TOWARD EFFICIENT KERNEL-BASED SOLVERS FOR NONLINEAR PDES

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper introduces a novel kernel learning framework toward efficiently solving nonlinear partial differential equations (PDEs). In contrast to the state-of-the-art kernel solver that embeds differential operators within kernels, posing challenges with a large number of collocation points, our approach eliminates these operators from the kernel. We model the solution using a standard kernel interpolation form and differentiate the interpolant to compute the derivatives. Our framework obviates the need for complex Gram matrix construction between solutions and their derivatives, allowing for a straightforward implementation and scalable computation. As an instance, we allocate the collocation points on a grid and adopt a product kernel, which yields a Kronecker product structure in the interpolation. This structure enables us to avoid computing the full Gram matrix, reducing costs and scaling efficiently to a large number of collocation points. We provide a proof of the convergence and rate analysis of our method under appropriate regularity assumptions. In numerical experiments, we demonstrate the advantages of our method in solving several benchmark PDEs.

1 Introduction

Solving partial differential equations (PDEs) stands as a central task in scientific and engineering domains. Recently, machine learning (ML)-based solvers have garnered significant attention. Unlike traditional numerical methods, ML-based solvers eliminate the need for complex mesh designs and intricate numerical techniques, enabling simpler, faster, and more convenient implementation and use. Among these solvers, kernel methods or Gaussian processes (GPs) (Williams and Rasmussen, 2006) hold great promise due to their solid mathematical foundations, offering high expressiveness, robustness, and the ability to quantify and reason under uncertainty. Recently, Chen et al. (2021a) introduced a general kernel method to approximate solutions of nonlinear PDEs. They augment the representation by incorporating differential operators (more generally, linear operators) into the kernels, and jointly fit the solution values and their derivatives of the PDE on a set of collocation points. Their approach has demonstrated promising performance in solving several benchmark nonlinear PDEs, backed by a rigorous error analysis, including both convergence and convergence rates (Chen et al., 2021a; Batlle et al., 2023).

Despite its success, the methodology requires manual construction of a Gram matrix between the solution and its derivatives (which show up in the PDEs). It enumerates combinations between the operators on the kernel to compute different sub-blocks. The process enlarges the size of the Gram matrix (as compared to the number of collocation points), and the computation becomes challenging with a large number of collocation points, a crucial factor for capturing complex PDE solutions such as with potential roughness (or even non-smoothness) and higher frequencies.

In response, this work proposes an alternative kernel-based framework for solving nonlinear PDEs with several key contributions:

- **Framework**: We remove the basis functions associated with the differential evaluation functionals in the approximation and use a standard kernel interpolation for solution modeling. To approximate solution derivatives, we directly differentiate the interpolant. By minimizing the RKHS norm along with a boundary and residual loss, we estimate the solution without the need for manually constructing a complex Gram matrix. Implementation is straightforward and convenient with the aid of modern automatic differential libraries.

- **Computational Method**: Our framework allows an immediate use of many existing efficient Gram matrix computation and/or approximation techniques. As an instance, we propose to place collocation points on a grid and employ a product kernel over each input dimension. This choice induces a Kronecker product structure within both the kernel interpolation and its differentiation, effectively bypassing the need for computing the entire Gram matrix. Such modification results in a substantial reduction in computational costs, enabling the efficient processing of tens of thousands or even millions of collocation points. This is achieved without resorting to any sparse and/or low-rank approximations.
- **Theorem:** We provide a rigorous analysis of our framework. We show the convergence and convergence rate of our method under appropriate PDE stability and regularity assumptions that are very similar to the assumptions used in the prior work (Batlle et al., 2023). However, our results are not a trivial extension of the prior work in that while our framework uses a reduced model space for efficient computation, our convergence results are as comparably strong as those in the prior work (Chen et al., 2021a; Batlle et al., 2023) that employs a richer model space. This is achieved through a much more sophisticated proof. We construct an interpolate of the true solution as an intermediate bridge connecting the true solution and our approximation, via which we prove our learning objective is not only feasible, the learned approximation also has a bounded RKHS norm. Next, via using domain decomposition, sampling inequality and mean inequality, we are able to bound the L_2 norm of the error w.r.t the PDE operators. Combined with the bounded RKHS norm of our approximation, we establish convergence and convergence rate. The results theoretically affirm the efficacy of our method in yielding accurate solutions.
- **Experiments**: Evaluation on Burgers', nonlinear elliptical, Eikonal, and Allen-Cahn equations have demonstrated our method's efficacy. For less challenging scenarios where a small number of collocation points, *e.g.*, 1000, is sufficient, our method achieves comparable or sometimes smaller errors than the existing methods. In more challenging scenarios, such as Burgers' equation with a viscosity of 0.001, our method seamlessly scales to tens of thousands of collocation points, yielding low errors on the order of 10^{-3} to 10^{-6}, underscoring its robustness and accuracy across a range of problem complexities.

2 Background

We consider a PDE of the general form,

$$\begin{aligned} \mathcal{P}(u) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ \mathcal{B}(u) &= g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_d)^\top$, and $\mathcal{P}$ and $\mathcal{B}$ are nonlinear differential operators in the interior Ω and boundary $\partial\Omega$, respectively. We assume $\mathcal{P}$ and $\mathcal{B}$ are composed by a series of linear operators, namely,

$$\begin{aligned} \mathcal{P}(u)(\mathbf{x}) &= P\left(L_1(u)(\mathbf{x}), \ldots, L_{Q_\Omega}(u)(\mathbf{x})\right), \ \mathbf{x} \in \Omega, \\ \mathcal{B}(u)(\mathbf{x}) &= B\left(L_{Q_\Omega+1}(u)(\mathbf{x}), \ldots, L_Q(u)(\mathbf{x})\right), \ \mathbf{x} \in \partial\Omega, \end{aligned}$$

where $P(\cdot)$ and $B(\cdot)$ are nonlinear functions, each L_j $(1 \le j \le Q)$ is a linear operator, such as derivatives of u and their linear combinations: $\partial_{x_1x_1}u$, $\partial_{x_1x_2}u$, $a \cdot \partial_{x_2x_2}u + b \cdot u$, *etc*.

To solve the PDE (1), Chen et al. (2021a) proposed to sample a set of collocation points, $\mathcal{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{M_\Omega} \in \Omega, \mathbf{x}_{M_\Omega+1}, \ldots, \mathbf{x}_M \in \partial\Omega\}$ in the domain, and estimated the solution values and all the relevant linear operators over the solution, $\{L_j(u)(\cdot)\}_j$, evaluated at the collocation points. Specifically, a nested optimization problem was formulated as

$$\begin{cases} \underset{\mathbf{z}}{\text{minimize}} \begin{cases} \underset{u \in \mathcal{U}}{\text{minimize}} \quad \|u\|_{\mathcal{U}} \\ \text{s.t. } \ L_j(u)(\mathbf{x}_m) = z_m^j, 1 \le j \le Q, \\ 1 \le m \le M_\Omega \ \text{ if } j \le Q_\Omega \\ \text{otherwise } 1 \le m \le M - M_\Omega \end{cases} \\ \text{s.t.} \quad P(z_m^1, \ldots, z_m^{Q_\Omega}) - f(\mathbf{x}_m) = 0, \ \ 1 \le m \le M_\Omega, \\ B(z_{m'}^{Q_\Omega+1}, \ldots, z_{m'}^{Q}) - g(\mathbf{x}_{m'}) = 0, \ \ 1 \le m' \le M - M_\Omega, \end{cases} \tag{2}$$

where $\mathbf{z}$ includes all $\{z_m^j\}$, $\mathcal{U}$ is the Reproducing Kernel Hilbert Space (RKHS) associated with a kernel $\kappa(\cdot,\cdot)$.With the RKHS $\mathcal{U}$, the similarity (or covariance) between any $z_{m_1}^{j_1}$ and $z_{m_2}^{j_2}$ in $\mathbf{z}$ is

$$c(z^{j_1}(\mathbf{x}_{m_1}), z^{j_2}(\mathbf{x}_{m_2})) = L_{j_1} \circ L_{j_2}(\kappa)(\mathbf{x}_{m_1}, \mathbf{x}_{m_2}),$$

where L_{j_1} is applied along the first argument of $\kappa(\cdot,\cdot)$ and L_{j_2} is applied along the second argument, and then we evaluate at $\mathbf{x}_{m_1}$ and $\mathbf{x}_{m_2}$. For example, suppose we have $L_1(u) = \partial_{x_1} u$ and $L_2(u) = \partial_{x_2} u$, then $L_1 \circ L_2(\kappa)(\mathbf{x}, \mathbf{x}') = \partial^2 \kappa(\mathbf{x}, \mathbf{x}')/\partial x_1 \partial x_2'$ where x_1 and x_2' are the first and second elements of the inputs $\mathbf{x}$ and $\mathbf{x}'$, respectively.

The true solution u^* is assumed to reside in $\mathcal{U}$, and $\|\cdot\|_{\mathcal{U}}$ is the RKHS norm associated with $\mathcal{U}$. From the representation theorem (Owhadi and Scovel, 2019), it is straightforward to show that the optimum of (2), denoted by v_M, has the following form

$$v_M(\mathbf{x}) = c(v_M(\mathbf{x}), \mathbf{z})\mathbf{C}(\mathbf{z}, \mathbf{z})^{-1}\mathbf{z}, \tag{3}$$

where $c(v_M(\mathbf{x}), \mathbf{z})$ is the similarity between $v_M(x)$ and each element in $\mathbf{z}$, namely, $\forall z_m^j \in \mathbf{z}$,

$$c(v_M(\mathbf{x}), z_m^j) = L_j(\kappa)(\mathbf{x}, \mathbf{x}_m),$$

where L_j is applied along the second argument of $\kappa(\cdot,\cdot)$, and $\mathbf{x}_m$ is the collocation point corresponding to z_m^j. Here $\mathbf{C}(\mathbf{z}, \mathbf{z})$ is the Gram matrix of $\mathbf{z}$, constructed from $Q \times Q$ sub-blocks,

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_{11} & \dots & \mathbf{C}_{1Q} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{Q1} & \dots & \mathbf{C}_{QQ} \end{pmatrix}, \tag{4}$$

where each $\mathbf{C}_{ij}$ is the similarity matrix associated with a pair of linear operators,

$$\mathbf{C}_{ij} = c(\mathbf{h}_i, \mathbf{h}_j) = L_i \circ L_j(\kappa)(\mathbf{h}_i, \mathbf{h}_j), \tag{5}$$

where $1 \le i, j \le Q$, L_i and L_j are applied to the first and second arguments of κ, respectively, $\mathbf{h}_i = \{z_m^i\}_m$ and $\mathbf{h}_j = \{z_m^j\}_m$. Therefore, the size of the Gram matrix $\mathbf{C}$ is $(Q_\Omega M_\Omega + (Q - Q_\Omega)(M - M_\Omega)) \times (Q_\Omega M_\Omega + (Q - Q_\Omega)(M - M_\Omega))$.

The method (Chen et al., 2021a) can be explained from a probabilistic perspective. That is, we assign a GP prior over u, and given a sufficiently smooth kernel κ, all the linear operators over u, namely, $L_j(u)$ also follow a GP prior, and their projection on the collocation points $\mathcal{M}$, namely, $\mathbf{z}$, follow a multi-variate Gaussian prior distribution, $p(\mathbf{z}) \sim \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{C})$. Softening the outer constraints in the aforementioned nested optimization by maximizing a likelihood, this method essentially seeks for an MAP estimation of $\mathbf{z}$, and the prediction (3) is the posterior mean conditioned on $\mathbf{z}$.

3 Our Framework

Despite the success of (Chen et al., 2021a), it requires computation of a Gram (covariance) matrix (see (4)) with dimensions typically exceeding the number of collocation points. This can exacerbate computational challenges, particularly when addressing complex PDEs that demand a considerable number of collocation points (Cho et al., 2024; Florido et al., 2024). Moreover, the construction of the Gram matrix relies on the particular set of linear operators present in the PDE, rendering it cumbersome for implementation and the adoption of efficient approximations, if needed.

We therefore propose an alternative kernel learning framework for nonlinear PDE solving, which simplifies the Gram matrix construction and computation. Specifically, we are inspired by the standard kernel/GP regression. Suppose the solution values at the collocation points are known, denoted as $\mathbf{u}_{\mathcal{M}}^* = (u^*(\mathbf{x}_1), \dots, u^*(\mathbf{x}_M))^\top$. The optimal solution estimate within the framework of standard kernel regression takes the interpolation form: $t(\mathbf{x}) = \kappa(\mathbf{x}, \mathcal{M})\mathbf{K}_{MM}^{-1}\mathbf{u}_{\mathcal{M}}^*$, where $\mathbf{K}_{MM} = \kappa(\mathcal{M}, \mathcal{M})$ denotes the kernel matrix computed on the collocation points (of size $M \times M$). This form is derived by minimizing the RKHS norm while aligning u^* at $\mathcal{M}$. In GP regression, $t(\mathbf{x})$ serves as the mean function of the posterior process.

In the broader context of PDE solving, as depicted in (1), one often lacks knowledge of the solution values at arbitrary collocation points. Therefore, we regard them as unknown, free variables denoted by $\boldsymbol{\eta}$. We model the solution estimate as

$$u(\mathbf{x}; \boldsymbol{\eta}) = \kappa(\mathbf{x}, \mathcal{M})\mathbf{K}_{MM}^{-1}\boldsymbol{\eta}. \tag{6}$$

We then apply each linear operator L_j in the PDE over $u(\mathbf{x};\boldsymbol{\eta})$ to approximate $L_j(u^*)$. Following this way, we do not need to explicitly estimate the values of $L_j(u^*)$ at the collocation points (namely z_m^j in (2)). Correspondingly, the Gram matrix $\mathbf{K}_{MM}$ is substantially smaller (of size $M \times M$) and it is more convenient to compute — there is no need to enumerate pairs of linear operators and apply them to the kernel function to compute different sub-blocks.

The learning is carried out by addressing the following constrained optimization problem:

$$\begin{cases} \underset{u\in\mathcal{U}}{\text{minimize}} \ \ \|u\|_{\mathcal{U}} \\ \text{s.t. } \frac{1}{M_\Omega}\sum_{m=1}^{M_\Omega}\left(\mathcal{P}(u)(\mathbf{x}_m) - f(\mathbf{x}_m)\right)^2 \\ +\frac{1}{M-M_\Omega}\sum_{m=M_+1}^{M}\left(\mathcal{B}(u)(\mathbf{x}_m) - g(\mathbf{x}_m)\right)^2 \le \epsilon, \\ u \text{ takes the kernel interpolation form (6).} \end{cases} \tag{7}$$

where $\epsilon \ge 0$ is a given relaxation parameter. Note that since our formulation (6) uses a reduced model space as opposed to (Chen et al., 2021a), we introduce ϵ to enable feasibility of the optimization and to establish the convergence; see our convergence analysis in Appendix Section A. In practical scenarios, addressing (7) directly can be cumbersome. We may opt to optimize an unconstrained objective with soft regularization instead,

$$\begin{aligned} \underset{\boldsymbol{\eta}}{\text{minimize}} \ \ \mathcal{L}(u(x;\boldsymbol{\eta});\alpha,\beta) := \|u\|_{\mathcal{U}}^2 + \alpha\cdot\left[\frac{1}{M_\Omega}\sum\nolimits_{m=1}^{M_\Omega}(\mathcal{P}(u)(\mathbf{x}_m) - f(\mathbf{x}_m))^2 - \epsilon/2\right] \\ + \beta\cdot\left[\frac{1}{M-M_\Omega}\sum\nolimits_{m=M_\Omega+1}^{M}(\mathcal{B}(u)(\mathbf{x}_m) - g(\mathbf{x}_m))^2 - \epsilon/2\right], \end{aligned} \tag{8}$$

where $\alpha, \beta > 0$ are the regularization strengths, and ϵ can be simply set to zero.

Efficient Computation. In scenarios where PDEs are complex and challenging, capturing the solution details might necessitate employing a vast array of collocation points. Since our framework uses the standard kernel matrix to construct the solution estimate (as illustrated in (6)), a wide range of existing kernel approximation and computation methods (Quinonero-Candela and Rasmussen, 2005; Rahimi and Recht, 2007; Farahat et al., 2011; Lindgren et al., 2011) can be readily employed to accelerate computation involving $\mathbf{K}_{MM}^{-1}$. This facilitates the reduction of computational costs and enables scalability to accommodate massive collocation points.

As an instance, we propose to induce a Kronecker product structure to accelerate computation and scale to a large number of collocation points. Specifically, we place the collocation points on a grid, namely, $\mathcal{M} = \mathbf{s}^1 \times \ldots \times \mathbf{s}^d$, where each $\mathbf{s}^k$ includes a collection of locations at input dimension k, *i.e.,* $\mathbf{s}^k = (s_1^k, \ldots, s_{m_k}^k)^\top \in \mathbb{R}^{m_k}$. These locations can be regular-spaced or randomly sampled. Accordingly, $\mathcal{M}$ is an d-dimensional array of size $m_1 \times \ldots \times m_d$. Next, we employ a product kernel that is decomposed as along the input dimensions, $\kappa(\mathbf{x},\mathbf{x}') = \prod_{j=1}^d \kappa_j(x_j, x_j')$, where each κ_j is a kernel function of two scalar variables at input dimension j. As a result, the kernel matrix on the collocation points $\mathcal{M}$ becomes a Kronecker product,

$$\mathbf{K}_{MM} = \mathbf{K}_1 \otimes \ldots \otimes \mathbf{K}_d,$$

where each $\mathbf{K}_j = \kappa_j(\mathbf{s}^j, \mathbf{s}^j)$ is a local kernel matrix for dimension j ($1 \le j \le d$), of size $m_j \times m_j$. We then leverage the Kronecker product properties to efficiently compute the solution estimate (6),

$$\begin{aligned} u(\mathbf{x};\boldsymbol{\eta}) &= \left[\kappa_1(x_1,\mathbf{s}^1)\otimes\ldots\otimes\kappa_d(\mathbf{x}_d,\mathbf{s}^d)\right]\cdot\left[\mathbf{K}_1\otimes\ldots\otimes\mathbf{K}_d\right]^{-1}\boldsymbol{\eta} \\ &= \left[\kappa_1(x_1,\mathbf{s}^1)\mathbf{K}_1^{-1}\otimes\ldots\otimes\kappa_d(x_d,\mathbf{s}^d)\mathbf{K}_d^{-1}\right]\boldsymbol{\eta} \\ &= \mathcal{A}\times_1\left[\kappa_1(x_1,\mathbf{s}^1)\mathbf{K}_1^{-1}\right]\times_2\ldots\times_d\left[\kappa_d(x_d,\mathbf{s}^d)\mathbf{K}_d^{-1}\right], \end{aligned} \tag{9}$$

where $\mathcal{A}$ is the tensor view of $\boldsymbol{\eta}$, namely reshaping $\boldsymbol{\eta}$ as a $m_1 \times \ldots \times m_d$ array, and $\times_k$ is the mode-k tensor-matrix multiplication (Kolda, 2006). In this way, we avoid explicitly computing the full kernel matrix $\mathbf{K}_{MM}$ and its inverse. We only need to invert each local kernel matrix $\mathbf{K}_j$, and hence the cost is substantially reduced. For example, considering a $100 \times 100 \times 100$ grid, the full kernel matrix is $10^6 \times 10^6$, rending it computationally prohibitive and impracticable for most hardware. By using (9), we only need to invert three 100×100 local kernel matrices, which is cheap and fast. Note that, since the kernel is decomposed across individual dimensions, taking derivatives over the solution estimate u will maintain the structure, *e.g.*, $\partial_{x_1x_d}u(\mathbf{x};\boldsymbol{\eta}) = \mathcal{A}\times_1\left[\partial_{x_1}\kappa_1(x_1,\mathbf{s}^1)\mathbf{K}_1^{-1}\right]\times_2\ldots\times_d\left[\partial_{x_d}\kappa_d(x_d,\mathbf{s}^d)\mathbf{K}_d^{-1}\right]$.

We then leverage the structure (9) to efficiently minimize the objective (7) or (8). The computation of each $\mathcal{P}(u)(\mathbf{x}_m)$ and $\mathcal{B}(u)(\mathbf{x}_m)$ is a straightforward application of the operators $\mathcal{P}$ and $\mathcal{B}$ to (9) and then evaluate them at the collocation points. This can be done by automatic differential libraries, such as JAX (Frostig et al., 2018). The RKHS norm in (7) and (8) can be efficiently computed by

$$\|\mathbf{u}\|_{\mathcal{U}}^2 = \boldsymbol{\eta}^\top \mathbf{K}_{MM}^{-1} \boldsymbol{\eta} = \boldsymbol{\eta}^\top \left[\mathbf{K}_1 \otimes \ldots \otimes \mathbf{K}_d\right]^{-1} \boldsymbol{\eta} = \boldsymbol{\eta}^\top \mathrm{vec}(\mathcal{A} \times_1 \mathbf{K}_1^{-1} \times_2 \ldots \times_d \mathbf{K}_d^{-1}). \quad (10)$$

We can apply any gradient-based optimization algorithm..

4 Convergence Analysis

We now show the convergence of our method. We inherit the road-map of (Batlle et al., 2023) and maintain the same assumption about PDE stability and the regularity of the domain and boundary (Batlle et al., 2023, Assumption 3.7)[1], with a slight modification.

Assumption 4.1. 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)* $\exists \gamma > 0$ and $\exists k, t \in \mathbb{N}$ 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 $\forall u_1, u_2 \in B_r(H^l(\Omega))$,

$$\|u_1 - u_2\|_{H^l(\Omega)} \leq C \left(\|\mathcal{P}(u_1) - \mathcal{P}(u_2)\|_{H^0(\Omega)} + \|\mathcal{B}(u_1) - \mathcal{B}(u_2)\|_{H^0(\partial\Omega)} \right), \quad (11)$$

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

$$\|\mathcal{P}(u_1) - \mathcal{P}(u_2)\|_{H^k(\Omega)} + \|\mathcal{B}(u_1) - \mathcal{B}(u_2)\|_{H^t(\partial\Omega)} \leq C \|u_1 - u_2\|_{H^s(\Omega)}, \quad (12)$$

  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 where each element and its weak derivatives up to the order of j have a finite L^2 norm.
- *(C3)* The RKHS $\mathcal{U}$ is continuously embedded in $H^{s+\tau}(\Omega)$ where $\tau > 0$.

Lemma 4.2. *Let u^* denote the unique strong solution of* (1)*. Suppose Assumption 4.1 is satisfied, and a set of collocation points $\mathcal{M} \subset \overline{\Omega}$ is given, where $\mathcal{M}_\Omega \subset \mathcal{M}$ denotes the collocation points in the interior of Ω and $\mathcal{M}_{\partial\Omega} \subset \mathcal{M}$ the collocation points on the boundary $\partial\Omega$. Define the fill-distances*

$$h_\Omega := \sup_{\mathbf{x} \in \Omega} \inf_{\mathbf{x}' \in \mathcal{M}_\Omega} |\mathbf{x} - \mathbf{x}'|, \quad h_{\partial\Omega} := \sup_{\mathbf{x} \in \partial\Omega} \inf_{\mathbf{x}' \in \mathcal{M}_{\partial\Omega}} \rho_{\partial\Omega}(\mathbf{x}, \mathbf{x}'), \quad (13)$$

where $|\cdot|$ is the Euclidean distance, and $\rho_{\partial\Omega}$ is a geodesic distance defined on $\partial\Omega$. Set $h = max(h_\Omega, h_{\partial\Omega})$. There is always a minimizer of (7) *with the set of collocation points $\mathcal{M}$ and $\epsilon = C_0 h^{2\tau}$ where $C_0 > 0$ is a sufficiently large constant independent of h. Let $u^\dagger$ denote such a minimizer. When h is sufficiently small, at least $h \leq C_1 M^{-\frac{1}{d}}$ where $C_1 > 0$ is a constant, then*

$$\|u^\dagger - u^*\|_{H^l(\Omega)} \leq C h^\rho \|u^*\|_{\mathcal{U}}, \quad (14)$$

where $\rho = \min(k, t, \tau)$, and $C > 0$ is independent of $u^\dagger$ and h.

Proposition 4.3. *Given the set of collocation points $\mathcal{M}$ and $\epsilon = C_0 h^{2\tau}$ where $C_0 > 0$ is a sufficiently large constant, there exists $\alpha_M, \beta_M > 0$ such that the minimizer of* (8) *with $\alpha = \alpha_M$ and $\beta = \beta_M$ is also the minimizer of* (7)*. That means, with proper choices of the regularization strengths, the minimizer of* (8) *enjoys the same convergence result as in* (14)*.*

We can see the convergence results of our framework are as comparably strong as the results for the method of Chen et al. (2021b); see (Batlle et al., 2023, Theorem 3.8), though the latter employs a richer model space. We leave the proof in Section A and B of the Appendix.

[1]Note that this assumption, along with its minor variants, is considered mild and widely used in convergence analysis. For many examples of nonlinear PDEs that satisfy this assumption, see (Batlle et al., 2023).

5 Related Work

The prior works of Graepel (2003); Raissi et al. (2017) propose Gaussian Processes (GPs) models for solving linear PDEs in the presence of noisy measurements of source terms. Recently, Wang et al. (2021) delve into the rationale and guarantees of using GPs as a prior for PDE solutions. The effectiveness of the product kernel is also justified in terms of sample path properties.

Chen et al. (2021a) introduced a kernel method capable of solving both linear and nonlinear PDEs. The solution approximation is constructed by both kernels and kernel derivatives (more generally, the linear operators of the PDE over the kernels). Hence, the differentiation operators need to be embedded into the kernels to construct the Gram matrix whose dimension is typically much greater than the number of collocation points. In (Batlle et al., 2023), a systematic theoretical framework is established to analyze the convergence and the convergence rate of the method of Chen et al. (2021a). To alleviate the computational challenge for massive collocation points, Chen et al. (2023) adapted the sparse inverse Cholesky factorization (Schafer et al., 2021) to approximate the Gram matrix of (Chen et al., 2021a). An alternative approach was proposed by Meng and Yang (2023), which adjusted the Nyström method (Jin et al., 2013) to obtain a sparse approximation of the Gram matrix. Despite the success of these methods, the construction of the sparse approximation needs to carefully handle different sub-blocks in the Gram matrix, where each sub-block corresponds a pair of linear operators over the kernels, and hence it is complex and relatively inconvenient for implementation. In our work, the solution is approximated by a standard kernel interpolation, and the Gram matrix is therefore just the kernel matrix over the collocation points. The size of the Gram matrix is smaller. More important, the existent sparse approximation methods can be readily applied to our model, without the need for complex adjustments or novel development.

The computational efficiency of Kronecker product structures has been recognized in various works (Saatcci, 2012; Xu et al., 2012; Wilson and Nickisch, 2015; Izmailov et al., 2018; Zhe et al., 2019). Wilson et al. (2015) highlighted that utilizing a regular (evenly-spaced) grid results in Toeplitz-structured kernel matrices, facilitating $O(n \log n)$ computation. However, in typical machine learning applications, data is not observed on a grid, limiting the utility of the Kronecker product. In contrast, for PDE solving, estimating solution values on a grid is natural, making Kronecker products combined with kernels a promising avenue for efficient computation. The recent work (Fang et al., 2023) uses a similar computational method to solve high-frequency and multi-scale PDEs. The major contribution is to introduce a spectral mixture kernel in each dimension to capture the dominant frequencies in the kernel space. This work can be viewed as an instance of our proposed framework. We in addition give a theoretical analysis about the convergence of our framework. For more extensive discussions on Bayesian learning and PDE problems, readers are referred to (Owhadi, 2015).

Our work is also connected to the radial basis function (RBF) method for solving PDEs (Hardy, 1971; Kansa, 1990; Tolstykh and Shirobokov, 2003; Shu et al., 2003; Fornberg et al., 2011; Safdari-Vaighani et al., 2015). The RBF method typically approximates the PDE solution as a linear combination of RBF bases, *i.e.,* kernels, $u(\mathbf{x}) \approx \sum_j \alpha_j \kappa(\mathbf{x} - \mathbf{x}_j)$ where $\{\mathbf{x}_j\}$ are collocation points. To estimate the coefficients $\{\alpha_j\}$, the RBF method typically converts the PDE into a linear system $\mathbf{A}\boldsymbol{\alpha} = \mathbf{b}$ and accordingly solves $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots)^\top$. Our method differs mainly in two folds. First, we use kernel regression form (6) to approximate the solution, and do not explicitly estimate the coefficients $\boldsymbol{\alpha}$. In other words, we have implicitly $\boldsymbol{\alpha} = \mathbf{K}_{MM}^{-1}\boldsymbol{\eta}$. We instead estimate $\boldsymbol{\eta}$ — the solution values at the collocation points. Second, we do not convert the PDE solving into solving a linear system; instead, we convert it into solving an optimization problem (7) or (8). While we can also use the RBF formulation to approximate the solution in our framework, we found that, optimizing $\boldsymbol{\alpha}$ (instead of $\boldsymbol{\eta}$) will severely degrade the performance, which might be due to the complexity of the optimization procedure.

6 Numerical Experiments

To evaluate our method, we considered four commonly-used benchmark PDE families in the literature of machine learning based solvers (Raissi et al., 2019; Chen et al., 2021a).

The Burgers' Equation. We first tested with a viscous Burgers' equation,

$$u_t + uu_x - \nu u_{xx} = 0, \ \ \forall (x,t) \in (-1,1) \times (0,1], \\ u(x,0) = -\sin(\pi x), \ \ u(-1,t) = u(1,t) = 0. \tag{15}$$

The solution is computed from the Cole–Hopf transformation with numerical quadrature (Chen et al., 2021a). We considered two cases: $\nu = 0.02$, and $\nu = 0.001$.

Nonlinear elliptic PDE. We next tested with the instance of nonlinear elliptic PDE used in (Chen et al., 2021a),

$$-\Delta u(\mathbf{x}) + u^3(\mathbf{x}) = f(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = 0, \quad \forall \mathbf{x} \in \partial\Omega, \tag{16}$$

where $\Omega = [0,1]^2$, the solution is crafted as $u(\mathbf{x}) = \sin(\pi x_1)\sin(\pi x_2) + 4\sin(4\pi x_1)\sin(4\pi x_2)$, and $f(\mathbf{x})$ is correspondingly computed via the equation.

Eikonal PDE. Third, we tested with a regularized Eikonal equation as used in (Chen et al., 2021a),

$$|\nabla u(\mathbf{x})|^2 = f(\mathbf{x})^2 + \epsilon \Delta u(\mathbf{x}), \ \ \forall \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = 0, \ \ \forall \mathbf{x} \in \partial\Omega, \tag{17}$$

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

Allen-Cahn Equation. Fourth, we considered a 2D stationary Allen-Cahn equation with a source function and Dirichlet boundary conditions.

$$u_{xx} + u_{yy} + \gamma(u^m - u) = f(x,y), \ \ (x,y) \in [0,1] \times [0,1], \tag{18}$$

where $\gamma = 1$ and $m = 3$. We crafted the solution in the form $u = \sin(2\pi a x_1)\cos(2\pi a x_2) + \sin(2\pi x_1)\cos(2\pi x_2)$, and f is computed through the equation. We tested with $a = 15$ and $a = 20$.

Method and Settings. We implemented our method with JAX (Frostig et al., 2018). We denote our method as SKS (Simple Kernel-based Solver) We compared with (Chen et al., 2021a) that uses kernel and kernel derivatives (more generally, linear operators) to approximate the solution, which we denote as DAKS (Derivative-Augmented Kernel-based Solver). We used the implementation from the original authors[2]. In addition, we compared with physics-informed neural network (PINN) (Raissi et al., 2019), a mainstream machine learning PDE solver. The PINN is implemented with PyTorch (Paszke et al., 2019). For SKS, we minimize (8) (with $\epsilon = 0$), and used ADAM optimization with learning rate 10^{-3}. The maximum number of epochs was set to 1M. For DAKS, we used the relaxed Gauss-Newton optimization propsoed in the original paper across all the experiments. The PINN was first trained by 10K ADAM epochs with learning rate 10^{-3} and then by L-BFGS with learning rate 10^{-1} with a maximum of 50K iterations. The tolerance level for L-BFGS was set to 10^{-9}. To identify the architecture for the PINN, we varied the number of layers from {2, 3, 5, 8, 10}, and the the width of each layer from $\{10, 20, 30, \ldots, 100\}$. We used *tanh* as the activation function. For DAKS and SKS, we used Square Exponential (SE) kernel with different length-scales across the input dimensions. We selected the nugget term from {5E-5, 1E-5, 5E-6, 1E-6, ..., 1E-13}. However, for solving the nonlinear elliptic PDE with DAKS, we used its default approach that assigns an adaptive nugget for the two sub-blocks in the Gram matrix. This gives the best performance for DAKS. The length-scales were selected from a grid search, from $[0.1, 0.2]^2$ for the nonlinear elliptic and Eikonal PDEs, $[0.05, 0.01]^2$ for Allen-Cahn equation, and $[0.003, 0.05] \times [0.02, 0.3]$ for Burgers' equation. We reported the best solution error of each method throughout the running. In Section C.4 of Appendix, we further examined the sensitivity of our method to the kernel parameters.

6.1 Solution Accuracy

Simpler Cases. We first tested all the methods on less challenging benchmarks, for which a small number of collocation points is sufficient. Specifically, we tested with Burgers' equation with viscosity $\nu = 0.02$, the nonlinear elliptic PDE, and Eikonal PDE. These are the same test cases employed in (Chen et al., 2021a). Following (Chen et al., 2021a), we varied the number of collocation points

[2] https://github.com/yifanc96/NonLinPDEs-GPsolver

Method	600 (25×25)	1200 (35×35)	2400 (49×49)	4800 (70×70)
DAKS	1.75E-02	7.90E-03	8.65E-04	**9.76E-05**
PINN	**2.68E-03**	**6.72E-04**	**3.60E-04**	3.73E-04
SKS	1.44E-02	5.40E-03	7.83E-04	3.21E-04

(a) The Burgers' equation (15) with viscosity $\nu = 0.02$.

Method	300 (18×18)	600 (25×25)	1200 (35×35)	2400 (49×49)
DAKS	1.15E-01	1.15E-04	8.65E-04	**1.68E-07**
PINN	3.39E-01	1.93E-02	1.28E-03	3.20E-04
SKS	**1.26E-02**	**6.93E-05**	**6.80E-06**	1.83E-06

(b) Nonlinear elliptic PDE (16)

Method	300 (18×18)	600 (25×25)	1200 (35×35)	2400 (49×49)
DAKS	1.01E-01	1.64E-02	2.27E-04	7.78E-05
PINN	2.95E-02	1.26E-02	4.53E-03	3.50E-03
SKS	**6.23E-04**	**2.68E-04**	**1.91E-04**	**2.51E-05**

(c) Eikonal PDE (17).

Table 1: L^2 error of solving less challenging 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 used by DAKS. Note that the Gram matrix of DAKS is larger than SKS.

from {600, 1200, 2400, 4800} for Burgers' equation, and {300, 600, 1200, 2400} for nonlinear elliptic PDE and Eikonal PDE. In (Chen et al., 2021a), the collocation points are randomly sampled, and hence we used the average L^2 of DAKS from ten runs on different sets of randomly sampled collocation points for comparison. For SKS, we used a regularly-spaced, square grid, for which the total number of grid points is close to that used for DAKS. Note that the size of the Gram matrix of DAKS is larger than SKS. We ran PINN on the same set of grid points used by SKS. The L^2 error is reported in Table 1. It can be seen that in most cases, SKS achieves smaller solution error than DAKS, with the exception for Burgers[3] and nonlinear elliptic PDE using 4800 and 2400 collocation points, respectively. This might be because DAKS needs to explicitly estimate more variables, including all kinds of linear operators (*e.g.*, derivatives) over the solution at the collocation points, which increases the optimization workload. In addition, the influence of the nugget term might vary across different heterogeneous blocks in the Gram matrix, which complicates the optimization. In most cases, the PINN shows worse performance than SKS, except on Burgers' equation with 600, 1200 and 2400 collocation points. The results have shown that in regimes where the computation of the Gram matrix is not a bottleneck, our method SKS, though adopting a simpler model design, can still achieve comparable or even better solution accuracy.

Difficult Cases. Next, we tested with more challenging cases, for which massive collocation points are necessary. These cases include Burgers' equation with $\nu = 0.001$, and Allen-Cahn equation with $a = 15$ and $a = 20$. For Burgers' equation, we found empirically that the spatial resolution is more important than the time resolution, so we set the ratio between the spatial and time resolutions to 3:1. For Allen-Cahn, we still used a square-shaped grid. We provide a more detailed ablation study in Appendix Section C.2. To verify the necessity of using massive collocation points, we first ran all the methods with the same number of collocation points as adopted in the simpler PDEs, namely, a few hundreds and/or thousands. As we can see from Table 2, the solution errors of all the methods are large, typically around the level of 10^{-1}, indicating failures. Note that, however, when the number of collocation points increases to 4800, SKS can achieve an L^2 error at level 10^{-4} for solving the 2D Allen-Cahn equation, while the other methods still struggle at 10^{-1} error level or even bigger. Together this indicates that a much larger number of collocation points is needed.

We then ran SKS and PINN with greatly increased collocation points, *i.e.,* dense grids, varying from 6400 to 120K. In such scenarios, running DAKS becomes extremely costly or even infeasible[4]. We therefore only report the results of SKS and PINN, as shown in Table 3. One can see that the solution

[3]However, if we use a 96×50 grid (still including 4800 collocation points), SKS gives L^2 error 7.54E-05, which can surpass DAKS. See the ablation study in Appendix Section C.2 for more details.

[4]For 6400 collocation points, the size of the Gram matrix of DAKS is 19200×19200 for Burgers' and 2D Allen-Cahn, because there are three linear operators in each equation.

Method	600 (42×14)	1200 (60×20)	2400 (84×28)	4800 (120×40)
DAKS	3.87E-01	3.12E-01	3.60E-01	2.37E-01
PINN	**1.27E-01**	2.59E-01	3.18E-01	2.65E-01
SKS	1.34E-01	**1.11E-01**	**8.04E-02**	**1.89E-02**

(a) The Burgers' equation (15) with viscosity $\nu = 0.001$.

Method	600 (25×25)	1200 (35×35)	2400 (49×49)	4800 (70×70)
DAKS	6.84E-01	6.62E-01	6.28E-01	5.74E-01
PINN	4.02E0	6.20E0	4.26E0	5.39E0
SKS	**6.80E-01**	**2.1E-01**	**5.15E-03**	**9.20E-05**

(b) The 2D Allen-Cahn equation (18) with $a = 15$

Method	600 (25×25)	1200 (35×35)	2400 (49×49)	4800 (70×70)
DAKS	6.81E-01	6.57E-01	6.19E-01	5.64E-01
PINN	4.98E0	5.78E0	5.87E0	3.04E0
SKS	7.07E-01	6.91E-01	**1.81E-01**	**9.83E-04**

(c) The 2D Allen-Cahn equation (18) with $a = 20$

Table 2: L^2 error of solving more challenging PDEs with a small number of collocation points.

Method	43200 (360×120)	67500 (450×150)	97200 (540×180)	120000 (600×200)
PINN	4.05E-03	6.01E-03	3.94E-03	4.13E-03
SKS	**3.90E-03**	**3.50E-03**	**2.60E-03**	**2.28E-03**

(a) The Burgers' equation (15) with viscosity $\nu = 0.001$.

Method	6400 (80×80)	8100 (90×90)	22500 (150×150)	40000 (200×200)
PINN	5.03E0	5.30E0	4.21E0	5.86E0
SKS	**8.27E-05**	**3.41E-05**	**4.34E-06**	**4.44E-06**

(b) The 2D Allen-Cahn equation (18) with $a = 15.0$

Method	6400 (80×80)	8100 (90×90)	22500 (150×150)	40000 (200×200)
PINN	4.18E0	4.45E0	5.86E0	5.93E0
SKS	**3.98E-04**	**1.82E-04**	**4.00E-05**	**2.98E-05**

(c) The 2D Allen-Cahn equation (18) with $a = 20.0$

Table 3: L^2 error of solving more challenging PDEs with a large number of collocation points.

error of SKS is substantially reduced, achieving 10^{-3} for Burgers' and 10^{-4} to 10^{-6} for Allen-Cahn. It is worth noting that PINN using the same set of collocation points also arrives at the 10^{-3} level L^2 error for Burgers' but the error on Allen-Cahn is still very large, with nearly no improvement upon using much fewer collocation points. This might be due to that the relatively high frequencies in the solution (see (18)) are difficult to be captured by neural networks, due to their known "spectral bias" (Rahaman et al., 2019). Thanks to our model design (3), we can induce a Kronecker product structure in the Gram matrix to scale to massive collocation points, without the need for designing complex approximations.

Point-wise Error. For a fine-grained comparison, we showcase the point-wise error of each method in solving Burgers' ($\nu = 0.001$) and 2D Allen-Cahn equations. The results and discussion are given by Appendix Section C.1.

Ablation Study on Grid Shape. We further examined the influence of the grid shape on the solution accuracy. We compared different choices on Burgers' equation with $\nu = 0.02$ and $\nu = 0.001$. We leave the details in Appendix Section C.2.

Comparison with Conventional Numerical Methods. In addition to comparing SKS with ML-based solvers, we also compared it to a finite difference solver — a widely used conventional numerical approach. We discretized the PDE using numerical differences, specifically employing a centered second-order numerical difference to approximate the derivatives. The equation was then solved using a Newton-Krylov solver, which computes the inverse of the Jacobian through an iterative Krylov method. We tested on solving the nonlinear elliptic PDE (16) and the Allen-cahn equation (18), since the ground-truth solutions of these PDEs are known and we can conduct a fair comparison. The

Method	18×18	25×25	35×35	49×49
Finite Difference	3.36E-02	1.78E-02	9.25E-03	4.78E-03
SKS	**1.26E-02**	**6.93E-05**	**6.80E-06**	**1.83E-06**

(a) Nonlinear Elliptic PDE (16).

Method	80×80	90×90	150×150	200×200
Finite difference	8.57E-02	6.68E-02	2.33E-02	1.30E-02
SKS	**8.27E-05**	**3.41E-05**	**4.34E-06**	**4.44E-06**

(b) The 2D Allen-Cahn equation (18) with $a = 15$.

Method	80×80	90×90	150×150	200×200
Finite difference	1.62E-01	1.24E-01	4.22E-02	2.34E-02
SKS	**3.98E-04**	**1.82E-04**	**4.00E-05**	**2.98E-05**

(c) The 2D Allen-Cahn equation (18) with $a = 20$.

Table 4: L^2 Error of a finite difference solver and SKS according to the ground-truth solution.

results are presented in Table 4. Note that the L^2 errors of PINN and SKS have already been reported in Table 1b, 3b and 3c. Our method (SKS) consistently outperforms finite difference. In most cases, the error of SKS is several orders of magnitudes smaller. It implies that using the same grid, SKS is much more efficient in approximating the solution. In addition, with the growth of the grid size, the relative improvement of our method is often more significant, in particular when solving the nonlinear elliptic PDE with the grid 18×18 increasing to 25×25, and Allen-Cahn ($a = 15$) with the grid 90×90 increasing to 150×150. Since our method is efficient in handling a large number of collocation points (*i.e.,* dense grid), it shows the potential of our method.

Irregular-shaped Domains. While our efficient computation is performed on grids, our method can be readily applied to irregular-shaped domains by introducing a virtual grid that encompasses such domains. This allows SKS to be used without any modifications. To validate the effectiveness of this strategy, we conducted additional tests, solving the nonlinear elliptic PDE (16) on a circular domain and the Allen-Cahn equation (18) on a triangular domain. In both cases, our method achieved reasonably good accuracy. Detailed results and discussions are provided in Appendix Section C.3.

Running Time. Finally, we examined the wall-clock runtime of SKS. We analyzed the runtime of each method when solving Burgers' equation $\nu = 0.001$ and the Allen-Cahn equation ($a = 15$) with varying numbers of collocation points. We ran the experiment on a Linux workstation equipped with an Intel(R) Xeon(R) Platinum 8360H Processor with 24GB memory. The results are presented in Appendix Table 9. SKS is several orders of magnitude faster than both DAKS and PINN per iteration. However, since DAKS employs the Gauss-Newton method, it converges much faster than the ADAM optimizer used by SKS and PINN. Nonetheless, the overall runtime of SKS is still less than 25% of that of DAKS when solving Burger's equation, and is close to DAKS when solving Allen-Cahn equation. Additionally, SKS can handle a much larger number of collocation points than DAKS. Overall, the runtime of SKS is significantly less than that of PINN.

7 Conclusion

We have proposed a new kernel method for nonlinear PDE solving. We use a standard kernel interpolation to model the solution estimate, which allows more convenient implementation and efficient computation. Our method can easily scale to massive collocation points, which are necessary for challenging PDEs. The performance on a series of benchmarks is encouraging. In the future, we plan to develop more efficient optimization, *e.g.*, Gaussian-Newton, to further accelerate our method.

References

Adams, R. A. and Fournier, J. J. (2003). Sobolev spaces. Elsevier.

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.

Chen, Y., Hosseini, B., Owhadi, H., and Stuart, A. M. (2021a). 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. (2021b). Physics-informed learning of governing equations from scarce data. Nature communications, 12(1):1–13.

Cho, J., Nam, S., Yang, H., Yun, S.-B., Hong, Y., and Park, E. (2024). Separable physics-informed neural networks. Advances in Neural Information Processing Systems, 36.

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.

Farahat, A., Ghodsi, A., and Kamel, M. (2011). A novel greedy algorithm for nyström approximation. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pages 269–277. JMLR Workshop and Conference Proceedings.

Florido, J., Wang, H., Khan, A., and Jimack, P. K. (2024). Investigating guiding information for adaptive collocation point sampling in pinns. In International Conference on Computational Science, pages 323–337. Springer.

Fornberg, B., Larsson, E., and Flyer, N. (2011). Stable computations with gaussian radial basis functions. SIAM Journal on Scientific Computing, 33(2):869–892.

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.

Hardy, R. L. (1971). Multiquadric equations of topography and other irregular surfaces. Journal of geophysical research, 76(8):1905–1915.

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, R., Yang, T., Mahdavi, M., Li, Y.-F., and Zhou, Z.-H. (2013). Improved bounds for the nyström method with application to kernel classification. IEEE Transactions on Information Theory, 59(10):6939–6949.

Kansa, E. J. (1990). Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—ii solutions to parabolic, hyperbolic and elliptic partial differential equations. Computers & mathematics with applications, 19(8-9):147–161.

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

Lindgren, F., Rue, H., and Lindström, J. (2011). An explicit link between gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. Journal of the Royal Statistical Society Series B: Statistical Methodology, 73(4):423–498.

Meng, R. and Yang, X. (2023). Sparse gaussian processes for solving nonlinear pdes. Journal of Computational Physics, 490:112340.

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, 32:8026–8037.

Quinonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. The Journal of Machine Learning Research, 6:1939–1959.

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.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. Advances in neural information processing systems, 20.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Machine learning of linear differential equations using Gaussian processes. Journal of Computational Physics, 348:683–693.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707.

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

Safdari-Vaighani, A., Heryudono, A., and Larsson, E. (2015). A radial basis function partition of unity collocation method for convection–diffusion equations arising in financial applications. Journal of Scientific Computing, 64(2):341–367.

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

Shu, C., Ding, H., and Yeo, K. (2003). Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible navier–stokes equations. Computer methods in applied mechanics and engineering, 192(7-8):941–954.

Tolstykh, A. I. and Shirobokov, D. (2003). On using radial basis functions in a "finite difference mode" with applications to elasticity problems. Computational Mechanics, 33(1):68–79.

Wang, J., Cockayne, J., Chkrebtii, O., Sullivan, T. J., and Oates, C. J. (2021). Bayesian numerical methods for nonlinear partial differential equations. Statistics and Computing, 31:1–20.

Williams, C. K. and Rasmussen, C. E. (2006). Gaussian processes for machine learning, volume 2. MIT press Cambridge, MA.

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., Yan, F., and Qi, Y. (2012). Infinite tucker decomposition: nonparametric bayesian models for multiway data analysis. In Proceedings of the 29th International Coference on International Conference on Machine Learning, pages 1675–1682.

Zhe, S., Xing, W., and Kirby, R. M. (2019). Scalable high-order gaussian process regression. In The 22nd International Conference on Artificial Intelligence and Statistics, pages 2611–2620.

Appendix

A Proof of Lemma 4.2

Since for any collocation point $\mathbf{x}_m$, $\mathcal{P}(u^*)(\mathbf{x}_m) = f(\mathbf{x}_m)$ when $\mathbf{x}_m \in \Omega$ and $\mathcal{B}(u^*)(\mathbf{x}_m) = g(\mathbf{x}_m)$ when $\mathbf{x}_m \in \partial\Omega$, we can re-write (7) as

$$\begin{cases} \underset{u\in\mathcal{U}}{\text{minimize}} \ \ \|u\|_{\mathcal{U}} \\ \text{s.t. } \frac{1}{M_\Omega}\sum_{m=1}^{M_\Omega}\left(\mathcal{P}(u)(\mathbf{x}_m) - \mathcal{P}(u^*)(\mathbf{x}_m)\right)^2 \\ +\frac{1}{M-M_\Omega}\sum_{m=M_+1}^{M}\left(\mathcal{B}(u)(\mathbf{x}_m) - \mathcal{B}(u^*)(\mathbf{x}_m)\right)^2 \le \epsilon, \\ u \text{ takes the kernel interpolation form (6).} \end{cases} \tag{19}$$

Step 1. In the first step, we show that for all ϵ above some threshold (depending on h), there exists a minimizer $u^\dagger$ for (19), and we would like also to bound the RKHS norm of $u^\dagger$, namely $\|u^\dagger\|_{\mathcal{U}}$. To this end, we utilize an intermediate optimization problem,

$$\begin{cases} \underset{u\in\mathcal{U}}{\text{minimize}} \ \ \|u\|_{\mathcal{U}} \\ \text{s.t. } u(\mathbf{x}_m) = u^*(\mathbf{x}_m), \ \ 1 \le m \le M. \end{cases} \tag{20}$$

Denote the minimizer of (20) by u_M^*. This is a standard kernel regression problem. According to the representation theorem, u_M^* takes the kernel interpolation form (6), and $\|u_M^*\|_{\mathcal{U}} \le \|u^*\|_{\mathcal{U}}$.

Since $u_M^* - u^*$ is zero at all the collocations points, according to the sampling inequality (see Proposition A.1 of (Batlle et al., 2023)), when the fill-distance h is sufficiently small (note $h_\Omega \le h$),

$$\|u_M^* - u^*\|_{H^s(\Omega)} \lesssim h^\tau \|u_M^* - u^*\|_{H^{s+\tau}(\Omega)} \tag{21}$$

where $\lesssim$ means the inequality holds with a positive constant factor multiplied by the right-hand side, and the constant is independent of the terms on both sides. Combining with (C2) of Assumption 4.1, we can obtain

$$\|\mathcal{P}(u_M^*) - \mathcal{P}(u^*)\|_{H^k(\Omega)} + \|\mathcal{B}(u_M^*) - \mathcal{B}(u^*)\|_{H^t(\partial\Omega)} \lesssim h^\tau \|u_M^* - u^*\|_{H^{s+\tau}(\Omega)}. \tag{22}$$

Since $\mathcal{U}$ is continuously embedded in $H^{s+\tau}(\Omega)$ — C(3) of Assumption 4.1, we have

$$\|u_M - u^*\|_{H^{s+\tau}(\mathcal{X})} \lesssim \|u_M - u^*\|_{\mathcal{U}}. \tag{23}$$

Combining (22), (23) and the fact $\|u_M^*\|_{\mathcal{U}} \le \|u^*\|_{\mathcal{U}}$, we have

$$\|\mathcal{P}(u_M^*) - \mathcal{P}(u^*)\|_{H^k(\Omega)} + \|\mathcal{B}(u_M^*) - \mathcal{B}(u^*)\|_{H^t(\partial\Omega)} \lesssim h^\tau \|u^*\|_{\mathcal{U}}. \tag{24}$$

According to (C2) of Assumption 4.1, since $k > \frac{d}{2}$ and $t > \frac{d-1}{2}$, according to Sobolev embedding theorem (Adams and Fournier, 2003, Theorem 4.12), both $H^k(\Omega)$ and $H^t(\partial\Omega)$ are continuously embedded into $C^0(\Omega)$ and $C^0(\partial\Omega)$, respectively. Therefore,

$$\begin{aligned} \|\mathcal{P}(u_M^*) - \mathcal{P}(u^*)\|_{C^0(\Omega)} &\lesssim \|\mathcal{P}(u_M^*) - \mathcal{P}(u^*)\|_{H^k(\Omega)}, \\ \|\mathcal{B}(u_M^*) - \mathcal{B}(u^*)\|_{C^0(\partial\Omega)} &\lesssim \|\mathcal{B}(u_M^*) - \mathcal{B}(u^*)\|_{H^t(\partial\Omega)}. \end{aligned} \tag{25}$$

At any collocation point, we obviously have

$$\begin{aligned} \left(\mathcal{P}(u_M^*) - \mathcal{P}(u^*)\right)^2 &\le \|\mathcal{P}(u_M^*) - \mathcal{P}(u^*)\|_{C^0(\Omega)}^2, \\ \left(\mathcal{B}(u_M^*)(\mathbf{x}_m) - \mathcal{B}(u^*)(\mathbf{x}_m)\right)^2 &\le \|\mathcal{B}(u_M^*) - \mathcal{B}(u^*)\|_{C^0(\partial\Omega)}^2. \end{aligned} \tag{26}$$

Combining (24), (25) and (26), we can obtain that

$$\begin{aligned} &\frac{1}{M_\Omega}\sum_{m=1}^{M_\Omega}\left(\mathcal{P}(u_M^*)(\mathbf{x}_m) - \mathcal{P}(u^*)(\mathbf{x}_m)\right)^2 \\ &+ \frac{1}{M-M_\Omega}\sum_{m=M_+1}^{M}\left(\mathcal{B}(u_M^*)(\mathbf{x}_m) - \mathcal{B}(u^*)(\mathbf{x}_m)\right)^2 \le Ch^{2\tau}\|u^*\|_{\mathcal{U}}^2, \end{aligned} \tag{27}$$

where $C > 0$ is a constant independent of h and other terms in the inequality.

The result (27) means that given the collocation points $\mathcal{M}$ and $\epsilon = Ch^{2\tau}\|u^*\|^2_{\mathcal{U}}$, the feasible region of the optimization problem (19) is nonempty and at least includes u^*_M. Therefore, the minimizer of (19) must exist and satisfy

$$\|u^\dagger\|_{\mathcal{U}} \leq \|u^*_M\|_{\mathcal{U}} \leq \|u^*\|_{\mathcal{U}}. \tag{28}$$

Step 2. Next, we analyzed the error of $\mathcal{P}(u^\dagger)$ and $\mathcal{B}(u^\dagger)$. For notation convenience, we define two error functions,

$$\begin{aligned} \xi_P(\mathbf{x}) &= \mathcal{P}(u^\dagger)(\mathbf{x}) - \mathcal{P}(u^*)(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ \xi_B(\mathbf{x}) &= \mathcal{B}(u^\dagger)(\mathbf{x}) - \mathcal{B}(u^*)(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega. \end{aligned} \tag{29}$$

We would like to bound the L^2 norm of the error functions, namely, $\|\xi_P\|_{H^0(\Omega)}$ and $\|\xi_B\|_{H^0(\partial\Omega)}$. We first consider the case for ξ_P. The idea is to decompose Ω into M_Ω regular non-overlapping regions, $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_{M_\Omega} = \Omega$, such that each region $\mathcal{T}_i$ only includes one collocation point $\mathbf{x}_i$, and its filled-distance $h_i \lesssim h$ ($1 \leq i \leq M_\Omega$). We therefore can decompose the squared L^2 norm as

$$\|\xi_P\|^2_{H^0(\Omega)} = \sum_{i=1}^{M_\Omega} \int_{\mathcal{T}_i} \xi_P(\mathbf{x})^2 \mathrm{d}\mathbf{x} = \sum_{i=1}^{M_\Omega} \|\xi_P\|^2_{H^0(\mathcal{T}_i)}. \tag{30}$$

Since according to the mean inequality,

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

we immediately obtain

$$\|\xi_P\|^2_{H^0(\mathcal{T}_i)} \lesssim \|\xi_P - \xi_P(\mathbf{x}_i)\|^2_{H^0(\mathcal{T}_i)} + \lambda(\mathcal{T}_i)\xi_P(\mathbf{x}_i)^2, \tag{31}$$

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

Since the function $\xi_P - \xi_P(\mathbf{x}_i)$ takes zero at $\mathbf{x}_i$, we can apply the sampling inequality again. That is, when the fill-distance h_i is sufficiently small, we have

$$\|\xi_P - \xi_P(\mathbf{x}_i)\|_{H^0(\mathcal{T}_i)} \lesssim h_i^k \|\xi_P - \xi_P(\mathbf{x}_i)\|_{H^k(\mathcal{T}_i)}. \tag{32}$$

Since $h_i \lesssim h$, when h is sufficiently small, we further have

$$\|\xi_P - \xi_P(\mathbf{x}_i)\|_{H^0(\mathcal{T}_i)} \lesssim h^k \|\xi_P - \xi_P(\mathbf{x}_i)\|_{H^k(\mathcal{T}_i)}, \tag{33}$$

and then using the mean inequality,

$$\|\xi_P - \xi_P(\mathbf{x}_i)\|^2_{H^0(\mathcal{T}_i)} \lesssim h^{2k}\left(\|\xi_P\|^2_{H^k(\mathcal{T}_i)} + \|\xi_P(\mathbf{x}_i)\|^2_{H^k(\mathcal{T}_i)}\right) = h^{2k}\left(\|\xi_P\|^2_{H^k(\mathcal{T}_i)} + \xi_P(\mathbf{x}_i)^2\right). \tag{34}$$

Since $\lambda(\mathcal{T}_i) \lesssim h^d$, combining (30), (31) and (34), we can obtain

$$\begin{aligned} \|\xi_P\|^2_{H^0(\Omega)} &\lesssim h^{2k} \sum_i \|\xi_P\|^2_{H^k(\mathcal{T}_i)} + (h^d + h^{2k}) \sum_i \xi_P(\mathbf{x}_i)^2 \\ &\lesssim h^{2k}\|\xi_P\|^2_{H^k(\Omega)} + (h^d + h^{2k}) \cdot M_\Omega \cdot \epsilon \end{aligned} \tag{35}$$

where ϵ comes from the constraint of (19). To ensure feasibility and to establish convergence, we set $\epsilon = Ch^{2\tau}\|u^*\|^2_{\mathcal{U}}$ as shown in (27). When $h \lesssim M^{-\frac{1}{d}}$ and is sufficiently small, we have $(h^d + h^{2k})M_\Omega \leq (h^d + h^{2k})M \leq 1 + h^{2k-d} \leq 2$ (since $k > d/2$). Therefore, we can extend the R.H.S of (35) to

$$\|\xi_P\|^2_{H^0(\Omega)} \lesssim h^{2k}\|\xi_P\|^2_{H^k(\Omega)} + h^{2\tau}\|u^*\|^2_{\mathcal{U}}. \tag{36}$$

We can follow a similar approach to show that

$$\|\xi_B\|^2_{H^0(\partial\Omega)} \lesssim h^{2t}\|\xi_B\|^2_{H^t(\partial\Omega)} + h^{2\tau}\|u^*\|^2_{\mathcal{U}}. \tag{37}$$

Combining (36) and (37),

$$\left(\|\xi_P\|_{H^0(\Omega)} + \|\xi_B\|_{H^0(\partial\Omega)}\right)^2 \lesssim h^{2\cdot\min(t,k)} \left(\|\xi_P\|_{H^k(\Omega)} + \|\xi_B\|_{H^t(\partial\Omega)}\right)^2 + h^{2\tau}\|u^*\|_{\mathcal{U}}^2. \tag{38}$$

Since $\mathcal{U} \hookrightarrow H^{s+\tau}$ — (C3) of Assumption 4.1, we have $\mathcal{U} \hookrightarrow H^s$. Leveraging (28) and (C2) of Assumption 4.1, we immediately obtain

$$\|\xi_P\|_{H^k(\Omega)} + \|\xi_B\|_{H^t(\partial\Omega)} \lesssim \|u^\dagger - u^*\|_{H^s(\Omega)} \lesssim \|u^\dagger - u^*\|_{\mathcal{U}} \lesssim \|u^*\|_{\mathcal{U}}. \tag{39}$$

Combining (38) and (39), and (C1) of Assumption 4.1, we arrive at

$$\|u^\dagger - u^*\|_{H^t(\Omega)} \lesssim h^\rho \|u^*\|_{\mathcal{U}} \tag{40}$$

where $\rho = \min(k, t, \tau)$. When $h \to 0$, obviously $u^\dagger$ converges to u^*.

B Proof of Proposition 4.3

The constraint optimization problem (7) is equivalent to the following mini-max optimization problem,

$$\min_u \max_{w \geq 0} \|u\|_{\mathcal{U}} + w \left[\frac{1}{M_\Omega} \sum_{m=1}^{M_\Omega} \left(\mathcal{P}(u)(\mathbf{x}_m) - \mathcal{P}(u^*)(\mathbf{x}_m)\right)^2 + \frac{1}{M - M_\Omega} \sum_{m=M_\Omega+1}^{M} \left(\mathcal{B}(u)(\mathbf{x}_m) - \mathcal{B}(u^*)(\mathbf{x}_m)\right)^2 - \epsilon \right]. \tag{41}$$

Suppose the feasible region is non-empty. Denote the optimum of (41) by $(u^\dagger, w^\dagger)$. Then $u^\dagger$ is a minimizer of (7). Now if we set $\alpha = \beta = w^\dagger$ in (8), and optimizing (8) will recover the minimizer $u^\dagger$.

C More Results

C.1 Point-wise Error

For a fine-grained evaluation, we examined how the point-wise error of DAKS and SKS varies along with the increase of collocation points. To this end, we altered the number of collocation points from 600, 4800 and 120K on Burgers' equation with $\nu = 0.001$, and from 600, 2400, and 40K on 2D Allen-Cahn equation with both $a = 15$ and $a = 20$. The results are shown in Fig. 1, 2 and 3. It can be seen that across all the three PDEs, the solution error of SKS decreases more and more along with the increase of collocation points. Note that for Allen-Cahn with $a = 15$, the visual difference between SKS using 2400 and 40K collocation points is little, though numerically the difference is at three orders of magnitudes (5.15E-03 *vs.* 4.44E-06). For DAKS, the point-wise error decreases substantially as the number of collocation points grows when solving Burgers' equation (see Fig. 1), but not obviously on solving Allen-Cahn equation (see Fig. 2 and 3). This is consistent with the global error shown in Table 2. This might be because the quantities of collocation point used are not sufficient to lead to a qualitative boost of DAKS. However, scaling up to much more collocation points, such as 400K, incurs a substantial increase of the computational cost.

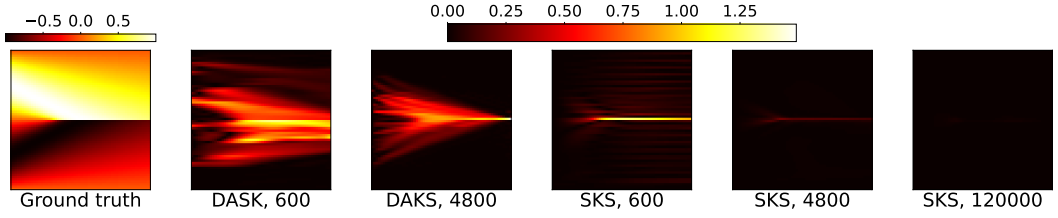

Figure 1: Point-wise solution error for Burgers' equation (15) with viscosity $\nu = 0.001$.

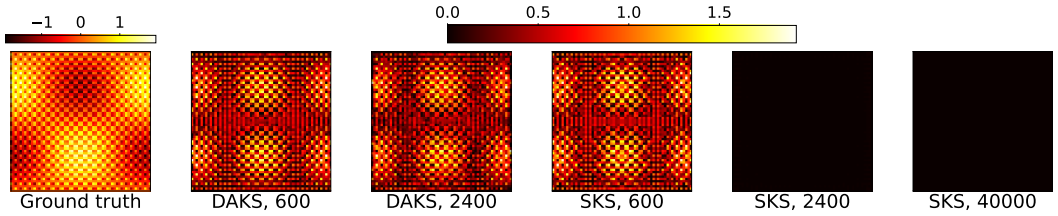

Figure 2: Point-wise solution error for 2D Allen-Cahn equation (18) with $a = 15.0$.

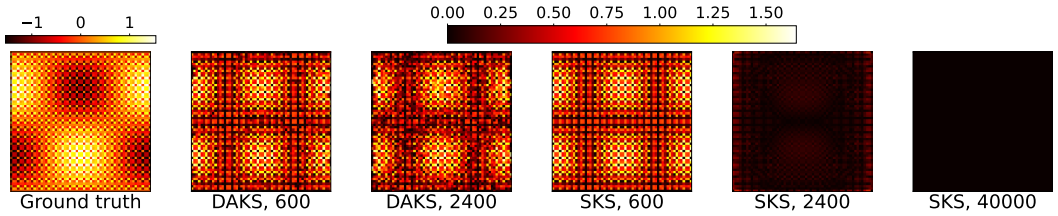

Figure 3: Point-wise solution error for 2D Allen-Cahn equation (18) with $a = 20.0$.

C.2 Ablation Study on Grid Shape

We investigated how the grid shape can influence the performance of our method. To this end, we tested on Burgers' equation with $\nu = 0.02$ and $\nu = 0.001$. For the former case, we fixed the number of collocation points to be 4800 and varied the time resolution from 10 to 80, and the spatial resolution is obtained by dividing 4800 by the time resolution and rounding up to an integer. Similarly, for $\nu = 0.001$, we fixed the number of the collocation points to 120K, and varied the time resolution from 100 to 600. We show the L^2 error of using each grid shape in Table 5. It can be seen that the grid shape does influence the error. In particular, when ν is small, *i.e.,* $\nu = 0.001$, the higher the spatial resolution, the smaller the error. The smallest error is achieved when we use the space-time resolution 1200×100. On the other hand, the time resolution seems to have much less effect on the solution accuracy. This is reasonable, because on Burgers' equation, a smaller viscosity (ν) increases the sharpness of the shock wave (spatial function). Naturally, the higher the spatial resolution, the more accurate the sharpness can be captured. In summary, we believe that in general the grid shape should be viewed as an influence factor in running our method, which needs to be carefully selected. The appropriate choice may also connect to the intrinsic property of the PDE itself.

C.3 Irregularly-Shaped Domains

We tested on solving the nonlinear elliptic PDE (16) and the Allen-Cahn equation (18) with $a = 15$. For the nonlinear elliptic PDE, the domain is an inscribed circle within $[0, 1] \times [0, 1]$. For the Allen-Cahn equation, the domain is a triangle with vertices at at $(0, 0)$, $(1, 0)$ and $(0.5, 1)$. The solution is prescribed as in our paper, with boundary conditions derived from the solution. For both PDEs, our method (SKS) used a virtual grid on $[0, 1] \times [0, 1]$ that covers the domain. For DAKS and PINN, we sampled the same number of collocation points from the domain. For a fair comparison, all the methods used the same set of 192 uniformly sampled collocation points on the boundary. The error of each method is given in Table 6. The point wise error is shown in Fig. 4. As we can see, on irregularly-shaped domains, our method SKS still obtains a reasonably good accuracy for both cases (note that the Allen-Cahn case is much more challenging).

C.4 Sensitivity to Hyper-Parameters

To examine the sensitivity to the choice of kernel parameters, we run SKS to solve nonlinear Elliptic PDE (16) and Allen-Cahn equation (18) with $a = 15$, with a varying set of length-scale parameters. For the nonlinear Elliptic PDE, we employed the grid of size 35×35 while for the Allen-Cahan equation, we used the grid of size 49×49. The results are given in Table 7. As we can see, different length-scale parameters results in changes of orders of magnitude in the solution error. For example, switching the length-scale from 0.05 to 0.1, the L^2 error for solving the nonlinear elliptic PDE

Grid shape	60×80	69×70	96×50	160×30	480×10
L^2 error	2.27E-03	4.10E-04	**7.54E-05**	1.72E-04	5.98E-04

(a) Viscosity $\nu = 0.02$, with 4800 collocation points.

Grid shape	200×600	240×500	400×300	600×200	1200×100
L^2 error	1.93E-02	6.68E-03	3.44E-03	2.28E-03	**1.63E-03**

(b) Viscosity $\nu = 0.001$ with 120000 collocation points.

Table 5: L^2 error of SKS using different grid shapes to solve Burgers' equation (15). The grid shape is depicted as "spatial-resolution $\times$ time-resolution".

L^2 Error	SKS	DAKS	PINN
Nonlinear Elliptic	8.40E-04	**4.86E-05**	4.20E-02
Allen-Cahn ($a = 15$)	**8.30E-02**	6.06E-01	1.00E+00

Table 6: L^2 error of solving PDEs on irregularly-shaped domains.

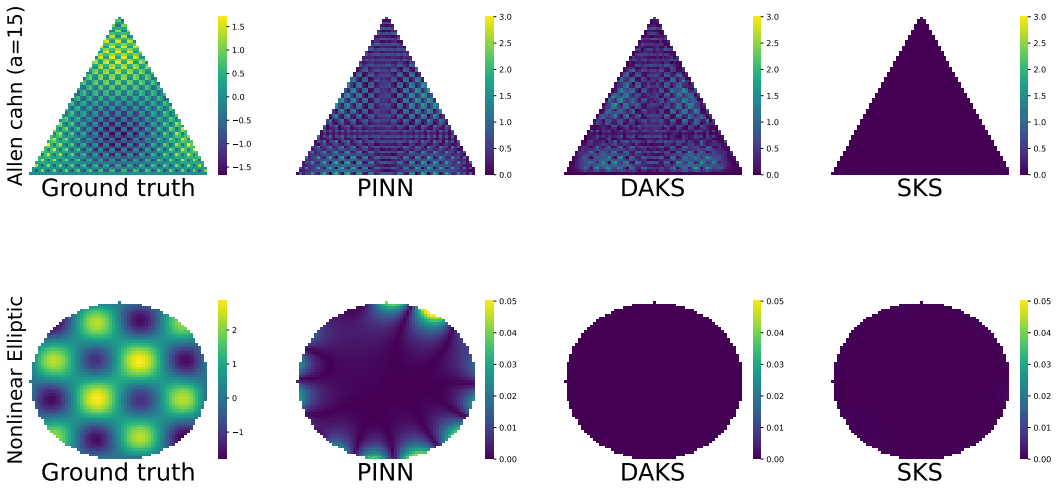

Figure 4: Point-wise solution error for Nonlinear Elliptic and 2D Allen-Cahn equation (18) with $a = 15.0$ on irregularly-shaped domains.

decreases from 1.129E-02 to 6.80E-06. Hence, our method is sensitive to the choice of the kernel parameters.

As a comparison, we also examined the sensitivity of PINN to the choice of the architectures. To this end, we fixed the depth at 8 and varied the layer width from 10 to 100, and also fixed the width at 30 while varying the depth from 3 to 30. We then tested PINN on solving the same PDEs. The solution error is reported in Table 8. One can see that when layer width is greater than 50 or the depth is beyond 8, there is no significant improvement in performance. The accuracy remains within the same magnitude with only minor variations. However, larger networks lead to substantially increased computational costs for PDE solving.

D Limitation and Discussion

Currently, the most effective training for SKS is fulfilled by stochastic optimization, namely ADAM. We need to run a large number of ADAM epochs to achieve a promising solution accuracy. It means that the PDE solving procedure is slow. The second-order optimization methods, such as L-BFGS, neither improve the solution accuracy nor accelerate the convergence. We have also tried the relaxed Gauss-Newton approach as used in DAKS. However, this method can only achieve good performance on the nonlinear elliptic PDE, and easily diverges on the other cases. This might stem from that we take derivatives (or other linear operators) over the kernel interpolation form, which makes the

Length-scale	0.05	0.1	0.2	0.3
L^2 error	1.19E-02	**6.80E-06**	4.62E-05	9.14E-04

(a) Nonlinear Elliptic PDE (16).

Length-scale	0.08	0.06	0.04	0.02
L^2 error	4.98E-01	3.83E-01	**5.15E-03**	2.19E-01

(b) Allen-Cahn Equation (18) where $a = 15$.

Table 7: L^2 error of SKS using different length scales.

Layer width	10	20	30	40	50
Nonlinear Elliptic	1.75E-01	7.68E-02	7.01E-02	3.00E-02	4.06E-02
Allan-cahn ($a = 15$)	6.31E+00	5.61E+00	7.08E+00	7.65E+00	1.03E+01

(a) L^2 error for depth=10 and layer width from 10 to 50.

Layer width	60	70	80	90	100
Nonlinear Elliptic	1.66E-02	3.58E-02	2.74E-02	2.32E-02	5.01E-02
Allan-cahn ($a = 15$)	1.30E+01	1.26E+01	1.20E+01	1.34E+01	1.09E+01

(b) L^2 error for depth=10 and layer width from 60 to 100.

Depth	3	5	8	10
Nonlinear Elliptic	3.78E-02	7.33E-02	7.01E-02	6.57E-02
Allan-cahn ($a = 15$)	1.13E+01	1.35E+01	7.08E+00	1.10E+01

(c) L^2 error for layer width=10 and depth from 3 to 10.

Depth	15	20	25	30
Nonlinear Elliptic	9.16E-02	1.13E-01	1.50E-01	7.49E-02
Allan-cahn ($a = 15$)	1.14E+00	9.30E+00	1.14E+00	9.35E+00

(d) L^2 error for layer width=10 and depth from 15 to 30.

Table 8: Sensitivity of PINN to the network width and depth.

convergence of the fixed point iterations used in DAKS much more difficult. We plan to develop novel Gauss-Newton relaxations to ensure convergence and stableness (at least in practice) so that we can further accelerate the PDE solving.

Method	2400	4800	43200
SKS (per-iter)	**4.6E-4**	**9.8E-4**	**6.8E-3**
DAKS (per-iter)	7.43	38.5	N/A
PINN (per-iter)	2.7E-1	5.2E-1	4.1E-1
SKS (total)	**22.15**	**94.25**	**2007.1**
DAKS (total)	89.14	462.18	N/A
PINN (total)	706.24	721.94	6454.7

(a) The Burgers' equation (15) with $\nu = 0.001$.

Method	2400	4800	6400	8100	22500
SKS (per-iter)	**3.6E-4**	**9.1E-4**	**1.2E-3**	**1.8E-3**	**5.9E-3**
DAKS (per-iter)	2.1	10.5	N/A	N/A	N/A
PINN (per-iter)	5.6E-2	1E-1	1.3E-1	1.5E-1	4.3E-1
SKS (total)	27.1	99.56	116.8	132.57	474.34
DAKS (total)	**16.44**	**84.18**	N/A	N/A	N/A
PINN (total)	2821	5112	6287	7614	21375

(b) Allen-Cahn equation (18) with $a = 15$.

Table 9: Runtime in seconds with respect to the number of collocation points. Note that N/A means the method is not able to run with the corresponding number of collocation points.