KP-PINNs: Kernel Packet Accelerated Physics Informed Neural Networks

Siyuan Yang [1,2], **Cheng Song** [1], **Zhilu Lai** [2,3] and **Wenjia Wang** [1] *

[1] Data Science and Analytics, The Hong Kong University of Science and Technology (Guangzhou)
[2] Internet of Things, The Hong Kong University of Science and Technology (Guangzhou)
[3] Department of CEE, The Hong Kong University of Science and Technology
{syang724, csong750}@connect.hkust-gz.edu.cn, zhilulai@ust.hk, wenjiawang@hkust-gz.edu.cn

Abstract

Differential equations are involved in modeling many engineering problems. Many efforts have been devoted to solving differential equations. Due to the flexibility of neural networks, Physics Informed Neural Networks (PINNs) have recently been proposed to solve complex differential equations and have demonstrated superior performance in many applications. While the L_2 loss function is usually a default choice in PINNs, it has been shown that the corresponding numerical solution is incorrect and unstable for some complex equations. In this work, we propose a new PINNs framework named Kernel Packet accelerated PINNs (KP-PINNs), which gives a new expression of the loss function using the Reproducing Kernel Hilbert Space (RKHS) norm and uses the Kernel Packet (KP) method to accelerate the computation. Theoretical results show that KP-PINNs can be stable across various differential equations. Numerical experiments illustrate that KP-PINNs can solve differential equations effectively and efficiently. This framework provides a promising direction for improving the stability and accuracy of PINNs-based solvers in scientific computing.

1 Introduction

Differential equations, including ordinary differential equations (ODEs) and partial differential equations (PDEs), are widely applied in engineering for modelling, analyzing, and optimizing dynamic systems [Kreyszig, 2007]. They provide a mathematical framework for describing natural phenomena and solving complex problems. Differential equations are widely used in simulating fluid dynamics [Ragheb, 1976], electromagnetics [Deschamps, 1981], heat transfer [Isachenko *et al.*, 1980] and acoustics [Kaltenbacher, 2018] Differential equations solutions are crucial for accurately predicting and optimizing the behaviour of dynamic systems, ensuring effective design and operation in engineering applications.

Because analytical solutions of differential equations are often difficult to obtain in practice, researchers seek numerical methods instead. For example, Euler's method and Runge-Kutta method [Atkinson *et al.*, 2009] can be applied to solving ODEs, and finite element method (FEM), finite difference method (FDM), finite volume method (FVM), and spectral methods can be used to solve PDEs. These methods provide a foundation for solving various differential equations encountered in engineering and scientific applications. However, such methods require substantial computational resources and memory, particularly in high-dimensional or complex physical scenarios. Moreover, they lack flexibility, as adapting to different physical problems or parameter changes often needs significant model reconfiguration [Karniadakis *et al.*, 2021].

With the development of deep learning, Physics informed neural networks (PINNs) incorporate physical laws into training neural networks to solve differential equations [Kianiharchegani, 2023], particularly PDEs, by effectively combining deep learning with domain-specific knowledge. PINNs have strong flexibility and can be effectively combined with existing traditional methods. However, PINNs can lead to incorrect solutions for some complex PDEs, affecting the accuracy of the solutions obtained [Zhang *et al.*, 2024]. Several methods are proposed to address this issue. For example, [Haitsiukevich and Ilin, 2022] used ensemble agreement for domain expansion, and the improved algorithm can make the training of PINNs more stable. [Hu *et al.*, 2021] theoretically analyzed the convergence and generalization of extended PINNs. [Yu *et al.*, 2022] proposed gradient-enhanced PINNs, which utilize the gradient information of PDEs residuals and embed the gradient into the loss function to improve the effectiveness of solving PDEs. Other works include [Nguyen *et al.*, 2023; Forootani *et al.*, 2024; Aliakbari, 2023; Yuan *et al.*, 2022; Lin *et al.*, 2022].

However, it has been noticed that the original PINNs are not stable [Wang *et al.*, 2022], due to the default L_2 loss function. To address the instability of the original PINNs, [Wang *et al.*, 2022] proposed the L_∞ loss function and proposed an adversarial training method. However, the computation of L_∞ norm is complicated, and the adversarial training is time-consuming. Another class of loss functions is based on the Sobolev norm, which is proposed by [Son *et al.*, 2021], and the corresponding method is named as Sobolev-PINNs. How-

*Corresponding author.

ever, the computation of the Sobolev norm in Sobolev-PINNs is complex, and accurate derivative information is hard to get.

In this work, we propose a novel form of loss function using the Reproducing Kernel Hilbert Space (RKHS) norm. Under specific choice of the kernel function, the RKHS norm is equivalent to the (tensored) Sobolev norm, thus addressing the stability issue of the original PINNs. The proposed method can also avoid directly computing the derivative because the RKHS norm of a function can be approximated by the RKHS norm of its interpolation, while the later has a close form. To compute the RKHS norm, we employ Kernel Packet (KP) method [Chen *et al.*, 2022], which can efficiently compute the inverse of the kernel matrix. Therefore, KP accelerated PINNs (KP-PINNs) can solve differential equations effectively and efficiently. We also provide theoretical guarantee for our proposed method.

The rest is organized as follows. Section 2 briefly introduces differential equations, PINNs and KP. Section 3 analyses the proposed KP-PINNs algorithm. Section 4 gives some theoretical results. Section 5 shows four numerical examples to demonstrate the performance of the KP-PINNs algorithm. Section 6 gives this research's discussion and future work.

2 Preliminaries

2.1 Differential Equations

Differential equations are mathematical equations that describe the relationship between a function and its derivatives, representing how a quantity changes over time or space. They are broadly classified into ordinary differential equations (ODEs), which involve functions of a single variable, and partial differential equations (PDEs), which involve functions of multiple variables. Differential equations are foundational in modeling dynamic systems and natural phenomena. By capturing the principles of change and interaction, they provide powerful tools for analyzing and solving complex problems. In general, a differential equation system can be expressed [Evans, 2022] by:

$$\begin{cases} \mathcal{L}_\theta u(\mathbf{x}) = f_\theta(\mathbf{x}), & \mathbf{x} \in \Omega \subset \mathbb{R}^d, \\ \mathcal{B}_\theta u(\mathbf{x}) = g_\theta(\mathbf{x}), & \mathbf{x} \in \partial\Omega, \end{cases} \tag{1}$$

where Ω is an open domain with boundary $\partial\Omega$, $\mathcal{L}_\theta$ represents a partial differential operator, $\mathcal{B}_\theta$ denotes the boundary condition, f_θ and g_θ are two known functions defined on Ω and $\partial\Omega$, respectively, θ is the parameter, and u is an unknown function. The goal of solving a differential equation system is to find u such that (1) holds. Usually, (1) is called ODE if $d = 1$, and PDE if $d > 1$. An example of PDEs is the second-order linear PDE, with

$$\mathcal{L}_\theta u = \sum_{i=1}^{d}\sum_{j=1}^{d} \alpha_{i,j}\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{k=1}^{d} \beta_k \frac{\partial u}{\partial x_k} + \gamma u, \tag{2}$$

$\mathbf{x} = (x_1, ..., x_d)^\top \in \mathbb{R}^d$, and the parameters are $\alpha_{i,j}, \beta_k, \gamma$ for $i, j, k \in \{1, \ldots, d\}$.

Besides the linear PDEs, where $\mathcal{L}_\theta$ is a linear differential operator, nonlinear PDEs, in which the solution or its derivatives appear nonlinearly, make them more complex to solve compared to linear PDEs. Because of the flexibility of nonlinear PDEs, it has been widely applied in fluid dynamics [Yan *et al.*, 2025], optics [Evans and Souganidis, 1989], and biology [Ghergu and Radulescu, 2011]. One particular example is the Hamilton-Jacobi-Bellman (HJB) equations, which are widely applied in electro-hydraulic systems [Guo, 2022], energy systems [Sieniutycz, 2000] and finance [Witte and Reisinger, 2011]. The HJB equations characterise the optimal control strategy for a dynamic system under certain constraints and objectives. Specifically, the form of the HJB equations [Yong and Zhou, 2012] can be expressed by

$$\begin{cases} \partial_t u(\mathbf{x}, t) + \frac{1}{2}\sigma^2 \Delta u(\mathbf{x}, t) + \min_{k\in\mathcal{K}}, \\ \quad [r(\mathbf{x}, k(\mathbf{x}, t)) + \nabla u \cdot k_t] = 0, \\ u(\mathbf{x}, T) = g(\mathbf{x}), (\mathbf{x}, t) \in \mathbb{R}^d \times [0, T], \end{cases} \tag{3}$$

where $u(\mathbf{x}, t)$ is the value function, $k(\mathbf{x}, t)$ is the given control function, $r(\mathbf{x}, k)$ is the cost rate during the process and $g(\mathbf{x})$ is the final cost at the terminal state [Wang *et al.*, 2022]. If the cost rate function is $r(\mathbf{x}, k) = a_1|k_1|^{b_1} + \cdots + a_d|k_d|^{b_d} - h(\mathbf{x}, t)$ then the HJB equations can be expressed [Yong and Zhou, 2012] as

$$\begin{cases} \mathcal{L}_{\text{HJB}} u := \partial_t u(\mathbf{x}, t) + \frac{1}{2}\sigma^2 \Delta u(\mathbf{x}, t) - \\ \quad \sum_{i=1}^{d} A_i |\partial_{\mathbf{x}_i} u|^{c_i} = h(\mathbf{x}, t), \\ \mathcal{B}_{\text{HJB}} u := u(\mathbf{x}, T) = g(\mathbf{x}), (\mathbf{x}, t) \in \mathbb{R}^d \times [0, T], \end{cases} \tag{4}$$

where $A_i = (a_i b_i)^{-\frac{1}{b_i - 1}} - a_i (a_i b_i)^{-\frac{b_i}{b_i - 1}} > 0$ with $c_i = \frac{b_i}{b_i - 1} > 1$, $\mathcal{L}_0 u := \frac{\partial}{\partial t} u - \Delta u$ is a linear operator, and $\tilde{\mathcal{L}}_{\text{HJB}} u := \mathcal{L}_0 u + \sum_{i=1}^{d} A_i |\partial_i u|^{c_i}$ is a nonlinear operator.

In the forward problem of differential equations, the goal is to obtain a function u such that (1) is satisfied, where all the operators and functions $\mathcal{L}_\theta$, $\mathcal{B}_\theta$, h_θ and g_θ are known. In practice, there are many scenarios where the parameter θ is unknown and needs to be estimated based on the equation form and observed data. This is called inverse problems, where the goal is to recover the parameters. In this work, we consider both forward problems and inverse problems.

2.2 Physics Informed Neural Networks

Physics informed neural networks (PINNs) provide a flexible way for numerically solving the differential equation systems (1).

For notational convenience, let

$$h_1 = \mathcal{L} u - f_\theta, h_2 = \mathcal{B} u - g_\theta. \tag{5}$$

Clearly, u is the solution to (1) if and only if $h_1 = h_2 = 0$. Therefore, for the forward problem, it is natural to minimize the loss function

$$\text{Loss}_{L_2} = ||h_1||^2_{L_2(\Omega)} + ||h_2||^2_{L_2(\partial\Omega)}, \tag{6}$$

which was considered by [Raissi *et al.*, 2019]. Choose $\mathbf{x}^i \in \Omega$, $i = 1, ..., N_{\mathcal{L}}$ and $\mathbf{s}^i \in \partial\Omega$, $i = 1, ..., N_{\mathcal{B}}$. Then (6) can be approximated by

$$\text{Loss}(u) = \text{Loss}_{\mathcal{L}}(u) + \text{Loss}_{\mathcal{B}}(u), \tag{7}$$

with

$$\text{Loss}_{\mathcal{L}} = \frac{1}{N_{\mathcal{L}}} \sum_{i=1}^{N_{\mathcal{L}}} h_1(\mathbf{x}^i)^2 \quad \text{and} \quad \text{Loss}_{\mathcal{B}} = \frac{1}{N_{\mathcal{B}}} \sum_{i=1}^{N_{\mathcal{B}}} h_2(\mathbf{s}^i)^2.$$

PINNs use a deep neural network, denoted by u_{nn}, to minimize the loss function in (7).

For the inverse problems, suppose that we have observed function values at points $\mathbf{x}^i \in \Omega$, $i = 1, ..., N_{\mathcal{L}}$ and $\mathbf{s}^i \in \partial\Omega$, $i = 1, ..., N_{\mathcal{B}}$, denoted by $u_d(\mathbf{x}^i)$ and $u_d(\mathbf{s}^i)$, respectively. We define the loss function in the inverse problem as

$$\text{Loss}_{\text{Inv}}(u) = \text{Loss}(u) + \text{Loss}_{\mathcal{D}}(u), \tag{8}$$

where $\text{Loss}(u)$ is as in (7), and

$$\text{Loss}_{\mathcal{D}}(u) = \frac{1}{N_{\mathcal{L}}} \sum_{i=1}^{N_{\mathcal{L}}} h_3(\mathbf{x}^i)^2 + \frac{1}{N_{\mathcal{B}}} \sum_{i=1}^{N_{\mathcal{B}}} h_3(\mathbf{s}^i)^2$$

with $h_3 = u - u_d$. The goal of solving an inverse problem is to solve

$$\min_{\theta \in \Theta} \text{Loss}_{\text{Inv}}(u), \tag{9}$$

where Θ is the parameter space. In practice, the minimizers of the loss functions in both forward and inverse problems can be found via optimization algorithms such as Adam [Kingma and Ba, 2014] and L-BFGS [Byrd *et al.*, 1995].

2.3 Kernel Packet

Our proposed method relies on *Kernel Packet* (KP), which was proposed in [Chen *et al.*, 2022]. In functional analysis, a Hilbert space of functions is termed a Reproducing Kernel Hilbert Space (RKHS) if the evaluation is a continuous linear operator at any point within the space. Assume that $\Omega \subset \mathbb{R}^d$ is compact with Lipschtiz boundary, and $\Phi : \Omega \times \Omega \to \mathbb{R}$ is a symmetric positive definite kernel function [Wang and Jing, 2022; Wang *et al.*, 2020]. Define the linear space

$$F_\Phi(\Omega) = \left\{ \sum_{i=1}^{N} \beta_i \Phi(\cdot, x_i) : \beta_i \in \mathbb{R}, x_j \in \Omega, N \in \mathbb{N} \right\}$$

and equip this space with the bilinear form

$$\left\langle \sum_{i=1}^{N} \beta_i \Phi(\cdot, x_i), \sum_{j=1}^{M} \gamma_j \Phi(\cdot, x'_j) \right\rangle_K := \sum_{i=1}^{N} \sum_{j=1}^{M} \beta_i \gamma_j \Phi(x_i, x'_j),$$

where the RKHS $\mathcal{N}_\Phi(\Omega)$ generated by kernel Φ is defined as the closure of $F_\Phi(\Omega)$ under the inner product $\langle \cdot, \cdot \rangle_\Phi$. The norm of $\mathcal{N}_\Phi(\Omega)$ is $||f||_{\mathcal{N}_\Phi(\Omega)} = \sqrt{\langle f, f \rangle_{\mathcal{N}_\Phi(\Omega)}}$, where $\langle \cdot, \cdot \rangle_{\mathcal{N}_\Phi(\Omega)}$ is induced by $\langle \cdot, \cdot \rangle_\Phi$.

Let $\mathbf{x}_1, ..., \mathbf{x}_n \in \Omega \cup \partial\Omega$ be the points of interest. KP provides an efficient way to compute the inverse of the kernel matrix $\mathbf{K} := (\Phi(\mathbf{x}_j, \mathbf{x}_k))_{jk} \in \mathbb{R}^{n \times n}$, which is essential in the application of the RKHSs, for example, prediction in kernel ridge regression, and computing the posterior variance in Bayesian optimization. In KP, it is assumed that the kernel function Φ is a Matérn kernel function [Abramowitz and Stegun, 1968] as

$$\Phi_\nu(\mathbf{x}_j, \mathbf{x}_k) = \exp\left(-\ell\sqrt{2\nu}|\mathbf{x}_j - \mathbf{x}_k|\right) \frac{(\nu - \frac{1}{2})!}{(2\nu - 1)!} \sum_{i=0}^{\nu - \frac{1}{2}} \frac{(\nu - \frac{1}{2} + i)!}{i!(\nu - \frac{1}{2} - i)!} \left(2\ell\sqrt{2\nu}|\mathbf{x}_j - \mathbf{x}_k|\right)^{\nu - \frac{1}{2} - i}, \tag{10}$$

where $\ell > 0$ is the scale parameter, and ν is the half integer smoothness parameter. For example, when ν is equal to $\frac{1}{2}$, the corresponding kernel function is $\Phi_{\frac{1}{2}}(\mathbf{x}_j, \mathbf{x}_k) = \exp(-\ell|\mathbf{x}_j - \mathbf{x}_k|)$. The Matérn kernel function is widely applied in practice; see [Muyskens *et al.*, 2024] for example.

For a moment, let us consider the case $d = 1$. Without loss of generality, assume $x_1 < x_2 < \ldots < x_n$. A nonzero function ϕ is termed an s degree KP if it can admit the representation as

$$\phi(x) = \sum_{j=1}^{n} A_j \Phi(x, x_j) \tag{11}$$

with the support of ϕ being the interval $[x_1, x_n]$. In (11), A_j is the coefficients of $\Phi(x, x_j)$, which can be obtained by solving the following linear systems:

$$\sum_{j=1}^{s} A_j x_j^l \exp(\delta c x_j) = 0, \tag{12}$$

where $l = 0, ..., \frac{s-3}{2}$, $\delta = \pm 1$, $c^2 = \frac{2\nu}{\ell^2}$ (ν and ℓ are the smoothness parameter and scale parameter of Matérn kernel function respectively). By (11), we have $\mathbf{AK} = \phi(\mathbf{x})$, and the inverse of the kernel matrix can be computed by

$$\mathbf{K}^{-1} = \mathbf{A}\phi(\mathbf{x})^{-1}, \tag{13}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top$ is the input vector, and the $(i, j)^{th}$ entry of $\phi(\mathbf{X})$ and $\mathbf{A}$ are $\phi_j(x_i)$ and the i element of A_j, respectively.

It has been shown in [Chen *et al.*, 2022] that both $\mathbf{A}$ and $\phi(\mathbf{x})$ are sparse banded matrices, hence the computation of $\mathbf{K}^{-1}$ can be fast. Specifically, [Chen *et al.*, 2022] showed that computing $\mathbf{A}$ and $\phi(\mathbf{x})$ needs only $O((2\nu + 2)^3 n)$ computation time and requires storage space of $O((2\nu+2)n)$, where ν is usually small in most practical applications. LU decomposition [Davis, 2006] can be applied for solving $[\phi(\mathbf{x})]^{-1}$, and [Chen *et al.*, 2022] showed that the total computation time is only $O(n^2)$.

If $d > 1$, i.e., the input $\mathbf{X}$ is multi-dimensional, then special structure of the input points $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_d})$ is needed. KP requires the input points to be tensor-like, that is, the input points can be described as the Cartesian combination of multiple one-dimensional point collections. Specifically, $\mathbf{X}^{FG} = \times_{i=1}^{N} \mathbf{x}^{(i)}$, where each $\mathbf{x}^{(i)}$ represents a distinct one-dimensional point collection. With tensor-like input points, $\mathbf{K}$ can be decomposed into Kronecker products as

$$\mathbf{K} = \bigotimes_{i=1}^{d} \mathbf{K}_{\mathbf{x_i}}, \tag{14}$$

where $\mathbf{K}_{\mathbf{x_i}} = (\Phi(x_{ij}, x_{ik})), 1 \leqslant i \leqslant d, 1 \leqslant j, k \leqslant n_i$, n_i is the number of points for dimension i and thus the dimension of $\mathbf{K}$ is $n \times n$, where $n = \prod_{i=1}^{d} n_i$. Then the inverse $\mathbf{K}$ can be directly computed by

$$\mathbf{K}^{-1} = \left[\bigotimes_{i=1}^{d} \mathbf{K}_{\mathbf{x_i}} \right]^{-1} = \bigotimes_{i=1}^{d} \mathbf{K}_{\mathbf{x_i}}^{-1}, \tag{15}$$

where each $\mathbf{K}_{\mathbf{x_i}}^{-1}$ can be computed using one-dimensional KP. Putting all things together, we obtain

$$\mathbf{K}^{-1} = \bigotimes_{i=1}^{d} \mathbf{K}_{\mathbf{x_i}}^{-1} = \bigotimes_{i=1}^{d} \mathbf{A}_{\mathbf{x_i}} \phi(\mathbf{x_i})^{-1}, \tag{16}$$

whose computation time is still $O(n^2)$.

3 KP Accelerated PINNs

In this section, we introduce the proposed method, KP accelerated PINNs (KP-PINNs). Instead of using L_2 loss in (6), which was shown to be unstable in some specific PDE systems, we consider tensor Sobolev norm. The tensor Sobolev space $H_T^\nu(\Omega)$ [Hochmuth *et al.*, 2000] is defined as the set of functions $f : \Omega \to \mathbb{R}$ satisfying

$$H_T^\nu(\Omega) = \left\{ f \in L^2(\Omega, \mathbb{R}) \,\middle|\, ||f||_{H_T^\nu(\Omega)} < \infty \right\}, \tag{17}$$

where $||f||_{H_T^\nu(\Omega)}$ is the norm on $H_T^\nu(\Omega)$, explicitly defined via the Fourier transform as

$$||f||^2_{H_T^\nu(\Omega)} = \int_{\mathbb{R}^d} \left| \tilde{f}(\omega) \right|^2 \prod_{i=1}^{d} \left(1 + |\omega_i|^2\right)^\nu d\omega, \tag{18}$$

where $\tilde{f}(\omega)$ represents the Fourier transform of f. We define the loss function with tensor Sobolev norm as

$$\text{Loss}_{H_T^\nu} = ||\mathcal{L}u_{\text{NN}} - f||^2_{H_T^\nu(\Omega)} + \lambda ||\mathcal{B}u_{\text{NN}} - g||^2_{H_T^\nu(\partial\Omega)}, \tag{19}$$

where $\lambda > 0$ is a tuning parameter that balances the weights between the difference between the space and its boundary, and u_{NN} is the predicted value obtained from the neural network. However, directly computing the tensor Sobolev norm may be time-consuming and inaccurate. By the equivalence between the RKHS norm and tensor Sobolev norm, we modify the loss function (19) to

$$\text{Loss}_{\mathcal{N}_{\Phi_T^\nu}} = ||\mathcal{L}u_{\text{NN}} - f||^2_{\mathcal{N}_{\Phi_T^\nu(\Omega)}} + \lambda ||\mathcal{B}u_{\text{NN}} - g||^2_{\mathcal{N}_{\Phi_T^\nu(\partial\Omega)}}, \tag{20}$$

where $\mathcal{N}_{\Phi_T^\nu}$ is tensor RKHS, i.e. the RKHS generated by $\Phi_T^\nu(x) = \prod_{i=1}^d \Phi^\nu(x_i)$. Tensor RKHS and Sobolev space are widely considered for complexity reduction in high-dimensional spaces [Ding *et al.*, 2020; Kühn *et al.*, 2015; Dũng and Nguyen, 2021].

Computing the loss function in (20) can be efficient via KP. Specifically, let

$$h_{1\text{NN}} = \mathcal{L}u_{\text{NN}} - f, h_{2\text{NN}} = \mathcal{B}u_{\text{NN}} - g, \tag{21}$$

and we consider $||h_{1\text{NN}}||^2_{\mathcal{N}_{\Phi_T^\nu(\Omega)}} = ||\mathcal{L}u_{\text{NN}} - f||^2_{\mathcal{N}_{\Phi_T^\nu(\Omega)}}$ first. Based on the Theorem 11.23 in [Wendland, 2004], $||h_{1\text{NN}}||^2_{\mathcal{N}_{\Phi_T^\nu(\Omega)}}$ can be approximated by

$$\mathbf{y}_{h1}^T \mathbf{K}^{-1} \mathbf{y}_{h1}, \tag{22}$$

where $\mathbf{y}_{h1} = (h_{1\text{NN}}(\mathbf{x^1}), ..., h_{1\text{NN}}(\mathbf{x^n}))$ and $\mathbf{x^1}, ..., \mathbf{x^n} \in \mathbb{R}^d$ means all the training data. $\mathbf{K}$ is as in (14). The other term $||h_{2\text{NN}}||^2_{\mathcal{N}_{\Phi_T^\nu(\partial\Omega)}}$ can be approximated in a similar manner.

Algorithm 1 KP-PINNs

Input: PDE systems (1), iterations n_{iter}, known points
Output: Approximated solution $\hat{u}(\mathbf{x})$ and equations' parameters (if inverse problem)

```
Initialize neural network parameters.
while i < n_iter do
    Forward pass and compute the predicted value.
    Compute derivatives using automatic differentiation.
    Compute A and φ(x) by (12) and (11).
    Compute the loss of KP-PINNs based on (23).
    Update parameters.
end while
```

With (13), (16) and approximation (22), the loss function (20) can be approximated by:

$$\text{Loss}_{\mathcal{N}_{\Phi_T^\nu}} \approx \mathbf{y}_{h1}^T \mathbf{A}_{h1} \phi(\mathbf{x})_{h1}^{-1} \mathbf{y}_{h1} + \lambda \mathbf{y}_{h2}^T \mathbf{A}_{h2} \phi(\mathbf{x})_{h2}^{-1} \mathbf{y}_{h2}. \tag{23}$$

By KP method introduced in Section 2.3, $\text{Loss}_{\mathcal{N}_{\Phi_T^\nu}}$ can be easily computed.Some well-established optimization algorithms, such as Adam or L-BFGS, can be used to minimize the loss effectively. The KP-PINNs algorithm is summarized in Algorithm 1. For the inverse problem, comparing (7) and (8), it can be seen that the loss function of the inverse problem has more $\text{Loss}_{\mathcal{D}}(u)$ part. For this part, the computation of the RKHS norm is similar to the analysis above. In this way, the KP-PINNs algorithm can effectively solve both forward and inverse problems.

4 Theoretical Results

In this section, we analyze the stability of PDEs solved with the loss function (19), or equivalently (20). The stability of a PDE system is defined as follows [Wang *et al.*, 2022].

Definition 1 (Stability of PDEs [Wang *et al.*, 2022])**.** *Suppose Z_1, Z_2, and Z_3 are three Banach spaces. For the exact solution $u^*(\mathbf{x})$ and the predicted solution u, if for $||\mathcal{L}u - f||_{Z_1}, ||\mathcal{B}u - g||_{Z_2} \to 0$, it has $||u^* - u||_{Z_3} = O(||\mathcal{L}u - f||_{Z_1} + ||\mathcal{B}u - g||_{Z_2})$, then the PDEs* (1) *is (Z_1, Z_2, Z_3)-stable.*

Stability defined in Definition 1 provides a theoretical guarantee for solving a PDE. Specifically, if the loss function is $||\mathcal{L}u - f||_{Z_1} + ||\mathcal{B}u - g||_{Z_2}$, then minimizing the loss function can ensure the convergence of the solution under Z_3-norm. In this section, we consider two classes of PDEs: second-order linear elliptic equations and Hamilton-Jacobi-Bellman (HJB) equations. We start with the second-order linear elliptic equations satisfying the following assumption. While in the previous section, we considered the parameters of the PDE as constants, here we extend the analysis to a more general case, where the parameters are functions. This allows us to establish the theoretical results in a broader context.

Assumption 1. *Suppose in* (1)*, the operator $\mathcal{L} = \sum_{i,j=1}^d a_{i,j}(\mathbf{x}) \partial_{x_i} \partial_{x_j} + \sum_{i=1}^d b_i(\mathbf{x}) \partial_{x_i} + c(x)$ and $\mathcal{B}$ is the identity operator. The functions $a_{i,j} \in C^2$ satisfy the uniformly elliptic condition. In addition, the only solution with zero input data is the zero solution.*

Lemma 1 (Theorem 2.1 in [Bramble and Schatz, 1970])**.** *In addition to Assumption* (1)*, assume that $\mathcal{L}$ is with C^∞ coefficients, defined on C^∞ bounded domain Ω on $\mathbb{R}^d$. For any real number l, $||u||_{H^l(\Omega)} \leqslant C\left(||\mathcal{L}u||_{H^{l-2}(\Omega)} + ||u||_{H^{l-\frac{1}{2}}(\partial\Omega)}\right)$ for all $u \in C^\infty(\bar{\Omega})$, where $\bar{\Omega}$ is a closure of Ω and C is independent of u.*

Theorem 1 (Stability of second-order linear elliptic equation)**.** *Suppose the RKHS $\mathcal{N}_{\Phi_T^1}$ coincides with the space H_T^1. For any C^∞ bounded domain $\Omega \subseteq \mathbb{R}^d$, if Assumption 1 is satisfied and all the coefficient functions are in C^∞, then the equation*

$$\begin{cases} \mathcal{L}_\theta u(\mathbf{x}) = h_\theta(\mathbf{x}), & \mathbf{x} \in \Omega \subset \mathbb{R}^d, \\ \mathcal{B}_\theta u(\mathbf{x}) = g_\theta(\mathbf{x}), & \mathbf{x} \in \partial\Omega \end{cases}$$

is $(\mathcal{N}_{\Phi_T^1}(\Omega), \mathcal{N}_{\Phi_T^1}(\partial\Omega), H^1(\Omega))$-stable.

Proof. Sets $l = 1$, by Lemma 1, we have

$$\begin{aligned} ||u||_{H^1(\Omega)} &\leqslant C_1(||\mathcal{L}u||_{H^{-1}(\Omega)} + ||u||_{H^{\frac{1}{2}}(\partial\Omega)}) \\ &\leqslant C_2(||\mathcal{L}u||_{L^2(\Omega)} + ||\mathcal{B}u||_{H^1(\partial\Omega)}) \\ &\leqslant C_3(||\mathcal{L}u||_{H^1(\Omega)} + ||\mathcal{B}u||_{H^1(\partial\Omega)}) \\ &\leqslant C_4(||\mathcal{L}u||_{H_T^1(\Omega)} + ||\mathcal{B}u||_{H_T^1(\partial\Omega)}). \end{aligned}$$

Since the RKHS $\mathcal{N}_{\Phi_T^1}$ coincides with the space H_T^1, we have that $||u||_{H^1(\Omega)} \leqslant C_4(||\mathcal{L}u||_{\mathcal{N}_{\Phi_T^1}(\Omega)} + ||\mathcal{B}u||_{\mathcal{N}_{\Phi_T^1}(\partial\Omega)})$. □

The proof of Theorem 1 is a natural result derived from the classical theory of second-order linear elliptic equations [Shin *et al.*, 2023]. It shows that the numerical solution for the second-order linear elliptic equation is stable. Next, we show the stability of the HJB equation (4).

Lemma 2 (Stability of HJB equation - L_∞ norm [Wang *et al.*, 2022])**.** *For $p, q \geqslant 1$, let $r_0 = \frac{(d+2)q}{d+q}$. Suppose the subsequent inequalities are valid for p, q and r_0:*

$$p \geqslant \max\left\{2, \left(1 - \frac{1}{\check{c}}\right) d\right\}, q > \frac{(\check{c}-1)d^2}{(2-\check{c})d + 2}, \frac{1}{r_0} \geqslant \frac{1}{p} - \frac{1}{d},$$

where $\check{c} = \max_{1 \leqslant i \leqslant d} c_i$ in (4)*. For any $r \in [1, r_0)$, any bounded open set $\Omega \subset \mathbb{R}^d \times [0, T]$, and for a large R such that $\Omega \subset B_R \times [0, T]$, where B_R is a ball with radius R,* (4) *maintains $(L_p(\Omega), L_q(B_R), W^{1,r}(\Omega))$-stability when $\check{c} \leqslant 2$.*

Theorem 2 (Stability of HJB equation)**.** *For $p, q \geqslant 1$, let $r_0 = \frac{(d+2)q}{d+q}$. Suppose the subsequent inequalities are valid for p, q and r_0:*

$$p \geqslant \max\left\{2, \left(1 - \frac{1}{\check{c}}\right) d\right\}, q > \frac{(\check{c}-1)d^2}{(2-\check{c})d + 2}, \frac{1}{r_0} \geqslant \frac{1}{p} - \frac{1}{d},$$

where $\check{c} = \max_{1 \leqslant i \leqslant d} c_i$ in (4)*. Then, for any $r \in [1, r_0)$, $\nu_1, \nu_2 \geqslant 1$ and any bounded open set $\Omega \subset \mathbb{R}^d \times [0, T]$,* (4) *maintains $(\mathcal{N}_{\Phi_T^{\nu_1}}(\mathbb{R}^d \times [0, T]), \mathcal{N}_{\Phi_T^{\nu_2}}(\mathbb{R}^d), W^{1,r}(\Omega))$-stable when $\check{c} \leqslant 2$.*

Proof. First of all, according to the Sobolev embedding theorem [Adams and Fournier, 2003; Wendland, 2004; Ding *et al.*, 2019], when bounded set $\Omega \subset \mathbb{R}^d$, $p < \infty$, $\forall f \in H^1$, $\exists c_1, c_2, c_3, c_4, c_5 > 0$, it has the norm inequality

$$\begin{aligned} c_1||f||_{L_p(\Omega)} \leqslant c_2||f||_{L_\infty(\Omega)} \leqslant c_3||f||_{H^1(\Omega)} \\ \leqslant c_4||f||_{H_T^1(\Omega)} \leqslant c_5||f||_{\Phi_T^1(\Omega)}. \end{aligned} \tag{24}$$

If $||\mathcal{L}_{\text{HJB}}u(\mathbf{x}) - \varphi(\mathbf{x})||_{H^{\nu_1}}, ||\mathcal{B}_{\text{HJB}}u(\mathbf{x}) - g(\mathbf{x})||_{H^{\nu_2}} \to 0$, known by (24), $||\mathcal{L}_{\text{HJB}}u(\mathbf{x}) - \varphi(\mathbf{x})||_{L_p}, ||\mathcal{B}_{\text{HJB}}u(\mathbf{x}) - g(\mathbf{x})||_{L_q} \to 0$. By Lemma 2, (4) is stable. It can be represented as $(L_p(\Omega), L_q(B_R), W^{1,r}(\omega))$-stable, and thus $(H^{\nu_1}(\mathbb{R}^d \times [0, T]), H^{\nu_2}(\mathbb{R}^d), W^{1,r}(\Omega))$-stable. Additionally, the equation is also $(\mathcal{N}_{\Phi_T^1}(\mathbb{R}^d \times [0, T]), \mathcal{N}_{\Phi_T^1}(\mathbb{R}^d), W^{1,r}(\Omega))$-stable, thus $(\mathcal{N}_{\Phi_T^{\nu_1}}(\mathbb{R}^d \times [0, T]), \mathcal{N}_{\Phi_T^{\nu_2}}(\mathbb{R}^d), W^{1,r}(\Omega))$-stable. □

Theorem 2 is a modified version of Theorem 4.3 in [Wang *et al.*, 2022]. It demonstrates that when the dimension of the state function d is large, selecting an appropriate RKHS ensures the stability of the equations.

5 Experiments

In this section, we apply KP-PINNs to four representative differential equations: an ODE - Stiff equation, a second-order linear PDE - Helmholtz equation, a HJB equation - LQG equation, and a nonlinear PDE - NS equation. For each differential equation, both the forward and inverse problems are addressed. We compare the KP-PINNs algorithm with several baseline PINNs approaches including L_2-PINNs, RKHS-PINNs, and Sobolev-PINNs, as summarized in Table 1. The implementation details and source code are available at: https://github.com/SiyuanYang-sy/KP-PINNs.

We consider the accuracy and computational time, where the accuracy is measured by the relative L_2 error defined by

$$||e||_{\text{relative } L_2} = \left(\frac{\sum_{i=1}^{N_{\text{test}}} |\hat{u}(\mathbf{x}_i) - u(\mathbf{x}_i)|^2}{\sum_{i=1}^{N_{\text{test}}} |u(\mathbf{x}_i)|^2}\right)^{\frac{1}{2}}, \tag{25}$$

where $u(\mathbf{x}_i)$ represents the true solution at $\mathbf{x}_i$, $\hat{u}(\mathbf{x}_i)$ denotes the prediction obtained from the algorithm, and N_{test} is the size of the test set. Also, the standard error (SE) is given to reflect the variability across multiple experiments.

5.1 Stiff Equation

The Stiff equation [Wanner and Hairer, 1996] exhibits steep gradients caused by fast dynamics, and is commonly treated with time-scale decomposition to reduce numerical and

Algorithm	Loss function
KP-PINNs ($\mathcal{N}_{\Phi_T^\nu}$ norm)	(23), ν equals to $\frac{1}{2}, \frac{3}{2}, \frac{5}{2}$
L_2-PINNs (L_2 norm)	$\frac{1}{n}\sum_{i=1}^n (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2$
RKHS-PINNs ($\mathcal{N}_{\Phi_T^\nu}$ norm)	(22), ν equals to $\frac{1}{2}, \frac{3}{2}, \frac{5}{2}$
Sobolev-PINNs (H_T^ν norm)	(18), ν equals to $1, 2, 3$

Table 1: KP-PINNs and the comparing algorithms.

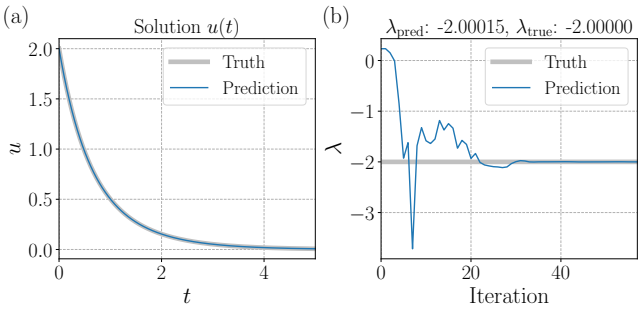

Figure 1: KP-PINNs ($\nu = \frac{3}{2}$) inverse results of Stiff equation. (a) Predicted $u(t)$. (b) Predicted λ.

reaction-related errors. It is defined by

$$u'(t) - \lambda u(t) = e^{-t}, t \in [0, 5], u(0) = \mu, \tag{26}$$

where λ and μ are two parameters. Here we set $\lambda = -2.0$ and $\mu = 2.0$. The analytic solution to (26) is

$$u(t) = \left(\mu + \frac{1}{1+\lambda}\right) e^{\lambda t} - \frac{e^{-t}}{1+\lambda}. \tag{27}$$

There is only one initial point so $N_{\mathcal{B}} = 1$, and $N_{\mathcal{L}} = 50$ in (7) in the forward problem. In the inverse problem, we assume that λ is unknown and set $N_{\mathcal{B}} + N_{\mathcal{L}} = 50$. In both cases N_{test} is equal to 2000. Table 2 provides the average results of three independent runs. KP-PINNs and RKHS-PINNs yield similar results due to the same loss function form, but differ in the computation of $\mathbf{K}^{-1}$, leading to the run-time gap observed in Table 6, where KP-PINNs demonstrate significantly faster computation across all four equations. The Sobolev-PINNs algorithm uses automatic differentiation to compute first- and higher-order derivatives, leading to error accumulation and difficulty in obtaining accurate results.

5.2 Helmholtz Equation

The Helmholtz equation is a second-order elliptic PDE commonly encountered in electromagnetism that describes the spatial behavior of electromagnetic wave propagation. It is defined by

Algorithm (-PINNs)	Forward	SE	Inverse	SE	Inv λ(-2.0)
KP ($\nu = \frac{1}{2}$)	2.23e-04	4.90e-05	3.95e-04	1.37e-04	-1.99983
KP ($\nu = \frac{3}{2}$)	5.57e-04	3.83e-04	5.62e-04	2.79e-04	-2.00015
KP ($\nu = \frac{5}{2}$)	1.55e-04	6.64e-05	3.25e-04	1.43e-04	-1.99991
L_2	3.31e-04	**4.13e-05**	3.32e-04	**1.34e-04**	**-1.99995**
RKHS ($\nu = \frac{1}{2}$)	2.90e-04	8.19e-05	3.93e-04	1.35e-04	-1.99983
RKHS ($\nu = \frac{3}{2}$)	5.83e-04	3.70e-04	3.75e-04	3.25e-04	-1.99984
RKHS ($\nu = \frac{5}{2}$)	**1.29e-04**	5.09e-05	**3.24e-04**	1.41e-04	-1.99991
Sobolev ($\nu = 1$)	5.62e-02	4.57e-02	4.38e-01	8.75e-03	-1.05663
Sobolev ($\nu = 2$)	3.58e-01	1.38e-02	4.12e-01	1.94e-02	-1.29371
Sobolev ($\nu = 3$)	3.56e-01	8.57e-03	4.29e-01	2.60e-02	-1.64380

Table 2: Experimental results of Stiff equation. These six columns represent algorithm types, forward error results, forward SE results, inverse error results, inverse SE results and the parameter to be estimated in the inverse problems. The following tables are also represented in this way.

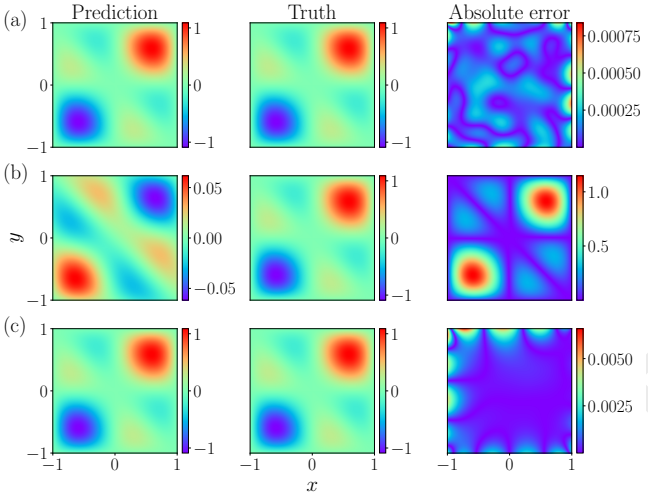

Figure 2: Comparison of predicted $u(x, y)$ for Helmholtz equation using different methods. (a) KP-PINNs ($\nu = \frac{3}{2}$) forward results. (b) L_2-PINNs inverse results. (c) KP-PINNs ($\nu = \frac{1}{2}$) inverse results.

$$\begin{cases} \Delta u + k^2 u = p, (x, y) \in \Omega = [-1, 1]^2, \\ u(x, y) = 0, (x, y) \in \partial\Omega, \end{cases} \tag{28}$$

where $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$, k is the wave number, p is the source term (excitation term) [Li *et al.*, 2022] defined by $p(x, y) = (x + y)\sin(\pi x)\sin(\pi y) - 2\pi^2(x + y)\sin(\pi x)\sin(\pi y) + 2\pi\cos(\pi y)\sin(\pi x) + 2\pi\cos(\pi x)\sin(\pi y)$. Here, we set $k = 1.0$. The analytic solution for $k = 1.0$ [Jagtap *et al.*, 2020] is

$$u(x, y) = (x + y)\sin(\pi x)\sin(\pi y). \tag{29}$$

In this experiment, $N_{\mathcal{B}} = 100$ and $N_{\mathcal{L}} = 100 \times 100$ in (7). For the inverse problem, parameter k is unknown and $N_{\mathcal{B}} + N_{\mathcal{L}} = 30 \times 30$ in (7). In both cases N_{test} is equal to 500×600. Table 3 gives the average results of five times. It can be seen that the first three algorithms can get the solutions well. For inverse problems, only KP-PINNs ($\nu = \frac{1}{2}$) and RKHS-PINNs ($\nu = \frac{1}{2}$) can predict the k relatively correctly, while the L_2-PINNs fails to work. Figure 2 (b), Figure 2 (c) and Figure 3 (a) show the inverse problem results of L_2-PINNs and KP-PINNs ($\nu = \frac{1}{2}$). The parameter ν determines the degree of smoothness for the kernel function, and a smaller value of ν is suitable for less smooth conditions. It can be seen that a smaller ν (such as $\nu = \frac{1}{2}$) can get better results.

Algorithm (-PINNs)	Forward	Standard	Inverse	Standard	Inv k (1.0)
KP ($\nu = \frac{1}{2}$)	1.13e-03	1.97e-04	4.88e-03	1.10e-03	0.98262
KP ($\nu = \frac{3}{2}$)	**9.13e-04**	1.91e-04	1.26e-02	7.35e-03	0.83391
KP ($\nu = \frac{5}{2}$)	1.15e-03	1.91e-04	2.75e-02	7.20e-03	0.51944
L_2	2.99e-03	3.06e-04	2.36e-01	2.05e-01	4.37582
RKHS ($\nu = \frac{1}{2}$)	2.10e-03	9.61e-04	**3.37e-03**	**6.26e-04**	**0.99927**
RKHS ($\nu = \frac{3}{2}$)	1.02e-03	**1.23e-04**	1.68e-02	7.58e-03	0.77236
RKHS ($\nu = \frac{5}{2}$)	1.10e-03	1.65e-04	1.88e-02	7.34e-03	0.69293
Sobolev ($\nu = 1$)	1.79e+00	1.01e+00	7.41e-01	4.82e-01	0.00475
Sobolev ($\nu = 2$)	8.95e-01	2.09e-02	3.85e-01	1.64e-02	0.00221
Sobolev ($\nu = 3$)	1.31e+01	8.85e+00	8.20e-01	1.00e-01	0.01159

Table 3: Experimental results of Helmholtz equation.

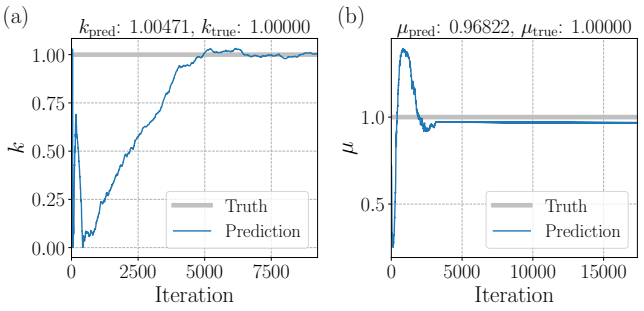

Figure 3: KP-PINNs ($\nu = \frac{1}{2}$) inverse results. (a) Helmholtz equation - predicted k. (b) LQG equation - predicted μ.

5.3 Linear Quadratic Gaussian (LQG) Equation

One application of the HJB equation is LQG control problem. The form of LQG [Han *et al.*, 2018] in d dimensions are

$$\begin{cases} \partial_t u(\mathbf{x},t) + \Delta u(\mathbf{x},t) - \mu||\nabla_{\mathbf{x}} u(\mathbf{x},t)||^2 = 0, \\ \mathbf{x} \in \mathbb{R}^d, t \in [0,T], \\ u(\mathbf{x},T) = g(\mathbf{x}), \mathbf{x} \in \mathbb{R}^d, \end{cases} \tag{30}$$

with the solution

$$u(\mathbf{x},t) = -\frac{1}{\mu} \ln \left(\int_{\mathbb{R}^d} (2\pi)^{\frac{d}{2}} \mathrm{e}^{-\frac{\|y\|^2}{2}} \mathrm{e}^{-\mu g(\mathbf{x} - \sqrt{2(T-t)}y)} \mathrm{d}y \right), \tag{31}$$

where $g(\mathbf{x}) = \ln\left(\frac{1+||\mathbf{x}||^2}{2}\right)$ is the terminal cost function.

We set $d = 2, \mu = 1.0, T = 1.0, N_{\mathcal{B}} = 2000$ and $N_{\mathcal{L}} = 30 \times 20 \times 10$ in (7) in the forward problem. For the inverse problem, μ is unknown and $N_{\mathcal{B}} + N_{\mathcal{L}} = 10 \times 10 \times 10$. In both cases, N_{test} is equal to $100 \times 80 \times 50$. Table 4 gives the average results of three times experiments. For the inverse problem, KP-PINNs with $\nu = \frac{1}{2}$ can obtain the closest estimation among all algorithms. Figure 3 (b) gives a one-case inverse problem results of KP-PINNs ($\nu = \frac{1}{2}$).

5.4 Navier-Stokes (NS) Equation

The widely-used NS equations are fundamental PDEs that describe the motion of viscous and incompressible fluids. In this experiment, the two-dimensional incompressible NS equations are considered to model the evolution of fluid flow. The governing equation for the vorticity $\omega(t,x,y)$ is

$$\begin{aligned} &\omega_t + u\omega_x + v\omega_y = \mu(\omega_{xx} + \omega_{yy}), \\ &t \in [0,T], x \in [x_{\text{start}}, x_{\text{end}}], y \in [y_{\text{start}}, y_{\text{end}}], \end{aligned} \tag{32}$$

Algorithm (-PINNs)	Forward	Standard	Inverse	Standard	Inv μ (1.0)
KP ($\nu = \frac{1}{2}$)	**6.70e-02**	2.43e-02	**6.90e-03**	8.84e-04	**0.93244**
KP ($\nu = \frac{3}{2}$)	2.26e-01	8.56e-02	1.81e-02	1.45e-03	1.26715
KP ($\nu = \frac{5}{2}$)	3.46e-01	1.89e-02	1.88e-02	2.29e-03	1.27243
L_2	2.39e-01	9.80e-02	1.77e-02	1.36e-03	1.20942
RKHS ($\nu = \frac{1}{2}$)	1.71e-01	6.90e-02	8.25e-03	2.28e-03	0.93072
RKHS ($\nu = \frac{3}{2}$)	3.21e-01	8.89e-03	1.88e-02	2.17e-03	1.27210
RKHS ($\nu = \frac{5}{2}$)	3.39e-01	5.54e-03	1.89e-02	1.94e-03	1.25183
Sobolev ($\nu = 1$)	1.42e+00	3.43e-02	6.05e-01	9.27e-02	-1.21866
Sobolev ($\nu = 2$)	1.43e+00	**3.37e-04**	7.12e-03	**9.39e-05**	-129.13841
Sobolev ($\nu = 3$)	7.05e-02	3.33e-02	1.51e-02	3.12e-03	1.16909

Table 4: Experimental results of LQG equation.

Algorithm (-PINNs)	Forward	Standard	Inverse	Standard	Inv μ (0.01)
KP ($\nu = \frac{1}{2}$)	**1.66e-01**	9.90e-03	**3.61e-02**	1.75e-02	**0.01176**
L_2	3.97e-01	1.18e-01	1.40e-01	5.46e-02	0.01233
RKHS ($\nu = \frac{1}{2}$)	1.73e-01	**6.66e-03**	4.87e-02	**1.07e-02**	0.01201

Table 5: Experimental results of NS equation. As demonstrated in the experiments above, a smaller value of ν is more suitable for less smooth conditions. For the NS equation, KP-PINNs ($\nu = \frac{1}{2}$) and RKHS-PINNs ($\nu = \frac{1}{2}$) are employed to evaluate the performance.

Algorithm (-PINNs)	Stiff	Helmholtz	LQG	NS
KP ($\nu = \frac{1}{2}$)	**0.2971**s	**5.3152**s	**90.7398**s	**26.9730**s
RKHS ($\nu = \frac{1}{2}$)	2.6984s	42.1662s	431.0374s	56.9642s

Table 6: Time of computing $\mathbf{K}^{-1}$ ($\nu = \frac{1}{2}$ as example). $N_{\mathcal{B}} + N_{\mathcal{L}} = 2000$ for Stiff equation. $N_{\mathcal{B}} + N_{\mathcal{L}} = 200 \times 300$ for Helmholtz equation. $N_{\mathcal{B}} + N_{\mathcal{L}} = 50 \times 50 \times 40$ for LQG equation. $N_{\mathcal{B}} + N_{\mathcal{L}} = 60 \times 40 \times 20$ for NS equation.

where μ is the viscosity coefficient. In our experiment, we set $\mu = 0.01, T = 1.9, x_{\text{start}} = 1.0, x_{\text{end}} = 8.0, y_{\text{start}} = -2.0$ and $y_{\text{end}} = 2.0$. The initial condition is the situation of $t = 0$. The boundary conditions are the situation of $x = x_{\text{start}}$, $x = x_{\text{end}}$, $y = y_{\text{start}}$ and $y = y_{\text{end}}$. The vorticity is defined in terms of the stream function $\psi(t,x,y)$ by the Poisson equation $\omega = -(\psi_{xx} + \psi_{yy})$. The velocity components (u,v) can be recovered from the stream function as $u = \psi_y, \quad v = -\psi_x$. The two components of the velocity field data $u(t,x,y)$ and $v(t,x,y)$ are obtained from [Raissi *et al.*, 2019], where the numerical solution is performed. We approximate the stream function ψ using a neural network. The velocity field and vorticity can be derived by automatic differentiation.

We set $N_{\mathcal{B}} = 5000$ and $N_{\mathcal{L}} = 30 \times 20 \times 10$ in (7). For the inverse problem, μ is unknown and $N_{\mathcal{B}} + N_{\mathcal{L}} = 10 \times 10 \times 10$. In both cases, $N_{\text{test}} = 98 \times 48 \times 20$. Table 5 shows the averaged results over three independent runs. It can be seen that KP-PINNs, L_2-PINNs and RKHS-PINNs can solve the equation in general. Among them, KP-PINNs and RKHS-PINNs consistently demonstrate superior performance in terms of both accuracy and robustness. For the inverse problem, KP-PINNs($\nu = \frac{1}{2}$) can obtain the closest estimation.

6 Conclusion

This work proposes KP-PINNs, an efficient method for solving differential equations, with stability guarantees for second-order linear elliptic and HJB equations under certain conditions. Numerical experiments demonstrate that KP-PINNs outperform L_2-PINNs and Sobolev-PINNs for both forward and inverse problems, significantly accelerating the computation of the inverse of kernel matrix $\mathbf{K}$ through the KP technique. Future directions include extending KP-PINNs to handle sparse or irregular data, optimizing neural network architecture, and refining kernel selection in the loss function to improve accuracy and robustness. These advancements aim to enhance KP-PINNs' versatility in solving complex differential equations.

Acknowledgments

This research is supported by Guangdong Provincial Fund - Special Innovation Project (2024KTSCX038).

References

[Abramowitz and Stegun, 1968] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1968.

[Adams and Fournier, 2003] Robert A Adams and John JF Fournier. *Sobolev Spaces*. Academic Press, 2nd edition, 2003.

[Aliakbari, 2023] Maryam Aliakbari. *Physics informed neural networks to solve forward and inverse fluid flow and heat transfer problems*. PhD thesis, Northern Arizona University, 2023.

[Atkinson *et al.*, 2009] Kendall Atkinson, Weimin Han, and David E Stewart. *Numerical solution of ordinary differential equations*, volume 81. John Wiley & Sons, 2009.

[Bramble and Schatz, 1970] James H Bramble and Alfred H Schatz. Rayleigh-ritz-galerkin methods for dirichlet's problem using subspaces without boundary conditions. *Communications on Pure and Applied Mathematics*, 23(4):653–675, 1970.

[Byrd *et al.*, 1995] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[Chen *et al.*, 2022] Haoyuan Chen, Liang Ding, and Rui Tuo. Kernel packet: An exact and scalable algorithm for gaussian process regression with matérn correlations. *Journal of machine learning research*, 23(127):1–32, 2022.

[Davis, 2006] Timothy A Davis. *Direct methods for sparse linear systems*. SIAM, 2006.

[Deschamps, 1981] Georges A Deschamps. Electromagnetics and differential forms. *Proceedings of the IEEE*, 69(6):676–696, 1981.

[Ding *et al.*, 2019] Liang Ding, Simon Mak, and CF Wu. Bdrygp: a new gaussian process model for incorporating boundary information. *arXiv preprint arXiv:1908.08868*, 2019.

[Ding *et al.*, 2020] Liang Ding, Lu Zou, Wenjia Wang, Shahin Shahrampour, and Rui Tuo. High-dimensional non-parametric density estimation in mixed smooth sobolev spaces. *arXiv preprint arXiv:2006.03696*, 2020.

[Dũng and Nguyen, 2021] Dinh Dũng and Van Kien Nguyen. Deep relu neural networks in high-dimensional approximation. *Neural Networks*, 142:619–635, 2021.

[Evans and Souganidis, 1989] Lawrence C Evans and Panagiotis E Souganidis. A pde approach to geometric optics for certain semilinear parabolic equations. *Indiana University mathematics journal*, 38(1):141–172, 1989.

[Evans, 2022] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

[Forootani *et al.*, 2024] Ali Forootani, Harshit Kapadia, Sridhar Chellappa, Pawan Goyal, and Peter Benner. Gspinn: Greedy sampling for parameter estimation in partial differential equations. *arXiv preprint arXiv:2405.08537*, 2024.

[Ghergu and Radulescu, 2011] Marius Ghergu and Vicentiu Radulescu. *Nonlinear PDEs: Mathematical models in biology, chemistry and population genetics*. Springer Science & Business Media, 2011.

[Guo, 2022] Qing Guo. Optimal robust control of electrohydraulic system based on hamilton–jacobi–bellman solution with backstepping iteration. *IEEE Transactions on Control Systems Technology*, 31(1):459–466, 2022.

[Haitsiukevich and Ilin, 2022] Katsiaryna Haitsiukevich and Alexander Ilin. Improved training of physics-informed neural networks with model ensembles. *arXiv preprint arXiv:2204.05108*, 2022.

[Han *et al.*, 2018] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[Hochmuth *et al.*, 2000] Reinhard Hochmuth, Stephan Knapek, and Gerhard Zumbusch. *Tensor products of Sobolev spaces and applications*. Citeseer, 2000.

[Hu *et al.*, 2021] Zheyuan Hu, Ameya D Jagtap, George Em Karniadakis, and Kenji Kawaguchi. When do extended physics-informed neural networks (xpinns) improve generalization? *arXiv preprint arXiv:2109.09444*, 2021.

[Isachenko *et al.*, 1980] Viktor Pavlovich Isachenko, Varvara Aleksandrovna Osipova, and Aleksandr Semenovič Sukomel. *Heat transfer*. Nirali Prakashan, 1980.

[Jagtap *et al.*, 2020] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.

[Kaltenbacher, 2018] Manfred Kaltenbacher. *Computational acoustics*. Springer, 2018.

[Karniadakis *et al.*, 2021] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[Kianiharchegani, 2023] Elham Kianiharchegani. *Data-Driven Exploration of Coarse-Grained Equations: Harnessing Machine Learning*. PhD thesis, The University of Western Ontario (Canada), 2023.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kreyszig, 2007] Erwin Kreyszig. *Advanced Engineering Mathematics 9th Edition with Wiley Plus Set*, volume 334. John Wiley & Sons US, 2007.

[Kühn *et al.*, 2015] Thomas Kühn, Winfried Sickel, and Tino Ullrich. Approximation of mixed order sobolev functions on the d-torus: asymptotics, preasymptotics, and d-dependence. *Constructive Approximation*, 42:353–398, 2015.

[Li *et al.*, 2022] Ying Li, Longxiang Xu, and Shihui Ying. Dwnn: Deep wavelet neural network for solving partial differential equations. *Mathematics*, 10(12):1976, 2022.

[Lin *et al.*, 2022] Guang Lin, Yating Wang, and Zecheng Zhang. Multi-variance replica exchange sgmcmc for inverse and forward problems via bayesian pinn. *Journal of Computational Physics*, 460:111173, 2022.

[Muyskens *et al.*, 2024] Amanda Muyskens, Benjamin W Priest, Imene R Goumiri, and Michael D Schneider. Identifiability and sensitivity analysis of kriging weights for the matern kernel. *arXiv preprint arXiv:2410.08310*, 2024.

[Nguyen *et al.*, 2023] Van Giang Nguyen, Van Linh Nguyen, Sungho Jung, Hyunuk An, and Giha Lee. Exploring the power of physics-informed neural networks for accurate and efficient solutions to 1d shallow water equations. *Journal of Korea Water Resources Association*, 56(12):939–953, 2023.

[Ragheb, 1976] M Ragheb. Computational fluid dynamics. *mragheb Website, Available Online at https://mragheb.com/NPRE*, 20475, 1976.

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

[Shin *et al.*, 2023] Yeonjong Shin, Zhongqiang Zhang, and George Em Karniadakis. Error estimates of residual minimization using neural networks for linear pdes. *Journal of Machine Learning for Modeling and Computing*, 4(4), 2023.

[Sieniutycz, 2000] Stanislaw Sieniutycz. Hamilton–jacobi–bellman framework for optimal control in multistage energy systems. *Physics Reports*, 326(4):165–258, 2000.

[Son *et al.*, 2021] Hwijae Son, Jin Woo Jang, Woo Jin Han, and Hyung Ju Hwang. Sobolev training for physics informed neural networks. *arXiv preprint arXiv:2101.08932*, 2021.

[Wang and Jing, 2022] Wenjia Wang and Bing-Yi Jing. Gaussian process regression: Optimality, robustness, and relationship with kernel ridge regression. *Journal of Machine Learning Research*, 23(193):1–67, 2022.

[Wang *et al.*, 2020] Wenjia Wang, Rui Tuo, and CF Jeff Wu. On prediction properties of kriging: Uniform error bounds and robustness. *Journal of the American Statistical Association*, 115(530):920–930, 2020.

[Wang *et al.*, 2022] Chuwei Wang, Shanda Li, Di He, and Liwei Wang. Is L^2 physics-informed loss always suitable for training physics-informed neural network? *arXiv preprint arXiv:2206.02016*, 2022.

[Wanner and Hairer, 1996] Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg New York, 1996.

[Wendland, 2004] Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

[Witte and Reisinger, 2011] Jan Hendrik Witte and Christoph Reisinger. A penalty method for the numerical solution of hamilton–jacobi–bellman (hjb) equations in finance. *SIAM Journal on Numerical Analysis*, 49(1):213–231, 2011.

[Yan *et al.*, 2025] Lei Yan, Qiulei Wang, Gang Hu, Wenli Chen, and Bernd R Noack. Deep reinforcement cross-domain transfer learning of active flow control for three-dimensional bluff body flow. *Journal of Computational Physics*, 529:113893, 2025.

[Yong and Zhou, 2012] Jiongmin Yong and Xun Yu Zhou. *Stochastic controls: Hamiltonian systems and HJB equations*, volume 43. Springer Science & Business Media, 2012.

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

[Yuan *et al.*, 2022] Lei Yuan, Yi-Qing Ni, Xiang-Yun Deng, and Shuo Hao. A-pinn: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. *Journal of Computational Physics*, 462:111260, 2022.

[Zhang *et al.*, 2024] Weiwei Zhang, Wei Suo, Jiahao Song, and Wenbo Cao. Physics informed neural networks (pinns) as intelligent computing technique for solving partial differential equations: Limitation and future prospects. *arXiv preprint arXiv:2411.18240*, 2024.