Learning with physical constraints

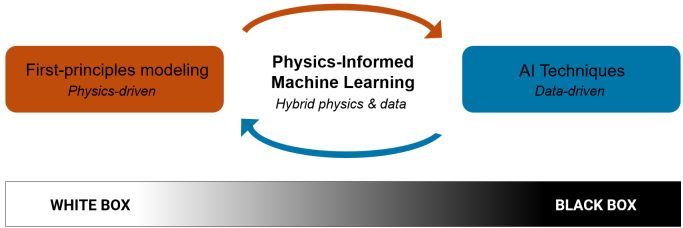
Gérard Biau
Sorbonne Université

January 2026

Team

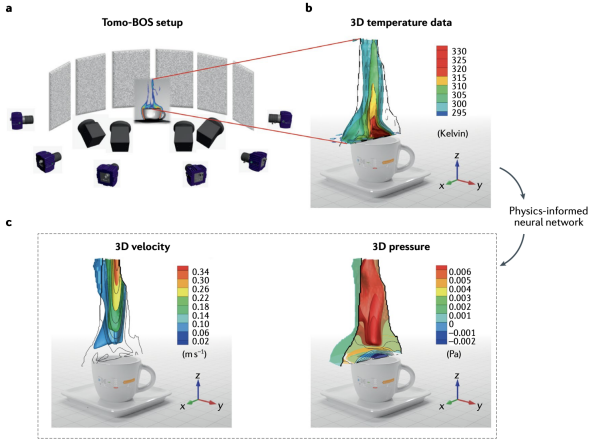
- **Francis Bach**, Inria Paris
- **Claire Boyer**, Université Paris-Saclay
- **Nathan Doumèche**, Sorbonne Université

Physics-informed machine learning



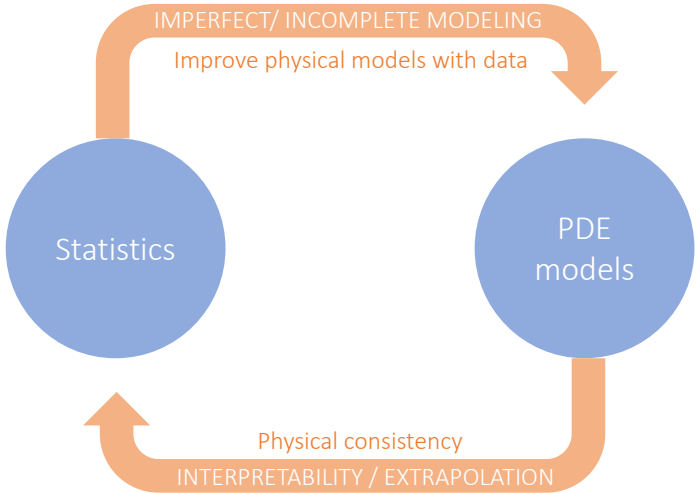
Source: What is physics-informed machine learning?

Hybrid modeling



Source: Karniadakis et al. (2021)

A mutual benefit



Agenda

1. **PDE constraints**
2. **Physics-informed kernel learner**
3. **PDE solving**
4. **Accelerating computation**

Agenda

1. **PDE constraints**
2. Physics-informed kernel learner
3. PDE solving
4. Accelerating computation

Mathematical context

- Model: $Y = f^\star(X) + \varepsilon$, with $\mathbb{E}(\varepsilon \mid X) = 0$.

Mathematical context

- Model: $Y = f^\star(X) + \varepsilon$, with $\mathbb{E}(\varepsilon \mid X) = 0$.
- Goal: estimate $f^\star$ using
  1. Supervised learning: an i.i.d. training sample $(X_i, Y_i)_{1 \leqslant i \leqslant n}$ on $\Omega \times \mathbb{R}$.

Mathematical context

- Model: $Y = f^\star(X) + \varepsilon$, with $\mathbb{E}(\varepsilon \mid X) = 0$.
- Goal: estimate $f^\star$ using
  1. Supervised learning: an i.i.d. training sample $(X_i, Y_i)_{1 \leqslant i \leqslant n}$ on $\Omega \times \mathbb{R}$.
  2. Physical modeling: a prior knowledge

$$\mathscr{D}(f^\star) \approx 0 \quad \text{on } \Omega,$$

  with a known differential operator $\mathscr{D}$.

Mathematical context

- Model: $Y = f^\star(X) + \varepsilon$, with $\mathbb{E}(\varepsilon \mid X) = 0$.
- Goal: estimate $f^\star$ using
  1. Supervised learning: an i.i.d. training sample $(X_i, Y_i)_{1 \leqslant i \leqslant n}$ on $\Omega \times \mathbb{R}$.
  2. Physical modeling: a prior knowledge

$$\mathscr{D}(f^\star) \approx 0 \quad \text{on } \Omega,$$

  with a known differential operator $\mathscr{D}$.
- Example: if $\partial^2_{t,t} f^\star(x,t) \approx \partial^2_{x,x} f^\star(x,t)$ (wave equation)

Mathematical context

- Model: $Y = f^\star(X) + \varepsilon$, with $\mathbb{E}(\varepsilon \mid X) = 0$.
- Goal: estimate $f^\star$ using
  1. Supervised learning: an i.i.d. training sample $(X_i, Y_i)_{1 \leqslant i \leqslant n}$ on $\Omega \times \mathbb{R}$.
  2. Physical modeling: a prior knowledge

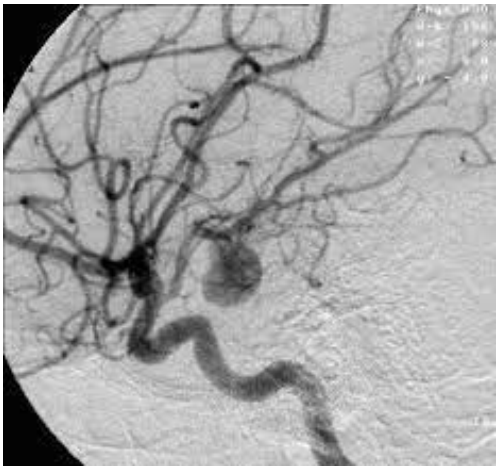
$$\mathscr{D}(f^\star) \approx 0 \quad \text{on } \Omega,$$

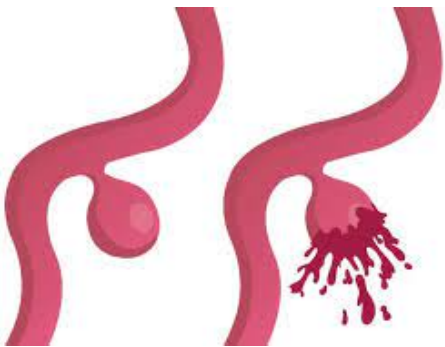
  with a known differential operator $\mathscr{D}$.

- Example: if $\partial^2_{t,t} f^\star(x,t) \approx \partial^2_{x,x} f^\star(x,t)$ (wave equation), then

$$\mathscr{D}(f)(x,t) = \partial^2_{t,t} f(x,t) - \partial^2_{x,x} f(x,t).$$

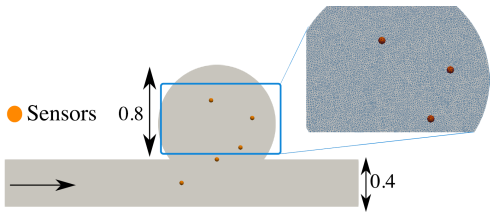
Blood flow in an aneurysm

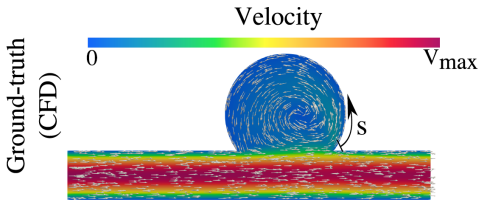




Modeling the blood flow

Arzani et al., 2021





Goal: estimate the blood flow $f = (f_1, f_2, P)$.

Navier-Stokes equations:

- $\mathscr{D}_1(f, \cdot) = f_1 \partial_x f_1 + f_2 \partial_y f_1 - \partial^2_{x,x} f_1 - \partial^2_{y,y} f_1 + \partial_x P$
- $\mathscr{D}_2(f, \cdot) = f_1 \partial_x f_2 + f_2 \partial_y f_2 - \partial^2_{x,x} f_2 - \partial^2_{y,y} f_2 + \partial_y P$
- $\mathscr{D}_3(f, \cdot) = \partial_x f_1 + \partial_y f_2$.

Physics-informed empirical risk

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.

Physics-informed empirical risk

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.
- Empirical risk: minimize

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2$$

over the class $\mathscr{F} \subseteq H^s(\Omega)$.

Physics-informed empirical risk

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.
- Empirical risk: minimize

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$$

over the class $\mathscr{F} \subseteq H^s(\Omega)$.

Physics-informed empirical risk

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.
- Empirical risk: minimize

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$$

  over the class $\mathscr{F} \subseteq H^s(\Omega)$.

- Minimizer:

$$\hat{f}_n = \operatorname*{argmin}_{f \in \mathscr{F}} R_n(f).$$

Physics-informed empirical risk

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.
- Empirical risk: minimize

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$$

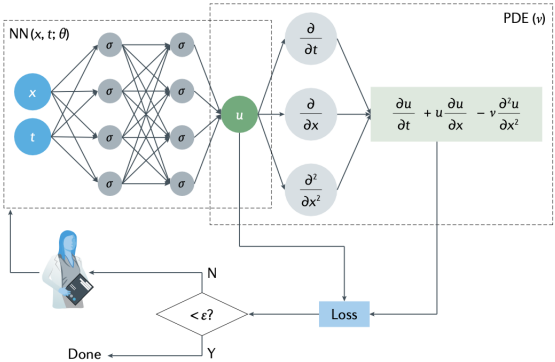
  over the class $\mathscr{F} \subseteq H^s(\Omega)$.

- Minimizer:

$$\hat{f}_n = \underset{f \in \mathscr{F}}{\operatorname{argmin}}\, R_n(f).$$

☞ Physics-Informed Neural Networks.

PINNs



Source: Karniadakis et al. (2021)

☞ Strong convergence

\+ statistical accuracy on the support of the data

\+ physical consistency anywhere else.

Bernoulli **31**(3), 2025, 2127–2151
https://doi.org/10.3150/24-BEJ1799

On the convergence of PINNs

NATHAN DOUMÈCHE[1,a], GÉRARD BIAU[1,2,b] and CLAIRE BOYER[1,2,c]

[1]*Sorbonne Université, CNRS, LPSM, F-75005 Paris, France,* [a]*nathan.doumeche@sorbonne-universite.fr,*
[b]*gerard.biau@sorbonne-universite.fr,* [c]*claire.boyer@sorbonne-universite.fr*
[2]*Institut universitaire de France (IUF), Paris, France*

Physics-informed neural networks (PINNs) are a promising approach that combines the power of neural networks with the interpretability of physical modeling. PINNs have shown good practical performance in solving partial differential equations (PDEs) and in hybrid modeling scenarios, where physical models enhance data-driven approaches. However, it is essential to establish their theoretical properties in order to fully understand their capabilities and limitations. In this study, we highlight that classical training of PINNs can suffer from systematic overfitting. This problem can be addressed by adding a ridge regularization to the empirical risk, which ensures that the resulting estimator is risk-consistent for both linear and nonlinear PDE systems. However, the strong convergence of PINNs to a solution satisfying the physical constraints requires a more involved analysis using tools from functional analysis and calculus of variations. In particular, for linear PDE systems, an implementable Sobolev-type regularization allows to reconstruct a solution that not only achieves statistical accuracy but also maintains consistency with the underlying physics.

Keywords: Consistency; hybrid modeling; PDE solver; physics-informed neural networks

1. Introduction

Physics-informed machine learning Advances in machine learning and deep learning have led to significant breakthroughs in almost all areas of science and technology. However, despite remarkable achievements, modern machine learning models are difficult to interpret and do not necessarily obey the fundamental governing laws of physical systems (Linardatos, Papastefanopoulos and Kotsiantis, 2021). Moreover, they often fail to extrapolate scenarios beyond those on which they were trained (Xu et al., 2021). On the contrary, numerical or pure physical methods struggle to capture nonlinear relationships in complex and high-dimensional systems, while lacking flexibility and being prone to computational problems. This state of affairs has led to a growing consensus that data-driven machine learning methods need to be coupled with prior scientific knowledge based on physics. This emerging field, often called physics-informed machine learning (Raissi, Perdikaris and Karniadakis, 2019), seeks to combine the predictive power of machine learning techniques with the interpretability and robustness of physical modeling. The literature in this field is still disorganized, with a somewhat unstable nomenclature. In particular, the terms physics-informed, physics-based, physics-guided, and theory-guided are used interchangeably. For a comprehensive account, we refer to the reviews by Rai and Sahu (2020), Karniadakis et al. (2021), Cuomo et al. (2022), and Hao et al. (2022), which survey some of the prevailing trends in embedding physical knowledge in machine learning, present some of the current challenges, and discuss various applications.

Vocabulary and use cases Depending on the nature of the interaction between machine learning and physics, physics-informed machine learning is usually achieved by preprocessing the features (Rai and Sahu, 2020), by designing innovative network architectures that incorporate the physics of the problem (Karniadakis et al., 2021), or by forcing physics infusion into the loss function (Cuomo et al., 2022). It is this latter approach, which is most often referred to as physics regularization (Rai and Sahu, 2020), to which our article is devoted. Note that other names are possible, including physics consistency penalty

1350-7265 © 2025 ISI/BS

arXiv:2305.01240

Agenda

1. PDE constraints
2. **Physics-informed kernel learner**
3. PDE solving
4. Accelerating computation

Leveraging Fourier analysis

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.

Leveraging Fourier analysis

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.
- Sobolev norm:

$$\|f\|_{H^s(\Omega)} = \Big(\sum_{|\alpha| \leqslant s} \|\partial^\alpha f\|^2_{L^2(\Omega)} \Big)^{1/2}.$$

Leveraging Fourier analysis

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.
- Sobolev norm:

$$\|f\|_{H^s(\Omega)} = \Big(\sum_{|\alpha| \leqslant s} \|\partial^\alpha f\|^2_{L^2(\Omega)} \Big)^{1/2}.$$

- Fourier series representation:

$$f(x) = \sum_{k \in \mathbb{Z}^d} z_k \, e^{-\frac{i\pi}{2L}\langle k, x \rangle}.$$

Leveraging Fourier analysis

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.
- Sobolev norm:

$$\|f\|_{H^s(\Omega)} = \Big(\sum_{|\alpha| \leqslant s} \|\partial^\alpha f\|^2_{L^2(\Omega)} \Big)^{1/2}.$$

- Fourier series representation:

$$f(x) = \sum_{k \in \mathbb{Z}^d} z_k \, e^{-\frac{i\pi}{2L}\langle k, x \rangle}.$$

- Equivalent norms:

$$\|f\|^2_{H^s(\Omega)} \leqslant \sum_{k \in \mathbb{Z}^d} |z_k|^2 \left(1 + \left(\frac{\|k\|_2^2}{(2L)^d}\right)^s\right) \leqslant C_\Omega \|f\|^2_{H^s(\Omega)}.$$

Leveraging Fourier analysis

- Assumptions: $\Omega \subseteq [-L, L]^d$ and $f^\star \in H^s(\Omega)$, $s > \frac{d}{2}$.
- Sobolev norm:

$$\|f\|_{H^s(\Omega)} = \Big(\sum_{|\alpha| \leqslant s} \|\partial^\alpha f\|^2_{L^2(\Omega)} \Big)^{1/2}.$$

- Fourier series representation:

$$f(x) = \sum_{k \in \mathbb{Z}^d} z_k \, e^{-\frac{i\pi}{2L}\langle k, x \rangle}.$$

- Equivalent norms:

$$\|f\|^2_{H^s(\Omega)} \leqslant \sum_{k \in \mathbb{Z}^d} |z_k|^2 \left(1 + \left(\frac{\|k\|_2^2}{(2L)^d}\right)^s\right) \leqslant C_\Omega \|f\|^2_{H^s(\Omega)}.$$

✓ Advantage: Fourier analysis.

Moving towards a kernel approach

- Empirical risk: minimize

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$$

over the class $\mathscr{F} \subseteq H^s(\Omega)$.

Moving towards a kernel approach

- Empirical risk: minimize

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \|f\|^2_{\mathrm{RKHS}}$$

with $\|f\|^2_{\mathrm{RKHS}} = \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$.

Moving towards a kernel approach

- Empirical risk: minimize

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \|f\|^2_{\mathrm{RKHS}}$$

with $\|f\|^2_{\mathrm{RKHS}} = \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$.

- Questions:

1. How does the PDE penalty influence learning?
2. How can we leverage the kernel toolbox?
3. How do we design a tractable estimator?

Moving towards a kernel approach

- Empirical risk: minimize

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \|f\|^2_{\mathrm{RKHS}}$$

with $\|f\|^2_{\mathrm{RKHS}} = \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$.

- Questions:
  1. How does the PDE penalty influence learning?
  2. How can we leverage the kernel toolbox?
  3. How do we design a tractable estimator?
- Linear assumption: $\mathscr{D}(\alpha f + g) = \alpha \mathscr{D}(f) + \mathscr{D}(g)$.

Proceedings of Machine Learning Research vol 196:1–52, 2024 37th Annual Conference on Learning Theory

Physics-informed machine learning as a kernel method

Nathan Doumèche NATHAN.DOUMECHE@SORBONNE-UNIVERSITE.FR
Sorbonne Université and Université Paris Cité, CNRS, Laboratoire de Probabilités, Statistique et Modélisation, F-75005 Paris, France

Francis Bach FRANCIS.BACH@INRIA.FR
INRIA Paris, France

Gérard Biau GERARD.BIAU@SORBONNE-UNIVERSITE.FR
Sorbonne Université and Université Paris Cité, CNRS, Laboratoire de Probabilités, Statistique et Modélisation, F-75005 Paris, France

Claire Boyer CLAIRE.BOYER@SORBONNE-UNIVERSITE.FR
Sorbonne Université and Université Paris Cité, CNRS, Laboratoire de Probabilités, Statistique et Modélisation, F-75005 Paris, France
Institut universitaire de France (IUF)

Editors: Shipra Agrawal and Aaron Roth

Abstract

Physics-informed machine learning combines the expressiveness of data-based approaches with the interpretability of physical models. In this context, we consider a general regression problem where the empirical risk is regularized by a partial differential equation that quantifies the physical inconsistency. We prove that for linear differential priors, the problem can be formulated as a kernel regression task. Taking advantage of kernel theory, we derive convergence rates for the minimizer $\hat{f}_n$ of the regularized risk and show that $\hat{f}_n$ converges at least at the Sobolev minimax rate. However, faster rates can be achieved, depending on the physical error. This principle is illustrated with a one-dimensional example, supporting the claim that regularizing the empirical risk with physical information can be beneficial to the statistical performance of estimators.

Keywords: Physics-informed machine learning, Kernel methods, Rates of convergence, Physical regularization

1. Introduction

Physics-informed machine learning. Physics-informed machine learning (PIML) refers to a subdomain of machine learning that combines physical knowledge and empirical data to enhance performance of tasks involving a physical mechanism. Following the influential work of Raissi et al. (2019), the field has experienced a notable surge in popularity, largely driven by scientific computing and engineering applications. We refer the reader to the surveys by Rai and Sahu (2020), Karniadakis et al. (2021), Cuomo et al. (2022), and Hao et al. (2022). In a nutshell, the success of PIML relies on the smart interaction between machine learning and physics. In its most standard form, this achievement is realized by integrating physical equations into the loss function. Three common use cases include solving systems of partial differential equations (PDEs), addressing inverse problems (e.g., learning the PDE governing an observed phenomenon), and further improving the statistical performance of empirical risk minimization. This article focuses on the latter approach, known as hybrid modeling (e.g., Rai and Sahu, 2020).

© 2024 N. Doumèche, F. Bach, G. Biau & C. Boyer.

arXiv:2402.07514

Journal of Machine Learning Research 26 (2025) 1-39 Submitted 9/24; Revised 3/25; Published 8/25

Physics-informed Kernel Learning

Nathan Doumèche NATHAN.DOUMECHE@SORBONNE-UNIVERSITE.FR
Sorbonne University, EDF R&D

Francis Bach FRANCIS.BACH@INRIA.FR
INRIA, École Normale Supérieure, PSL University

Gérard Biau GERARD.BIAU@SORBONNE-UNIVERSITE.FR
Sorbonne University, Institut universitaire de France

Claire Boyer CLAIRE.BOYER@UNIVERSITE-PARIS-SACLAY.FR
Université Paris-Saclay, Institut universitaire de France

Editor: Animashree Anandkumar

Abstract

Physics-informed machine learning typically integrates physical priors into the learning process by minimizing a loss function that includes both a data-driven term and a partial differential equation (PDE) regularization. Building on the formulation of the problem as a kernel regression task, we use Fourier methods to approximate the associated kernel, and propose a tractable estimator that minimizes the physics-informed risk function. We refer to this approach as physics-informed kernel learning (PIKL). This framework provides theoretical guarantees, enabling the quantification of the physical prior's impact on convergence speed. We demonstrate the numerical performance of the PIKL estimator through simulations, both in the context of hybrid modeling and in solving PDEs. In particular, we show that PIKL can outperform physics-informed neural networks in terms of both accuracy and computation time. Additionally, we identify cases where PIKL surpasses traditional PDE solvers, particularly in scenarios with noisy boundary conditions.

Keywords: Physics-informed machine learning, Kernel methods, Physics-informed neural networks, Rates of convergence, Physical regularization

1. Introduction

Physics-informed machine learning. Physics-informed machine learning (PIML), as described by Raissi et al. (2019), is a promising framework that combines statistical and physical principles to leverage the strengths of both fields. PIML can be applied to a variety of problems, such as solving partial differential equations (PDEs) using machine learning techniques, leveraging PDEs to accelerate the learning of unknown functions (hybrid modeling), and learning PDEs directly from data (inverse problems). For an introduction to the field and a literature review, we refer to Karniadakis et al. (2021) and Cuomo et al. (2022).

Hybrid modeling setting. We consider in this paper the classical regression model, which aims at learning the unknown function $f^\star : \mathbb{R}^d \to \mathbb{R}$ such that $Y = f^\star(X) + \varepsilon$, where $Y \in \mathbb{R}$ is the output, $X \in \Omega$ are the features with $\Omega \subset [-L, L]^d$ the input domain, and ε is a random noise. Using n observations $(X_1, Y_1), \dots, (X_n, Y_n)$, independent copies of (X, Y), the goal is to construct an estimator $\hat{f}_n$ of $f^\star$. What makes PIML special compared to other regression settings is the prior knowledge that $f^\star$ approximately follows a PDE. Therefore, we assume that $f^\star$ is

©2025 Nathan Doumèche, Francis Bach, Gérard Biau, and Claire Boyer.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v26/24-1536.html.

arXiv:2409.13786

Kernel methods in a nutshell

- Kernel: a symmetric pd. function $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$

Kernel methods in a nutshell

- Kernel: a symmetric pd. function $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R} \quad \to \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \geqslant 0.$

Kernel methods in a nutshell

- Kernel: a symmetric pd. function $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R} \quad \to \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \geqslant 0.$
- There exists a RKHS of functions $f : \mathscr{X} \to \mathbb{R}$ such that
  1. $\forall x \in \mathscr{X}, K(\cdot, x) \in \mathscr{H}$.
  2. $\forall f \in \mathrm{RKHS}, \langle f, K(\cdot, x) \rangle_{\mathrm{RKHS}} = f(x)$.

Kernel methods in a nutshell

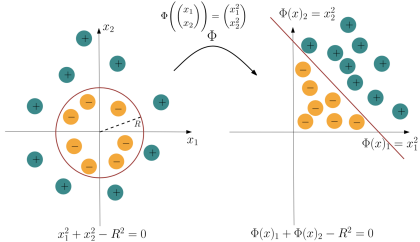
- Kernel: a symmetric pd. function $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R} \quad \to \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \geqslant 0.$
- There exists a RKHS of functions $f : \mathscr{X} \to \mathbb{R}$ such that
  1. $\forall x \in \mathscr{X}, K(\cdot, x) \in \mathscr{H}$.
  2. $\forall f \in \mathrm{RKHS}, \langle f, K(\cdot, x) \rangle_{\mathrm{RKHS}} = f(x)$.
- Important example: $K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathscr{T}}$

Kernel methods in a nutshell

- Kernel: a symmetric pd. function $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R} \quad \to \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \geqslant 0.$
- There exists a RKHS of functions $f : \mathscr{X} \to \mathbb{R}$ such that
  1. $\forall x \in \mathscr{X}, K(\cdot, x) \in \mathscr{H}$.
  2. $\forall f \in \mathrm{RKHS}, \langle f, K(\cdot, x) \rangle_{\mathrm{RKHS}} = f(x)$.
- Important example: $K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathscr{F}} = \left\langle \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}, \begin{pmatrix} y_1^2 \\ y_2^2 \end{pmatrix} \right\rangle_{\mathbb{R}^2}.$

Kernel methods in a nutshell

- Kernel: a symmetric pd. function $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R} \quad \to \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \geqslant 0.$
- There exists a RKHS of functions $f : \mathscr{X} \to \mathbb{R}$ such that
  1. $\forall x \in \mathscr{X}, K(\cdot, x) \in \mathscr{H}$.
  2. $\forall f \in \mathrm{RKHS}, \langle f, K(\cdot, x) \rangle_{\mathrm{RKHS}} = f(x)$.
- Important example: $K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathscr{T}} = \left\langle \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}, \begin{pmatrix} y_1^2 \\ y_2^2 \end{pmatrix} \right\rangle_{\mathbb{R}^2}$.



Kernel methods in a nutshell

- Regularized empirical risk minimization:

$$\hat{f}_n = \underset{f \in \mathrm{RKHS}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}}.$$

Kernel methods in a nutshell

- Regularized empirical risk minimization:

$$\hat{f}_n = \underset{f \in \mathrm{RKHS}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}}.$$

- Solution:

$$\hat{f}_n(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x, X_i), \quad \text{with } \hat{\alpha} = (\mathbb{K} + n\lambda_n \mathrm{I}_n)^{-1} \mathbb{Y}.$$

Kernel methods in a nutshell

- Regularized empirical risk minimization:

$$\hat{f}_n = \underset{f \in \mathrm{RKHS}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}}.$$

- Solution:

$$\hat{f}_n(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x, X_i), \quad \text{with } \hat{\alpha} = (\mathbb{K} + n\lambda_n \mathrm{I}_n)^{-1} \mathbb{Y}.$$

- Integral operator: $L_K f(x) = \int_{\mathscr{X}} K(x, y) f(y) d\mathbb{P}_X(y)$.

Kernel methods in a nutshell

- Regularized empirical risk minimization:

$$\hat{f}_n = \underset{f \in \mathrm{RKHS}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}}.$$

- Solution:

$$\hat{f}_n(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x, X_i), \quad \text{with } \hat{\alpha} = (\mathbb{K} + n\lambda_n \mathrm{I}_n)^{-1} \mathbb{Y}.$$

- Integral operator: $L_K f(x) = \int_{\mathscr{X}} K(x, y) f(y) d\mathbb{P}_X(y)$.

- Effective dimension: $\mathrm{tr}(L_K(\mathrm{Id} + L_K)^{-1})$.

The RKHS quest

Goal: find a RKHS such that $\|f\|^2_{\mathrm{RKHS}} = \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$.

The RKHS quest

Goal: find a RKHS such that $\|f\|^2_{\mathrm{RKHS}} = \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$.

Theorem

The space $H^s(\Omega)$, equipped with the inner product

$$\langle f, g \rangle_{\mathrm{RKHS}} = \langle \mathscr{O}_n^{-1/2} f, \mathscr{O}_n^{-1/2} g \rangle_{L^2([-2L,2L]^d)},$$

is a RKHS.

The RKHS quest

Goal: find a RKHS such that $\|f\|^2_{\mathrm{RKHS}} = \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$.

Theorem

The space $H^s(\Omega)$, equipped with the inner product

$$\langle f, g \rangle_{\mathrm{RKHS}} = \langle \mathscr{O}_n^{-1/2} f, \mathscr{O}_n^{-1/2} g \rangle_{L^2([-2L,2L]^d)},$$

is a RKHS. For $f \in H^s(\Omega)$,

$$\|f\|^2_{\mathrm{RKHS}} = \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)},$$

The RKHS quest

Goal: find a RKHS such that $\|f\|^2_{\mathrm{RKHS}} = \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$.

Theorem

The space $H^s(\Omega)$, equipped with the inner product

$$\langle f, g \rangle_{\mathrm{RKHS}} = \langle \mathscr{O}_n^{-1/2} f, \mathscr{O}_n^{-1/2} g \rangle_{L^2([-2L,2L]^d)},$$

is a RKHS. For $f \in H^s(\Omega)$,

$$\|f\|^2_{\mathrm{RKHS}} = \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)},$$

and the associated kernel is $K(x,y) = \sum_{m \in \mathbb{N}} a_m v_m(x) v_m(y)$, $a_m > 0$.

PIML as a kernel method

- Conclusion:
  - ✓ Kernel method: $\hat{f}_n = \operatorname{argmin}_{f \in H^s(\Omega)} \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \|f\|^2_{\mathrm{RKHS}}$.
  - ✗ Computing the kernel is not straightforward.

PIML as a kernel method

- Conclusion:
  - ✓ Kernel method: $\hat{f}_n = \operatorname{argmin}_{f \in H^s(\Omega)} \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \|f\|^2_{\mathrm{RKHS}}$.
  - ✗ Computing the kernel is not straightforward.

Theorem

Assumptions: $\frac{d\mathbb{P}_X}{dx} \leqslant \kappa$ and ε is (M, σ)-sub-Gamma.

PIML as a kernel method

- Conclusion:
  - ✓ Kernel method: $\hat{f}_n = \operatorname{argmin}_{f \in H^s(\Omega)} \frac{1}{n} \sum_{i=1}^n |f(X_i) - Y_i|^2 + \|f\|^2_{\mathrm{RKHS}}$.
  - ✗ Computing the kernel is not straightforward.

Theorem

Assumptions: $\frac{d\mathbb{P}_X}{dx} \leqslant \kappa$ and ε is (M, σ)-sub-Gamma. Then

$$\mathbb{E} \int_\Omega |\hat{f}_n - f^\star|^2 d\mathbb{P}_X \lesssim \log^2(n) \Big(\lambda_n \|f^\star\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f^\star)\|^2_{L^2(\Omega)} + \frac{M^2}{n^2 \lambda_n} + \frac{\sigma^2 \mathscr{N}(\lambda_n, \mu_n)}{n} \Big).$$

PIML as a kernel method

- Conclusion:
  - ✓ Kernel method: $\hat{f}_n = \operatorname{argmin}_{f \in H^s(\Omega)} \frac{1}{n} \sum_{i=1}^n |f(X_i) - Y_i|^2 + \|f\|^2_{\mathrm{RKHS}}$.
  - ✗ Computing the kernel is not straightforward.

Theorem

Assumptions: $\frac{d\mathbb{P}_X}{dx} \leqslant \kappa$ and ε is (M, σ)-sub-Gamma. Then

$$\mathbb{E} \int_\Omega |\hat{f}_n - f^\star|^2 d\mathbb{P}_X \lesssim \log^2(n) \Big(\lambda_n \|f^\star\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f^\star)\|^2_{L^2(\Omega)} + \frac{M^2}{n^2 \lambda_n} + \frac{\sigma^2 \mathscr{N}(\lambda_n, \mu_n)}{n} \Big).$$

- Simple bound:

$$\mathbb{E} \int_\Omega |\hat{f}_n - f^\star|^2 d\mathbb{P}_X = \mathcal{O}_n\big(n^{-2s/(2s+d)} \log^3(n) \big).$$

PIML as a kernel method

- Conclusion:
  - ✓ Kernel method: $\hat{f}_n = \operatorname{argmin}_{f \in H^s(\Omega)} \frac{1}{n} \sum_{i=1}^n |f(X_i) - Y_i|^2 + \|f\|^2_{\mathrm{RKHS}}$.
  - ✗ Computing the kernel is not straightforward.

Theorem

Assumptions: $\frac{d\mathbb{P}_X}{dx} \leqslant \kappa$ and ε is (M, σ)-sub-Gamma. Then

$$\mathbb{E} \int_\Omega |\hat{f}_n - f^\star|^2 d\mathbb{P}_X \lesssim \log^2(n) \Big(\lambda_n \|f^\star\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f^\star)\|^2_{L^2(\Omega)} + \frac{M^2}{n^2 \lambda_n} + \frac{\sigma^2 \mathscr{N}(\lambda_n, \mu_n)}{n} \Big).$$

- Simple bound:

$$\mathbb{E} \int_\Omega |\hat{f}_n - f^\star|^2 d\mathbb{P}_X = \mathcal{O}_n\big(n^{-2s/(2s+d)} \log^3(n) \big).$$

- Question: can we do better?

An example

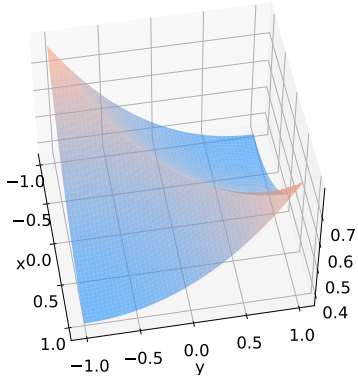
- $d = 1$, $\Omega = [-L, L]$, $f^\star \in H^1(\Omega)$, $\mathscr{D} = \frac{d}{dx}$.

An example

- $d = 1$, $\Omega = [-L, L]$, $f^\star \in H^1(\Omega)$, $\mathscr{D} = \frac{d}{dx}$.
- $\hat{f}_n = \underset{f \in H^1(\Omega)}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{H^1(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$.

An example

- $d = 1$, $\Omega = [-L, L]$, $f^\star \in H^1(\Omega)$, $\mathscr{D} = \frac{d}{dx}$.
- $\hat{f}_n = \underset{f \in H^1(\Omega)}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{H^1(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}$.



K with $\lambda_n = \mu_n = 1$.

Impact of physics

Theorem

Assumptions: $\lambda_n = n^{-1}\log(n)$ and

$$\mu_n = \begin{cases} n^{-2/3}/\|\mathscr{D}(f^\star)\|_{L^2(\Omega)} & \text{if} \quad \|\mathscr{D}(f^\star)\|_{L^2(\Omega)} \neq 0 \\ 1/\log(n) & \text{if} \quad \|\mathscr{D}(f^\star)\|_{L^2(\Omega)} = 0. \end{cases}$$

Impact of physics

Theorem

Assumptions: $\lambda_n = n^{-1}\log(n)$ and

$$\mu_n = \begin{cases} n^{-2/3}/\|\mathscr{D}(f^\star)\|_{L^2(\Omega)} & \text{if} \quad \|\mathscr{D}(f^\star)\|_{L^2(\Omega)} \neq 0 \\ 1/\log(n) & \text{if} \quad \|\mathscr{D}(f^\star)\|_{L^2(\Omega)} = 0. \end{cases}$$

Then

$$\mathbb{E}\int_{[-L,L]} |\hat{f}_n - f^\star|^2 d\mathbb{P}_X = \|\mathscr{D}(f^\star)\|_{L^2(\Omega)}\, \mathcal{O}_n\big(n^{-2/3}\log^3(n)\big) + \big(\|f^\star\|^2_{H^s(\Omega)} + \sigma^2 + M^2\big)\, \mathcal{O}_n\big(n^{-1}\log^3(n)\big).$$

In practice

- Kernel estimator: $\hat{f}_n(x) = (K(x, X_1), \dots, K(x, X_n))(\mathbb{K} + n\mathrm{I}_n)^{-1}\mathbb{Y}$.

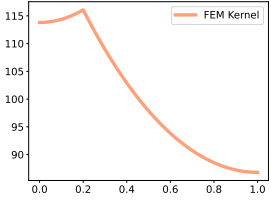
In practice

- Kernel estimator: $\hat{f}_n(x) = (K(x, X_1), \dots, K(x, X_n))(\mathbb{K} + n\mathrm{I}_n)^{-1}\mathbb{Y}$.
- Problem: what if K is not explicit?

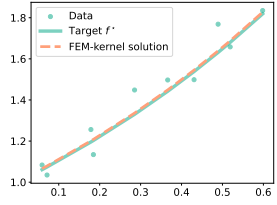
In practice

- Kernel estimator: $\hat{f}_n(x) = (K(x, X_1), \dots, K(x, X_n))(\mathbb{K} + n\mathrm{I}_n)^{-1}\mathbb{Y}$.
- Problem: what if K is not explicit?
- Solution 1: finite element method for a weak PDE

$$\lambda_n \int_\Omega \Big[K(x,\cdot)\,\varphi + \sum_{|\alpha|=s} \partial^\alpha K(x,\cdot)\,\partial^\alpha \varphi\Big] + \mu_n \int_\Omega \mathscr{D}(K(x,\cdot))\,\mathscr{D}(\varphi) = \varphi(x).$$



Kernel function $K(0.4, \cdot)$



Kernel estimator $\hat{f}_n$

Example with $\mathscr{D}(f) = \frac{d}{dx}f - f$.

In practice

- Kernel estimator: $\hat{f}_n(x) = (K(x, X_1), \dots, K(x, X_n))(\mathbb{K} + n\mathrm{I}_n)^{-1}\mathbb{Y}$.
- Problem: what if K is not explicit?
- Solution 2: Fourier analysis.

In practice

- Kernel estimator: $\hat{f}_n(x) = (K(x, X_1), \dots, K(x, X_n))(\mathbb{K} + n\mathrm{I}_n)^{-1}\mathbb{Y}$.
- Problem: what if K is not explicit?
- Solution 2: Fourier analysis.
  1. Goal:
  $$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{H^s(\Omega)} + \mu_n \|\mathscr{D}(f)\|^2_{L^2(\Omega)}.$$
  2. Restrict the minimization to
  $$H_m = \mathrm{Span}((\phi_k)_{\|k\|_\infty \leqslant m}), \quad \text{with} \quad \phi_k(x) = (4L)^{-d/2} e^{\frac{i\pi}{2L}\langle k, x\rangle}.$$
  3. Physics-Informed Kernel Learner:
  $$\hat{f}^{\mathrm{PIKL}} = \underset{f \in H_m}{\mathrm{argmin}}\, R_n(f).$$

Computing the PIKL

- Kernel:

$$K_m(x, y) = \langle M_m^{-1/2} \Phi_m(x), M_m^{-1/2} \Phi_m(y) \rangle_{\mathbb{C}^{(2m+1)^d}},$$

Computing the PIKL

- Kernel:

$$K_m(x,y) = \langle M_m^{-1/2}\Phi_m(x), M_m^{-1/2}\Phi_m(y)\rangle_{\mathbb{C}^{(2m+1)^d}},$$

where

$$(M_m)_{j,k} = \lambda_n \underbrace{\left(1 + \left(\frac{\|k\|_2^2}{(2L)^d}\right)^s\right)\delta_{j,k}}_{\text{Sobolev norm}} + \mu_n \underbrace{\frac{P(j)\bar{P}(k)}{(4L)^d}\int_\Omega e^{\frac{i\pi}{2L}\langle k-j,x\rangle}dx}_{\text{PDE norm}}$$

and $P(k) = \sum_{|\alpha|\leqslant s} a_\alpha (\frac{-i\pi}{2L})^{|\alpha|} \prod_{\ell=1}^d (k_\ell)^{\alpha_\ell}$.

Computing the PIKL

- Kernel:

$$K_m(x,y) = \langle M_m^{-1/2}\Phi_m(x), M_m^{-1/2}\Phi_m(y)\rangle_{\mathbb{C}^{(2m+1)^d}},$$

where

$$(M_m)_{j,k} = \lambda_n \underbrace{\left(1 + \left(\frac{\|k\|_2^2}{(2L)^d}\right)^s\right)\delta_{j,k}}_{\text{Sobolev norm}} + \mu_n \underbrace{\frac{P(j)\bar{P}(k)}{(4L)^d}\int_\Omega e^{\frac{i\pi}{2L}\langle k-j,x\rangle}dx}_{\text{PDE norm}}$$

and $P(k) = \sum_{|\alpha|\leqslant s} a_\alpha (\frac{-i\pi}{2L})^{|\alpha|} \prod_{\ell=1}^d (k_\ell)^{\alpha_\ell}$.

- Numerical integration or closed-form formulas:
  - ✓ When $\Omega = [-L, L]^d$, integral $\propto \prod_{j=1}^d \frac{\sin(\pi k_j/2)}{\pi k_j}$.
  - ✓ When $\Omega = B_2^2$, integral $\propto \frac{\text{Bessel function}_1(\pi\|k\|_2/2)}{4\|k\|_2}$.

Computing the PIKL

Physics-Informed Kernel Learner

$$\hat{f}^{\mathrm{PIKL}}(x) = (K_m(x, X_1), \ldots, K_m(x, X_n))(\mathbb{K}_m + n\mathrm{I}_n)^{-1}\mathbb{Y}$$

Computing the PIKL

Physics-Informed Kernel Learner

$$\begin{aligned}\hat{f}^{\mathrm{PIKL}}(x) &= (K_m(x, X_1), \ldots, K_m(x, X_n))(\mathbb{K}_m + n\mathrm{I}_n)^{-1}\mathbb{Y} \\ &= \Phi_m(x)^* \underbrace{(\Phi^*\Phi + nM_m)^{-1}}_{(2m+1)^d \times (2m+1)^d} \Phi^*\mathbb{Y}.\end{aligned}$$

Fourier-basis matrix:

$$\Phi = \begin{pmatrix} \Phi_m(X_1)^* \\ \vdots \\ \Phi_m(X_n)^* \end{pmatrix} \in \mathbb{C}^{n \times (2m+1)^d}.$$

Computing the PIKL

Physics-Informed Kernel Learner

$$\begin{aligned}\hat{f}^{\mathrm{PIKL}}(x) &= (K_m(x, X_1), \dots, K_m(x, X_n))(\mathbb{K}_m + n\mathrm{I}_n)^{-1}\mathbb{Y} \\ &= \Phi_m(x)^* \underbrace{(\Phi^*\Phi + nM_m)^{-1}}_{(2m+1)^d \times (2m+1)^d} \Phi^*\mathbb{Y}.\end{aligned}$$

Fourier-basis matrix:

$$\Phi = \begin{pmatrix} \Phi_m(X_1)^* \\ \vdots \\ \Phi_m(X_n)^* \end{pmatrix} \in \mathbb{C}^{n \times (2m+1)^d}.$$

- Complexity/storage: $(2m+1)^d \times (2m+1)^d$ vs. $n \times n$.
- Possible computation of $\Phi^*\Phi$ and $\Phi^*\mathbb{Y}$ online and in parallel.

Experience 1: Perfect modeling

- Harmonic oscillator: $d = 1$, $\Omega = [-\pi, \pi]$, $\mathscr{D}(f) = \frac{d^2 f}{dx^2} + \frac{df}{dx} + f$.

Experience 1: Perfect modeling

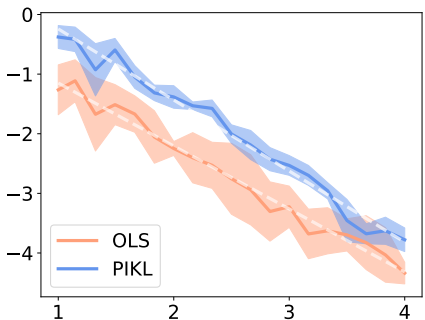
- Harmonic oscillator: $d = 1$, $\Omega = [-\pi, \pi]$, $\mathscr{D}(f) = \frac{d^2 f}{dx^2} + \frac{df}{dx} + f$.
- PDE solutions: $f = a_1 f_1 + a_2 f_2$, where $(a_1, a_2) \in \mathbb{R}^2$.

☞ Comparison with OLS via $(\hat{a}_1, \hat{a}_2)$.

Experience 1: Perfect modeling

- Harmonic oscillator: $d = 1$, $\Omega = [-\pi, \pi]$, $\mathscr{D}(f) = \frac{d^2 f}{dx^2} + \frac{df}{dx} + f$.
- PDE solutions: $f = a_1 f_1 + a_2 f_2$, where $(a_1, a_2) \in \mathbb{R}^2$.

☞ Comparison with OLS via $(\hat{a}_1, \hat{a}_2)$.



- Expected parametric rate of n^{-1}.
- PIKL ($m = 300$) is just as effective as the OLS estimator.

Experience 2: Imperfect modeling

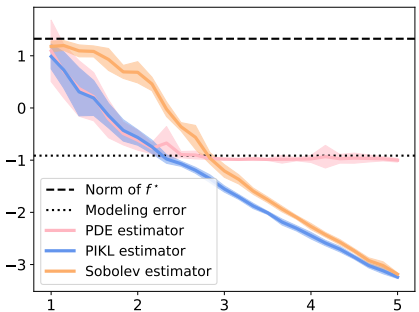
- Heat equation: $d = 2$, $\Omega = [-\pi, \pi]^2$, $\mathscr{D}(f) = \partial_1 f - \partial^2_{2,2} f$.

Experience 2: Imperfect modeling

- Heat equation: $d = 2$, $\Omega = [-\pi, \pi]^2$, $\mathscr{D}(f) = \partial_1 f - \partial^2_{2,2} f$.
- $f^\star(t, x) = \exp(-t)\cos(x) + 0.5\sin(2x)$.
- Imperfect modeling: $\|\mathscr{D}(f^\star)\|^2_{L^2(\Omega)} = \pi > 0$ and $\frac{\|\mathscr{D}(f^\star)\|^2_{L^2(\Omega)}}{\|f^\star\|^2_{L^2(\Omega)}} \approx 4 \times 10^{-3}$.

Experience 2: Imperfect modeling

- Heat equation: $d = 2$, $\Omega = [-\pi, \pi]^2$, $\mathscr{D}(f) = \partial_1 f - \partial^2_{2,2} f$.
- $f^\star(t, x) = \exp(-t)\cos(x) + 0.5\sin(2x)$.
- Imperfect modeling: $\|\mathscr{D}(f^\star)\|^2_{L^2(\Omega)} = \pi > 0$ and $\dfrac{\|\mathscr{D}(f^\star)\|^2_{L^2(\Omega)}}{\|f^\star\|^2_{L^2(\Omega)}} \approx 4 \times 10^{-3}$.



- Sobolev minimax rate of $n^{-2/3}$.
- PIKL successfully combines ML and physics.

Agenda

1. PDE constraints
2. Physics-informed kernel learner
3. **PDE solving**
4. Accelerating computation

PDE solving: PINNs vs. PIKL

- Using PIKL as PDE solver means $\mathscr{D}(f^\star) = 0$, $\varepsilon = 0$, and data $=$ uniform on $\partial\Omega$.

PDE solving: PINNs vs. PIKL

- Using PIKL as PDE solver means $\mathscr{D}(f^\star) = 0$, $\varepsilon = 0$, and data = uniform on $\partial\Omega$.
- Convection equation: on $\Omega = [0,1] \times [0, 2\pi]$,

$$\mathscr{D}(f) = \partial_t f + \beta \partial_x f, \quad \text{with} \quad \begin{cases} \forall x \in [0, 2\pi], & f(0,x) = \sin(x) \\ \forall t \in [0,1], & f(t,0) = f(t, 2\pi) = 0. \end{cases}$$

- Solution: $f^\star(t,x) = \sin(x - \beta t) \notin H_m$.

PDE solving: PINNs vs. PIKL

- Using PIKL as PDE solver means $\mathscr{D}(f^\star) = 0$, $\varepsilon = 0$, and data = uniform on $\partial\Omega$.
- Convection equation: on $\Omega = [0,1] \times [0, 2\pi]$,

$$\mathscr{D}(f) = \partial_t f + \beta \partial_x f, \quad \text{with} \quad \begin{cases} \forall x \in [0, 2\pi], & f(0,x) = \sin(x) \\ \forall t \in [0,1], & f(t,0) = f(t, 2\pi) = 0. \end{cases}$$

- Solution: $f^\star(t,x) = \sin(x - \beta t) \notin H_m$.

◇ Krishnapriyan et al. (2021)

	Vanilla PINNs◇	Curriculum-trained PINNs◇	PIKL
$\beta = 20$	7.50×10^{-1}	9.84×10^{-3}	$(\mathbf{1.56}_{\pm 3.46}) \times \mathbf{10^{-8}}$
$\beta = 30$	8.97×10^{-1}	2.02×10^{-2}	$(\mathbf{0.91}_{\pm 2.20}) \times \mathbf{10^{-7}}$
$\beta = 40$	9.61×10^{-1}	5.33×10^{-2}	$(\mathbf{7.31}_{\pm 6.44}) \times \mathbf{10^{-9}}$

- PIKL ($m = 20$) improves the solution accuracy without being sensitive to β.

PDE solving: PINNs vs. PIKL

- 1D wave equation: on $\Omega = [0,1]^2$,

$$\mathscr{D}(f) = \partial^2_{t,t} f - 4\partial^2_{x,x} f, \quad \text{with} \begin{cases} \forall x \in [0,1], & f(0,x) = \sin(\pi x) + \sin(4\pi x)/2 \\ \forall x \in [0,1], & \partial_t f(0,x) = 0 \\ \forall t \in [0,1], & f(t,0) = f(t,1) = 0. \end{cases}$$

PDE solving: PINNs vs. PIKL

- 1D wave equation: on $\Omega = [0,1]^2$,

$$\mathscr{D}(f) = \partial^2_{t,t} f - 4\partial^2_{x,x} f, \quad \text{with} \begin{cases} \forall x \in [0,1], & f(0,x) = \sin(\pi x) + \sin(4\pi x)/2 \\ \forall x \in [0,1], & \partial_t f(0,x) = 0 \\ \forall t \in [0,1], & f(t,0) = f(t,1) = 0. \end{cases}$$

- Solution: $f^\star(t,x) = \sin(\pi x)\cos(2\pi t) + \sin(4\pi x)\cos(8\pi t)/2$.
- Significant variation: $\|\partial_t f^\star\|_2^2 / \|f^\star\|_2^2 = 16\pi^2$.

PDE solving: PINNs vs. PIKL

- 1D wave equation: on $\Omega = [0,1]^2$,

$$\mathscr{D}(f) = \partial_{t,t}^2 f - 4\partial_{x,x}^2 f, \quad \text{with} \begin{cases} \forall x \in [0,1], & f(0,x) = \sin(\pi x) + \sin(4\pi x)/2 \\ \forall x \in [0,1], & \partial_t f(0,x) = 0 \\ \forall t \in [0,1], & f(t,0) = f(t,1) = 0. \end{cases}$$

- Solution: $f^\star(t,x) = \sin(\pi x)\cos(2\pi t) + \sin(4\pi x)\cos(8\pi t)/2$.
- Significant variation: $\|\partial_t f^\star\|_2^2 / \|f^\star\|_2^2 = 16\pi^2$.

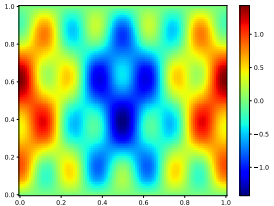
◇ Wang et al. (2022)

	Vanilla PINNs◇	NTK-optimized PINNs◇	PIKL
L^2 *relative error*	4.52×10^{-1}	1.73×10^{-3}	$(\mathbf{8.70}_{\pm 0.08}) \times \mathbf{10^{-4}}$
Training data (n)	2.4×10^6	2.4×10^6	$\mathbf{10^5}$
# *parameters*	5.03×10^5	5.03×10^5	$\mathbf{1.68 \times 10^3}$

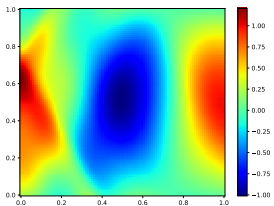
PDE solving: PINNs vs. PIKL

- 1D wave equation: on $\Omega = [0,1]^2$,

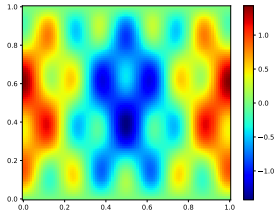
$$\mathscr{D}(f) = \partial^2_{t,t} f - 4\partial^2_{x,x} f, \quad \text{with} \begin{cases} \forall x \in [0,1], & f(0,x) = \sin(\pi x) + \sin(4\pi x)/2 \\ \forall x \in [0,1], & \partial_t f(0,x) = 0 \\ \forall t \in [0,1], & f(t,0) = f(t,1) = 0. \end{cases}$$



Ground truth



PINNs



PIKL

- PIKL is more accurate and requires fewer data points and parameters.

PIKL vs. PDE solvers

- Performance of traditional PDE solvers for the wave equation on $\Omega = [0,1]^2$:

	Euler explicit	RK4	CN	PIKL
L^2 *relative error*	$\mathbf{3.8 \times 10^{-6}}$	6.8×10^{-6}	5.6×10^{-3}	8.70×10^{-4}
Training data (n)	$\mathbf{10^4}$	10^4	10^4	10^5

☞ Traditional PDE solvers outperform PIKL (even with limited data).

PIKL vs. PDE solvers

- Performance of traditional PDE solvers for the wave equation on $\Omega = [0,1]^2$:

	Euler explicit	RK4	CN	PIKL
L^2 *relative error*	$\mathbf{3.8 \times 10^{-6}}$	6.8×10^{-6}	5.6×10^{-3}	8.70×10^{-4}
Training data (n)	$\mathbf{10^4}$	10^4	10^4	10^5

☞ Traditional PDE solvers outperform PIKL (even with limited data).

- Performance for the wave equation with noisy boundary conditions:

	PINNs	Euler explicit	RK4	CN	PIKL
L^2 *relative error*	4.61×10^{-1}	1.25×10^{-1}	6.05×10^{-2}	2.01×10^{-2}	$\mathbf{1.87 \times 10^{-2}}$
Training data (n)	2.4×10^6	4×10^4	4×10^4	4×10^4	$\mathbf{4 \times 10^4}$

☞ PIKL outperforms PDE solvers.

Agenda

1. PDE constraints
2. Physics-informed kernel learner
3. PDE solving
4. **Accelerating computation**

FAST KERNEL METHODS: SOBOLEV, PHYSICS-INFORMED, AND ADDITIVE MODELS

Nathan Doumèche
LPSM
Sorbonne Université

Francis Bach
INRIA Paris

Gérard Biau
LPSM
Sorbonne Université

Claire Boyer
LMO
Université Paris-Saclay
{nathan.doumeche, gerard.biau}@sorbonne-universite.fr
claire.boyer@universite-paris-saclay.fr
francis.bach@inria.fr

ABSTRACT

Kernel methods are powerful tools in statistical learning, but their cubic complexity in the sample size n limits their use on large-scale datasets. In this work, we introduce a scalable framework for kernel regression with $\mathcal{O}(n \log n)$ complexity, fully leveraging GPU acceleration. The approach is based on a Fourier representation of kernels combined with non-uniform fast Fourier transforms (NUFFT), enabling exact, fast, and memory-efficient computations. We instantiate our framework in three settings: Sobolev kernel regression, physics-informed regression, and additive models. The proposed estimators are shown to achieve minimax convergence rates, consistent with classical kernel theory. Empirical results demonstrate that our methods can process up to tens of billions of samples within minutes, providing both statistical accuracy and computational scalability. These contributions establish a flexible approach, paving the way for the routine application of kernel methods in large-scale learning tasks.

1 INTRODUCTION

Kernel methods. Kernel methods play a central role in statistical learning and nonparametric regression, providing flexible and theoretically sound tools for modeling complex data structures (Steinwart & Christmann, 2008). Despite their appeal, their practical application is often hindered by severe computational constraints: standard kernel ridge regression has a time complexity of $\mathcal{O}(n^3)$ and a memory cost of $\mathcal{O}(n^2)$, making it unsuitable for large-scale datasets (Bach, 2024, Chapter 7). Several approximation schemes, such as Nyström methods and random feature expansions, have been proposed to reduce this burden, reducing the computational cost to $\mathcal{O}(n^{3/2})$. However, these approximations introduce additional error terms, complicating theoretical guarantees and often degrading empirical performance (Meanti et al., 2020).

Contributions. We show that a broad class of kernel estimators can be trained exactly at a substantially reduced computational cost, achieving $\mathcal{O}(n \log n)$ complexity in terms of both time and memory. This improvement is made possible by taking advantage of the structural properties of kernels that can be expanded on a Fourier basis. Furthermore, our approach lends itself well to parallel computation and can be efficiently implemented on modern GPU architectures. These advances are a significant step forward in both the theory and practice of kernel methods, with clear potential for routine application to large-scale datasets. A key component of our approach is the recent implementation of the non-uniform fast Fourier transform on GPU (NUFFT, Shih et al., 2021). The resulting computational efficiency enables the processing of tens of billions of data points in under a minute on a standard GPU, whereas conventional kernel approaches typically fail to scale beyond a few hundred thousand samples (Meanti et al., 2020; Doumèche et al., 2025a).

arXiv:2509.02649

Kernel methods

- Empirical risk:

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}}$$

Kernel methods

- Empirical risk:

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}} \qquad \text{over } H_m = \mathrm{Span}((\phi_k)_{\|k\|_\infty \leqslant m}).$$

Kernel methods

- Empirical risk:

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}} \qquad \text{over } H_m = \mathrm{Span}((\phi_k)_{\|k\|_\infty \leqslant m}).$$

- Finite-dimensional approximation:

$$R_n(\theta) = n^{-1}\|\Phi\theta - \mathbb{Y}\|_2^2 + \lambda_n \|M\theta\|_2^2.$$

Kernel methods

- Empirical risk:

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}} \qquad \text{over } H_m = \mathrm{Span}((\phi_k)_{\|k\|_\infty \leqslant m}).$$

- Finite-dimensional approximation:

$$R_n(\theta) = n^{-1}\|\Phi\theta - \mathbb{Y}\|_2^2 + \lambda_n \|M\theta\|_2^2.$$

- Estimated parameter:

$$\hat{\theta}_n = \underset{\theta \in (2m+1)^d}{\operatorname{argmin}} R_n(\theta) = \left(n^{-1}\Phi^*\Phi + \lambda_n M^* M\right)^{-1} n^{-1}\Phi^*\mathbb{Y}.$$

Kernel methods

- Empirical risk:

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}} \qquad \text{over } H_m = \mathrm{Span}((\phi_k)_{\|k\|_\infty \leqslant m}).$$

- Finite-dimensional approximation:

$$R_n(\theta) = n^{-1}\|\Phi\theta - \mathbb{Y}\|_2^2 + \lambda_n \|M\theta\|_2^2.$$

- Estimated parameter:

$$\hat{\theta}_n = \underset{\theta \in (2m+1)^d}{\mathrm{argmin}} \; R_n(\theta) = \left(n^{-1}\Phi^*\Phi + \lambda_n M^* M\right)^{-1} n^{-1}\Phi^*\mathbb{Y}.$$

- Physics-informed:

$$\hat{\theta}_n = \left(n^{-1}\Phi^*\Phi + \lambda_n S^2 + \mu_n D^* C D\right)^{-1} n^{-1}\Phi^*\mathbb{Y}.$$

Kernel methods

- Empirical risk:

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} |f(X_i) - Y_i|^2 + \lambda_n \|f\|^2_{\mathrm{RKHS}} \qquad \text{over } H_m = \mathrm{Span}((\phi_k)_{\|k\|_\infty \leqslant m}).$$

- Finite-dimensional approximation:

$$R_n(\theta) = n^{-1}\|\Phi\theta - \mathbb{Y}\|_2^2 + \lambda_n \|M\theta\|_2^2.$$

- Estimated parameter:

$$\hat{\theta}_n = \underset{\theta \in (2m+1)^d}{\mathrm{argmin}} \; R_n(\theta) = \left(n^{-1}\Phi^*\Phi + \lambda_n M^* M\right)^{-1} n^{-1}\Phi^*\mathbb{Y}.$$

- Physics-informed:

$$\hat{\theta}_n = \left(n^{-1}\Phi^*\Phi + \lambda_n S^2 + \mu_n D^* C D\right)^{-1} n^{-1}\Phi^*\mathbb{Y}.$$

- Final estimator:

$$\hat{f}_n(x) = \langle \Phi_m(x), \hat{\theta}_n \rangle.$$

Computational considerations

- Number of Fourier modes: $(2m+1)^d$

Computational considerations

- Number of Fourier modes: $(2m+1)^d = \mathcal{O}(n^{d/(2s+d)})$ with $m = n^{1/(2s+d)}$.

Computational considerations

- Number of Fourier modes: $(2m+1)^d = \mathcal{O}(n^{d/(2s+d)})$ with $m = n^{1/(2s+d)}$.
- Naive computational cost: $\mathcal{O}(n^{1+2d/(2s+d)})$.

Computational considerations

- Number of Fourier modes: $(2m+1)^d = \mathcal{O}(n^{d/(2s+d)})$ with $m = n^{1/(2s+d)}$.
- Naive computational cost: $\mathcal{O}(n^{1+2d/(2s+d)})$.
- Optimized cost: $\mathcal{O}(n \log n)$.

Computational considerations

- Number of Fourier modes: $(2m+1)^d = \mathcal{O}(n^{d/(2s+d)})$ with $m = n^{1/(2s+d)}$.
- Naive computational cost: $\mathcal{O}(n^{1+2d/(2s+d)})$.
- Optimized cost: $\mathcal{O}(n \log n)$.
- Key ingredients: non-uniform FFT (NUFFT) + GPU acceleration.

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n$

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \rightarrow$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \rightarrow$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗
- Each component of v is

$$v_k = \frac{1}{n} \sum_{j=1}^{n} Y_j \, e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \rightarrow$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗
- Each component of v is

$$v_k = \frac{1}{n} \sum_{j=1}^{n} Y_j \, e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ This is a type-I NUFFT

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \rightarrow$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗
- Each component of v is

$$v_k = \frac{1}{n} \sum_{j=1}^{n} Y_j \, e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ This is a type-I NUFFT $\quad \rightarrow \mathcal{O}(n \log n)$ ✓

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \rightarrow$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗
- Each component of v is

$$v_k = \frac{1}{n} \sum_{j=1}^{n} Y_j \, e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ This is a type-I NUFFT $\quad \rightarrow \mathcal{O}(n \log n)$ ✓

- Covariance matrix: $\hat{\Sigma} = \Phi^* \Phi / n$

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \to$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗
- Each component of v is

$$v_k = \frac{1}{n} \sum_{j=1}^{n} Y_j \, e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ This is a type-I NUFFT $\quad \to \mathcal{O}(n \log n)$ ✓

- Covariance matrix: $\hat{\Sigma} = \Phi^* \Phi/n \quad \to$ naive $\mathcal{O}(n^{1+2d/(2s+d)})$ ✗

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \to$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗
- Each component of v is

$$v_k = \frac{1}{n} \sum_{j=1}^{n} Y_j \, e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ This is a type-I NUFFT $\quad \to \mathcal{O}(n \log n)$ ✓

- Covariance matrix: $\hat{\Sigma} = \Phi^* \Phi / n \quad \to$ naive $\mathcal{O}(n^{1+2d/(2s+d)})$ ✗
- But $\hat{\Sigma}$ is both Hermitian and d-level block Toeplitz.

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \rightarrow$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗
- Each component of v is

$$v_k = \frac{1}{n} \sum_{j=1}^{n} Y_j \, e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ This is a type-I NUFFT $\quad \rightarrow \mathcal{O}(n \log n)$ ✓

- Covariance matrix: $\hat{\Sigma} = \Phi^* \Phi/n \quad \rightarrow$ naive $\mathcal{O}(n^{1+2d/(2s+d)})$ ✗
- But $\hat{\Sigma}$ is both Hermitian and d-level block Toeplitz.
- Each entry may be written as

$$\hat{\Sigma}_{0,k} = \frac{1}{n} \sum_{j=1}^{n} e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \rightarrow$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗
- Each component of v is

$$v_k = \frac{1}{n} \sum_{j=1}^{n} Y_j \, e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ This is a type-I NUFFT $\quad \rightarrow \mathcal{O}(n \log n)$ ✓

- Covariance matrix: $\hat{\Sigma} = \Phi^* \Phi / n \quad \rightarrow$ naive $\mathcal{O}(n^{1+2d/(2s+d)})$ ✗
- But $\hat{\Sigma}$ is both Hermitian and d-level block Toeplitz.
- Each entry may be written as

$$\hat{\Sigma}_{0,k} = \frac{1}{n} \sum_{j=1}^{n} e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ Again a type-I NUFFT

NUFFT

- Covariance vector: $v = \Phi^* \mathbb{Y}/n \quad \rightarrow$ naive $\mathcal{O}(n^{1+d/(2s+d)})$ ✗
- Each component of v is

$$v_k = \frac{1}{n} \sum_{j=1}^{n} Y_j \, e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ This is a type-I NUFFT $\quad \rightarrow \mathcal{O}(n \log n)$ ✓

- Covariance matrix: $\hat{\Sigma} = \Phi^* \Phi / n \quad \rightarrow$ naive $\mathcal{O}(n^{1+2d/(2s+d)})$ ✗
- But $\hat{\Sigma}$ is both Hermitian and d-level block Toeplitz.
- Each entry may be written as

$$\hat{\Sigma}_{0,k} = \frac{1}{n} \sum_{j=1}^{n} e^{\frac{i\pi}{2L}\langle k, X_j \rangle}.$$

☞ Again a type-I NUFFT $\rightarrow \mathcal{O}(n \log n)$ with GPU acceleration ✓

Back to PIKL

- Estimated parameter:

$$\hat{\theta}_n = \left(n^{-1}\Phi^*\Phi + \lambda_n S^2 + \mu_n D^* C D\right)^{-1} n^{-1}\Phi^*\mathbb{Y},$$

Back to PIKL

- Estimated parameter:

$$\hat{\theta}_n = \left(n^{-1}\Phi^*\Phi + \lambda_n S^2 + \mu_n D^* C D\right)^{-1} n^{-1}\Phi^*\mathbb{Y},$$

where

1. $S_{k,k} = \sqrt{1 + (\|k\|_2^2/(2L)^d)^s}$

Back to PIKL

- Estimated parameter:

$$\hat{\theta}_n = \left(n^{-1}\Phi^*\Phi + \lambda_n S^2 + \mu_n D^* C D\right)^{-1} n^{-1}\Phi^*\mathbb{Y},$$

where

1. $S_{k,k} = \sqrt{1 + (\|k\|_2^2/(2L)^d)^s}$
2. $C_{k_1,k_2} = (4L)^{-d} \int_\Omega e^{\frac{i\pi}{2L}\langle k_1 - k_2, x\rangle} dx$

Back to PIKL

- Estimated parameter:

$$\hat{\theta}_n = (n^{-1}\Phi^*\Phi + \lambda_n S^2 + \mu_n D^* C D)^{-1}\, n^{-1}\Phi^*\mathbb{Y},$$

where

1. $S_{k,k} = \sqrt{1 + (\|k\|_2^2/(2L)^d)^s}$
2. $C_{k_1,k_2} = (4L)^{-d} \int_\Omega e^{\frac{i\pi}{2L}\langle k_1 - k_2, x\rangle} dx$
3. $D_{k,k} = \sum_{|\alpha|\leqslant s} a_\alpha (\frac{-i\pi}{2L})^{|\alpha|} \prod_{\ell=1}^d (k_\ell)^{\alpha_\ell}$.

Back to PIKL

- Estimated parameter:

$$\hat{\theta}_n = (n^{-1}\Phi^*\Phi + \lambda_n S^2 + \mu_n D^* C D)^{-1}\, n^{-1}\Phi^*\mathbb{Y},$$

where

1. $S_{k,k} = \sqrt{1 + (\|k\|_2^2/(2L)^d)^s}$
2. $C_{k_1,k_2} = (4L)^{-d} \int_\Omega e^{\frac{i\pi}{2L}\langle k_1 - k_2, x\rangle} dx$
3. $D_{k,k} = \sum_{|\alpha|\leqslant s} a_\alpha (\frac{-i\pi}{2L})^{|\alpha|} \prod_{\ell=1}^d (k_\ell)^{\alpha_\ell}$.

- Experiment: X is uniform over $\Omega =]0,1[$, $Y = \exp(X) + \mathcal{N}(0,1)$.

Back to PIKL

- Estimated parameter:

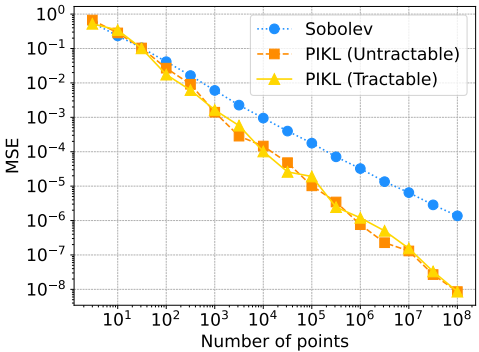
$$\hat{\theta}_n = (n^{-1}\Phi^*\Phi + \lambda_n S^2 + \mu_n D^* C D)^{-1}\, n^{-1}\Phi^*\mathbb{Y},$$

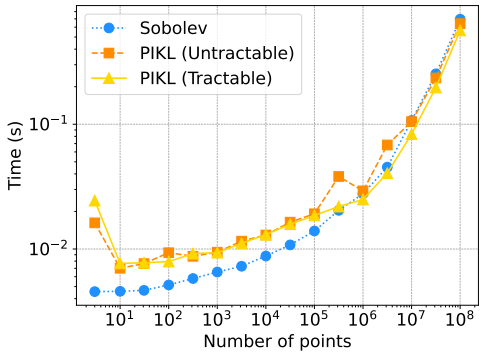
where

1. $S_{k,k} = \sqrt{1 + (\|k\|_2^2/(2L)^d)^s}$
2. $C_{k_1,k_2} = (4L)^{-d} \int_\Omega e^{\frac{i\pi}{2L}\langle k_1 - k_2, x\rangle} dx$
3. $D_{k,k} = \sum_{|\alpha|\leqslant s} a_\alpha (\frac{-i\pi}{2L})^{|\alpha|} \prod_{\ell=1}^d (k_\ell)^{\alpha_\ell}$.

- Experiment: X is uniform over $\Omega =]0,1[$, $Y = \exp(X) + \mathcal{N}(0,1)$.
- Prior knowledge: $(f^\star)' - f^\star = 0$.

Back to PIKL


Sobolev
PIKL (Untractable)
PIKL (Tractable)
MSE
Number of points


Sobolev
PIKL (Untractable)
PIKL (Tractable)
Time (s)
Number of points

Additive models

- Model: $f^\star(x_1, \ldots, x_d) = g_1^\star(x_1) + \cdots + g_d^\star(x_d)$.

Additive models

- Model: $f^\star(x_1, \ldots, x_d) = g_1^\star(x_1) + \cdots + g_d^\star(x_d)$.
- Estimated parameter:

$$\hat{\theta}_n = (\hat{\Sigma} + \lambda_n \mathrm{I})^{-1}(n^{-1}\Phi_1^* \mathbb{Y}, \ldots, n^{-1}\Phi_d^* \mathbb{Y})^\top.$$

Additive models

- Model: $f^\star(x_1, \dots, x_d) = g_1^\star(x_1) + \cdots + g_d^\star(x_d)$.
- Estimated parameter:

$$\hat{\theta}_n = (\hat{\Sigma} + \lambda_n \mathrm{I})^{-1}(n^{-1}\Phi_1^* \mathbb{Y}, \dots, n^{-1}\Phi_d^* \mathbb{Y})^\top.$$

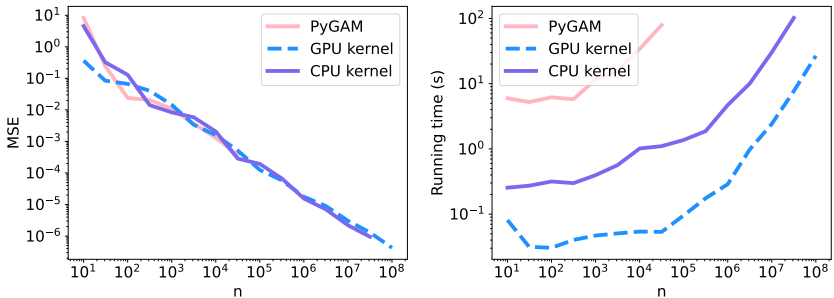
- Choosing $\lambda_n = n^{-2s/(2s+1)}$ and $m = n^{1/(2s+1)}/d$ yields

$$\mathbb{E} \int_\Omega |f_{\hat{\theta}_n} - f^\star|^2 d\mathbb{P}_X = \mathcal{O}(n^{-2s/(2s+1)}).$$

Additive models

- Model: $f^\star(x_1, \dots, x_d) = g_1^\star(x_1) + \cdots + g_d^\star(x_d)$.
- Estimated parameter:

$$\hat{\theta}_n = (\hat{\Sigma} + \lambda_n \mathrm{I})^{-1} (n^{-1} \Phi_1^* \mathbb{Y}, \dots, n^{-1} \Phi_d^* \mathbb{Y})^\top.$$



Conclusion

- Minimizing an empirical risk regularized by a PDE = a kernel method.
- Incorporating physical structure can significantly improve statistical performance.
- PIKL: a kernel-based toolbox for physics-informed machine learning.
- Next step: integrating constraints into forecasting.

Thank you!