b, rank 10 consistently yielded the best performance. Smaller ranks (3 or 5) reduced expressivity and resulted in errors one to two orders of magnitude larger. Increasing the rank to 20 offered no further gain and, in some cases (*e.g.*,TGPS-NT with 600 collocation points), worsened performance. A similar trend was observed for the Allen-Cahn equation, although its performance was somewhat more robust to rank variations.

Overall, these studies demonstrate that both kernel length-scales and rank are crucial hyperparameters. Too small a rank limits model expressivity, degrading accuracy, while excessively large ranks increase computational and optimization burdens without clear benefits.

K Limitation

While the partial freezing strategy and Newton's method allow us to derive closed-form ALS updates, they also make the update process effectively behave like a sequence of fixed-point iterations. A well-known limitation of such iterations is their sensitivity to initialization: if the starting point is poorly chosen, the iterations may diverge instead of converging. To mitigate this risk, in future work we plan to design additional regularization techniques that explicitly incorporate parameter estimates from earlier iterations into the update rules. This would provide a stabilizing effect, improving both the robustness and the reliability of our method.

Table 7: Relative L^2 error of TGPS with different numbers of factor functions (rank).

(a) Solving Burgers' equation (17) with viscosity $\nu = 0.02$. Inside parentheses are the number of collocation points.

Rank	3	5	10	20
TGPS-PF (600)	2.59E-01	1.47E-02	7.03E-03	5.54E-03
TGPS-NT (600)	3.03E-01	8.40E-01	8.42E-03	1.14E-02
TGPS-PF (2400)	1.64E-02	1.04E-03	1.85E-04	2.43E-04
TGPS-NT (2400)	1.97E-02	9.82E-04	2.37E-04	4.06E-04

(b) Solving 2D Allen-Cahn equation (20): $a = 15, d = 2$.

Rank	3	5	10	20
TGPS-PF (4800)	1.61E-05	1.23E-05	6.52E-06	1.01E-05
TGPS-NT(4800)	8.92E-06	6.50E-06	5.87E-06	9.66E-06
TGPS-PF (22500)	1.63E-05	7.10E-06	5.51E-06	9.22E-06
TGPS-NT (22500)	1.57E-05	5.27E-06	2.21E-06	7.34E-06

Table 8: Relative L^2 error of solving *higher dimensional* PDEs.

(a) 4D Allen-Cahn equation (20): $a = 15, d = 4$.

Method	8000	16000	32000	48000
PINN	8.01E-01	7.87E-01	7.68E-01	7.62E-01
SKS	9.85E-01	9.91E-01	9.93E-01	7.50E-01
TGPS-CP-PF	**4.00E-04**	1.10E-04	**2.76E-05**	1.65E-05
TGPS-CP-NT	**3.93E-04**	**9.80E-05**	**1.87E-05**	1.66E-05
TGPS-TR-PF	5.50E-01	**5.44E-05**	**2.59E-05**	1.09E-05
TGPS-TR-NT	5.65E-01	**7.48E-05**	**1.85E-05**	**7.47E-06**

(b) 6D Allen-Cahn equation (20): $a = 15, d = 6$.

Method	16000	32000	48000	96000
PINN	6.79E-01	6.10E-01	8.66E-01	6.12E-01
TGPS-CP-PF	6.62E-01	4.79E-04	3.50E-04	8.34E-05
TGPS-CP-NT	6.62E-01	4.98E-04	3.00E-04	7.75E-05
TGPS-TR-PF	**1.50E-02**	**4.23E-05**	**3.39E-05**	**1.11E-05**
TGPS-TR-NT	7.52E-01	**4.62E-05**	**4.48E-05**	**1.12E-05**

(c) 6D Nonlinear Darcy flow equation (22).

Method	1000	2000	4000	8000	16000
PINN	**1.04E-02**	**3.65E-02**	7.34E-02	8.87E-02	1.22E-02
DAKS	3.87E0	3.81E0	NA	NA	NA
TGPS-CP-PF	5.24E-01	3.02E-01	1.67E-01	2.96E-02	**6.59E-03**
TGPS-CP-NT	5.19E-01	3.70E-01	1.71E-01	2.42E-02	1.31E-02
TGPS-TR-PF	5.97E-01	**5.15E-02**	**2.67E-02**	**7.53E-03**	**5.48E-03**
TGPS-TR-NT	5.75E-01	**5.39E-02**	**1.48E-02**	**5.31E-03**	**4.02E-03**