



SEA-CROGS

Data-efficient Kernel Methods for PDE Discovery

Yasamin Jalalian

California Institute of Technology

Joint work with Juan F. Osorio, Alex W. Hsu, Bamdad Hosseini and Houman Owhadi

Problem Statement

$$P(x, u(x), Du(x), D^2u(x)) = f(x), \quad \forall x \in \Omega \subset \mathbb{R}^d$$

Problem:

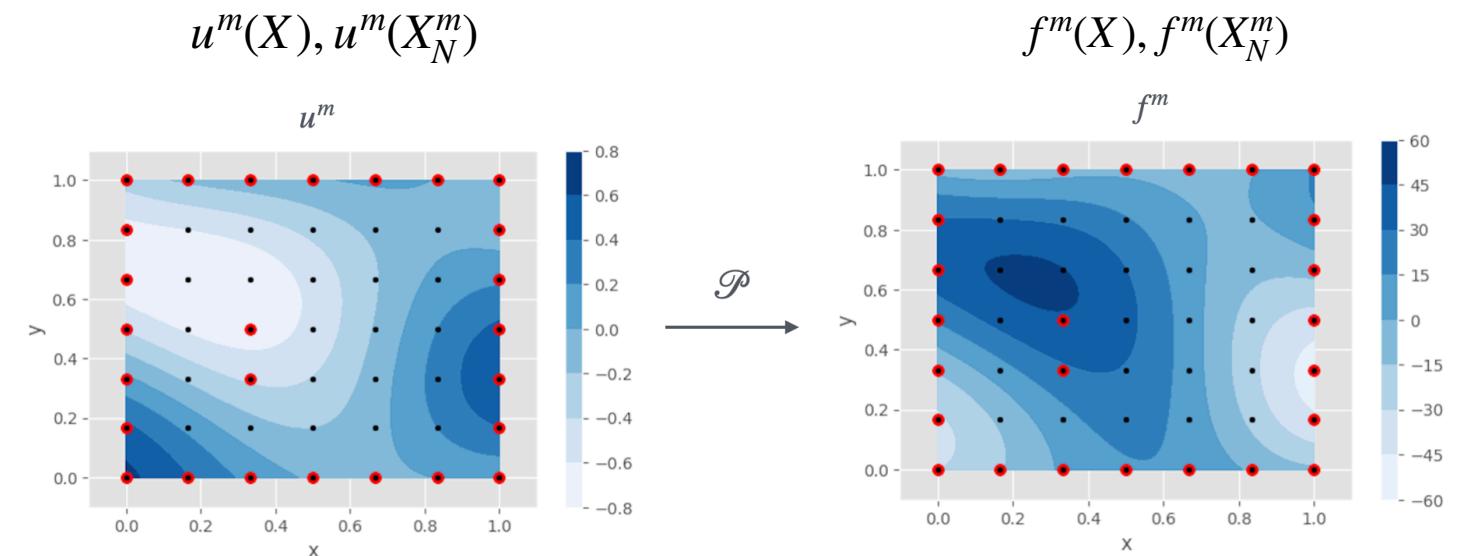
- $P : \mathbb{R}^{d+q} \rightarrow \mathbb{R}$ **unknown** possibly nonlinear function of known order 2
- $f : \Omega \rightarrow \mathbb{R}$ **partially observed** source term (continuous function)
- $u : \Omega \rightarrow \mathbb{R}$ **partially observed** function describing the state of the physical system (strong solution to the PDE with source term f)

Data:

- data points $X \subset \Omega$
 M pairs of functions (u^m, f^m)
 N observation points $X_N^m = \{x_1^m, \dots, x_N^m\} \subset X$

Objective:

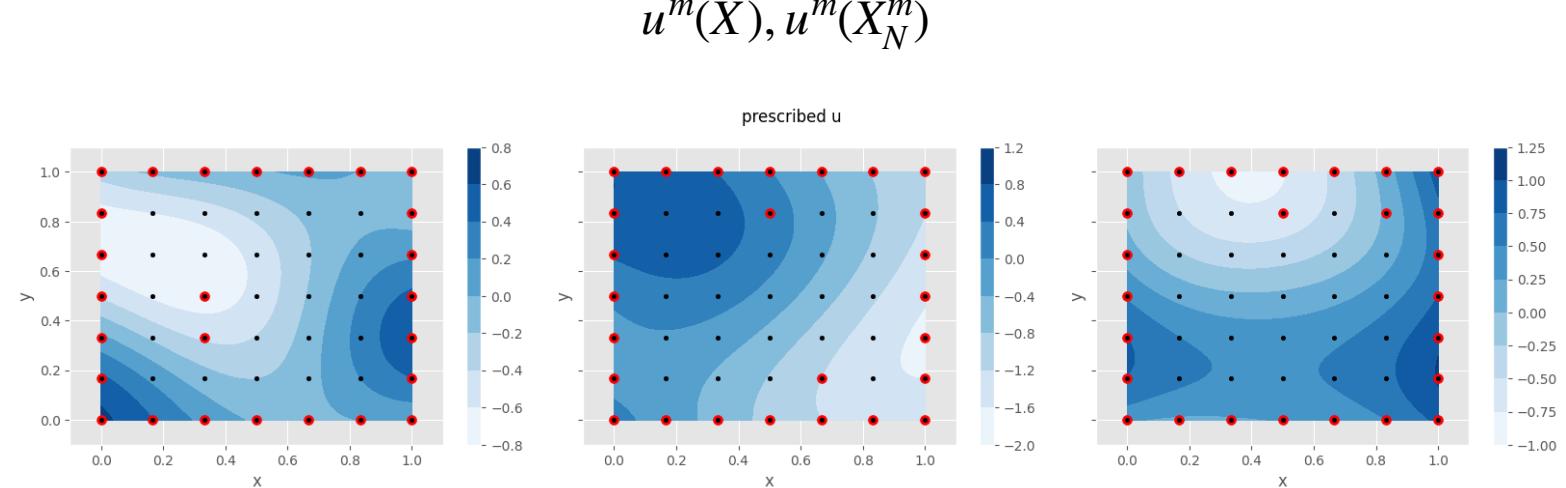
- learn the function P and the functions u^m on the continuum from data



Example: Darcy Flow

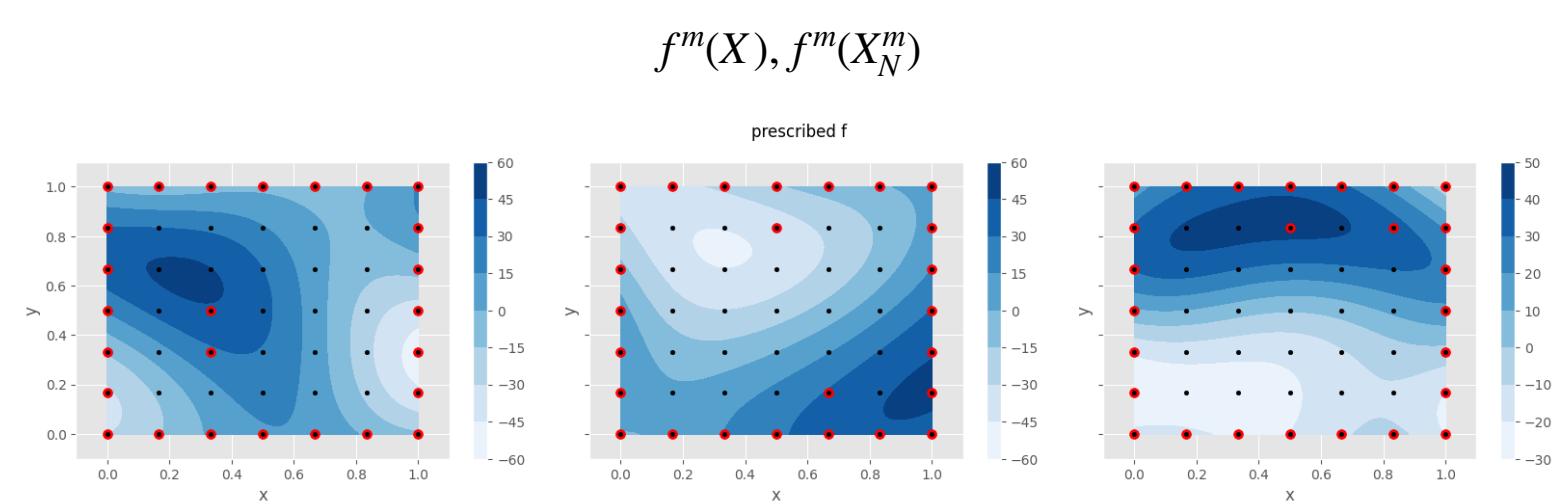
Problem:

$$P(x, u(x), Du(x), D^2u(x)) = \nabla \cdot (a(x) \nabla u(x)) = f(x), \quad \forall x \in (0,1)^2$$



Data:

- $M = 3$ pairs of (u^m, f^m) , for $m \in \{1,2,3\}$
- 7×7 data points $X, N = 26$ observation points X_N^m



Objective:

- learn the function P and the functions u^m on the continuum from data





Data-driven Methods for Solving & Learning PDEs

Current Approaches:

1. ANN-based methods

- Solving high-dimensional PDEs using deep learning. Han, Jentzen, E, 2018
- Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations, Maziar Raissi, 2018
- Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. M. Raissi, P. Perdikaris, G.E. Karniadakis, JCP 2019
- Model reduction and neural networks for parametric PDEs. K. Bhattacharya, B. Hosseini, N. Kovachki, A. Stuart. 2020.
- Physics-informed learning of governing equations from scarce data. Zhao Chen, Yang Liu, Hao Sun. 2021
- Competitive Physics Informed Networks. Q. Zeng, Y. Kothari, S. Bryngelson, F. Schäfer., 2022
- Self-adaptive physics-informed neural networks. L. McClelly, U. M Braga-Neto. 2023

2. Kernel-based methods

- Bayesian Numerical Homogenization. Houman Owhadi. 2015
- Gamblets for opening the complexity-bottleneck of implicit schemes for hyperbolic and parabolic ODEs/PDEs with rough coefficients. Houman Owhadi, Lei Zhang, JCP 2016
- Machine learning of linear differential equations using Gaussian processes. Maziar Raissi, Paris Perdikaris, George Em Karniadaki, 2017
- Numerical Gaussian Processes for Time-dependent and Non-linear Partial Differential Equations. Maziar Raissi, Paris Perdikaris, George Em Karniadakis, 2017
- Inferring solutions of differential equations using noisy multi-fidelity data. Maziar Raissi, Paris Perdikaris, George Em Karniadakis, SISC 2018
- Solving and Learning Nonlinear PDEs with Gaussian Processes. Yifan Chen, Bamdad Hosseini, Houman Owhadi, Andrew M Stuart, 2021



Data-driven Methods for Solving & Learning PDEs

Current Approaches:

1. ANN-based methods
2. Kernel-based methods

Advantages of kernel-based method:

- Theoretically more well-founded
- Interpretable and amenable to numerical analysis
- Number of kernel parameters to learn is smaller
- Come with rigorous a priori error estimates



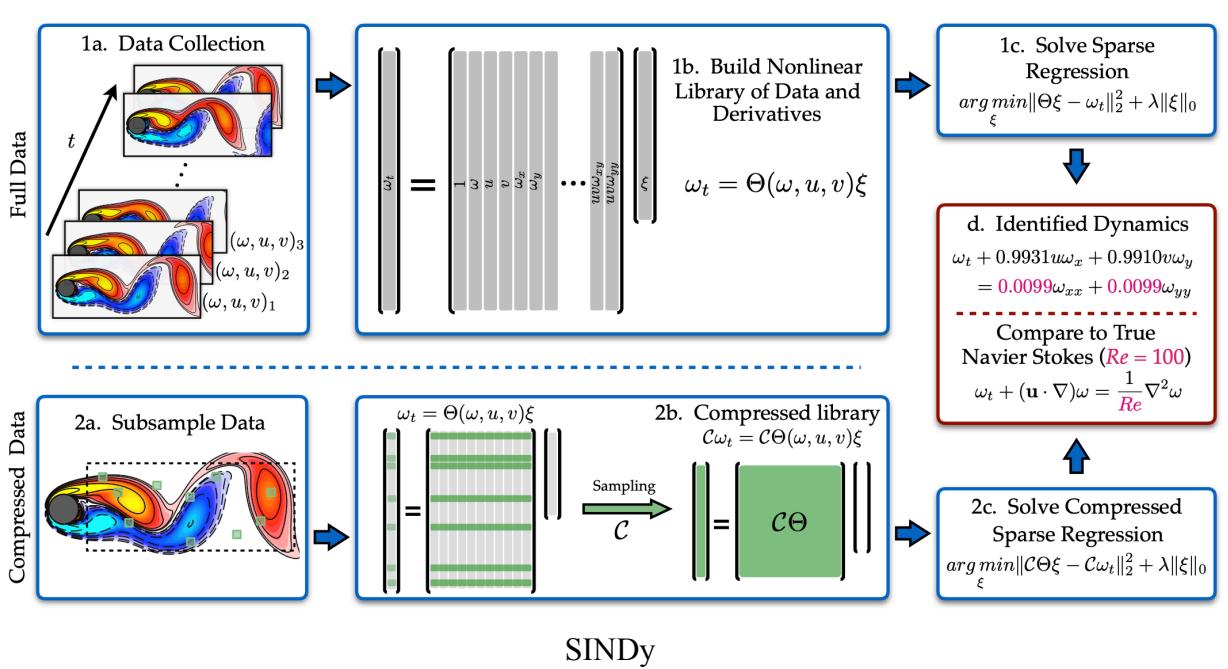
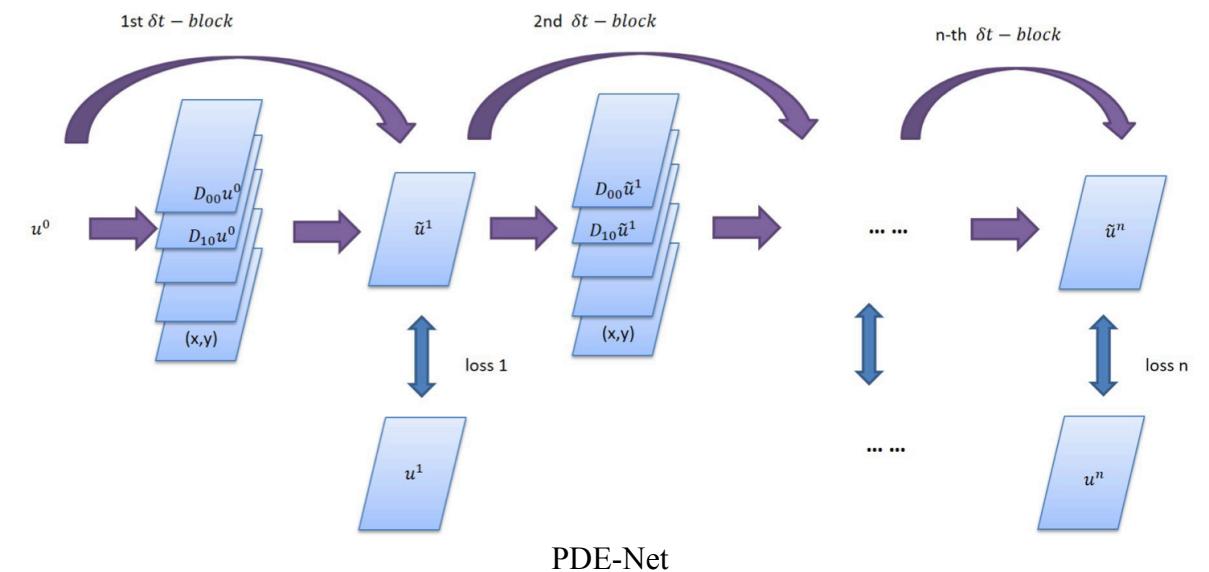
Equation Learning

Other Approaches:

- **Symbolic Regression** (Bongard & Lipson (2007), Schmidt & Lipson (2009))
- **SINDy and PDE-Find** (Brunton et al. (2016), Rudy et al. (2016))
- **PDE-Net** (Zichao Long et al. (2018))

Advantages of kernel-based method:

- **Handle sparse data** (Da Long et al. (2023))
- **A priori error estimates** (Battelle et al. (2023))





Kernel Interpolation Problem

Data:

$$\text{inputs } X = (x_1, \dots, x_n)^\top, \text{outputs } Y = (y_1, \dots, y_n)^\top$$

Goal:

$$\text{find } f : \mathcal{X} \rightarrow \mathbb{R} \text{ such that } f(x_i) = y_i$$

Interpolation Problem:

$$\begin{cases} \underset{f \in \mathcal{H}}{\text{minimize}} & \|f\|_{\mathcal{H}} \\ \text{s.t.} & f(x_i) = y_i \quad \text{for } i = 1, \dots, n. \end{cases}$$

Solution: Using representer theorem, we get a closed form solution

$$f(x) = K(x, X)K(X, X)^{-1}Y,$$

with norm

$$\|f\|_{\mathcal{H}}^2 = Y^\top K(X, X)^{-1}Y.$$



Summary of Methods

Problem:

$$P(x, u^m(x), Du^m(x), D^2u^m(x)) = f^m(x) \quad \forall x \in \Omega$$

Two Approaches:

- **two-step learning**
 1. approximate the solutions u^m
 2. using the approximant, learn the function P
- **one-step learning**
 1. learn the solution functions u^m and the function P at once





Two-step learning

Problem:

$$P(x, u^m(x), Du^m(x), D^2u^m(x)) = f^m(x) \quad \forall x \in \Omega$$

Kernel Interpolation Problem:

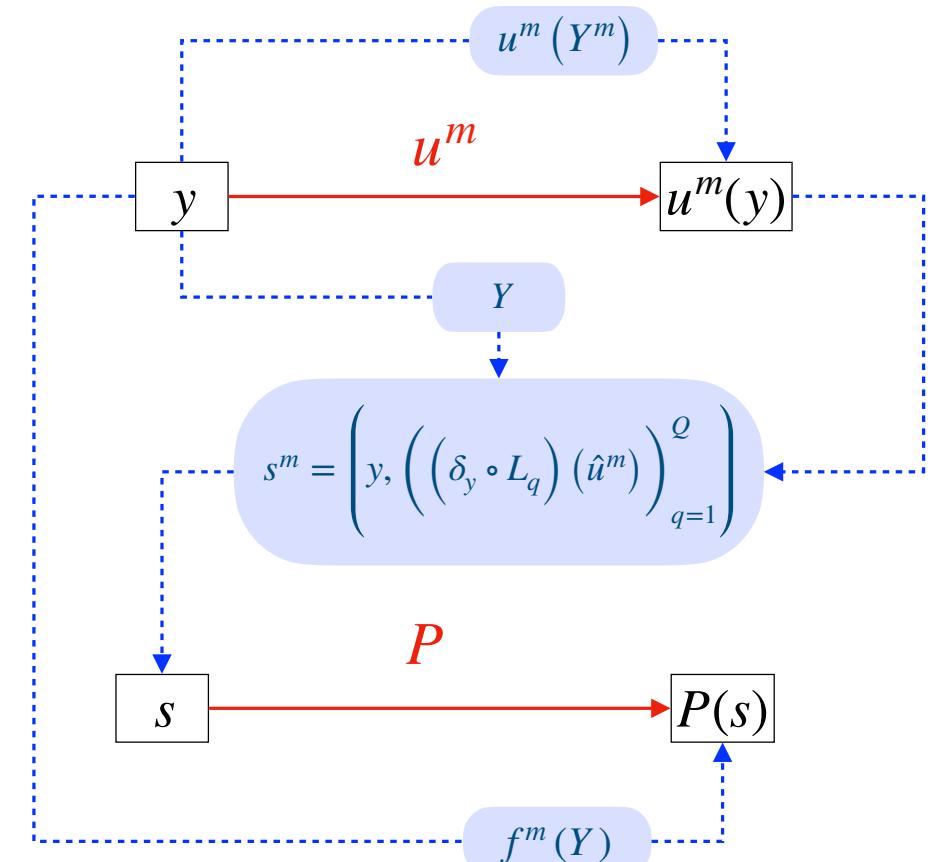
1. Learn u^m 's:

$$\begin{cases} \text{minimize}_{v^m \in \mathcal{H}_K} & \|v^m\|_{\mathcal{H}_K} \\ \text{s.t.} & v^m(X_N) = u^m(X_N) \end{cases}$$

2. Learn P : for minimizer v_N^m ,

- compute Dv_N^m and $D^2v_N^m$,
- define $\phi_N^m := (v_N^m(X_N), Dv_N^m(X_N), D^2v_N^m(X_N))$,
- solve

$$\underset{Q \in \mathcal{H}_\Gamma}{\text{minimize}} \quad \|Q\|_{\mathcal{H}_\Gamma}^2 + \lambda \sum_{m=1}^M \|Q(X_N, \phi_N^m) - f^m(X_N)\|_2^2$$





One-step learning

Problem:

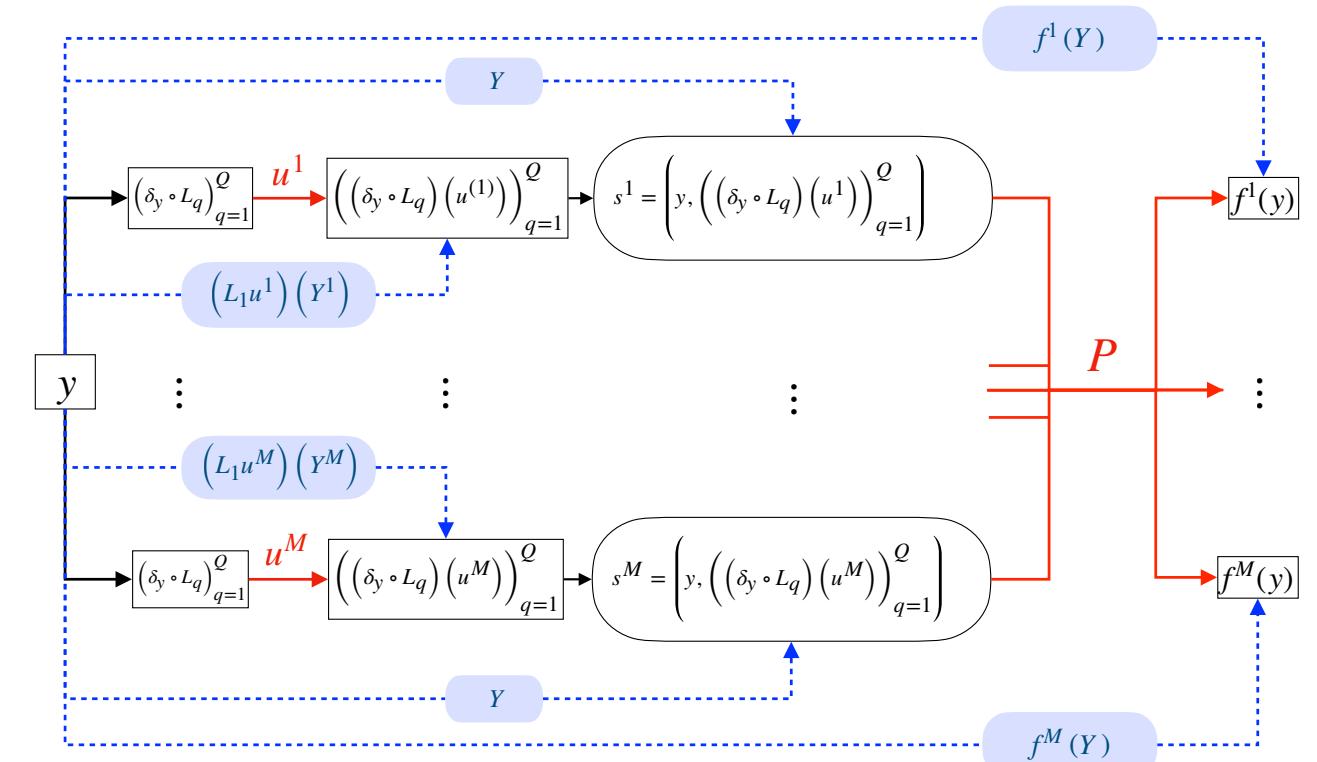
$$P(x, u^m(x), Du^m(x), D^2u^m(x)) = f^m(x) \quad \forall x \in \Omega$$

Kernel Interpolation Problem:

- **solve**

$$\begin{cases} \underset{Q \in \mathcal{H}_\Gamma, v^m \in \mathcal{H}_K}{\text{minimize}} \|Q\|_\Gamma^2 + \lambda \sum_{m=1}^M \|v^m\|_K^2 \\ \text{s.t.} \quad \quad \quad v^m(X_N) = u^m(X_N), \quad \text{for } m = 1, \dots, M, \\ \quad \quad \quad Q(X_N, \phi_N^m) = f^m(X_N), \quad \text{for } m = 1, \dots, M, \end{cases}$$

where $\phi_N^m := (v^m(X_N), Dv^m(X_N), D^2v^m(X_N))$.

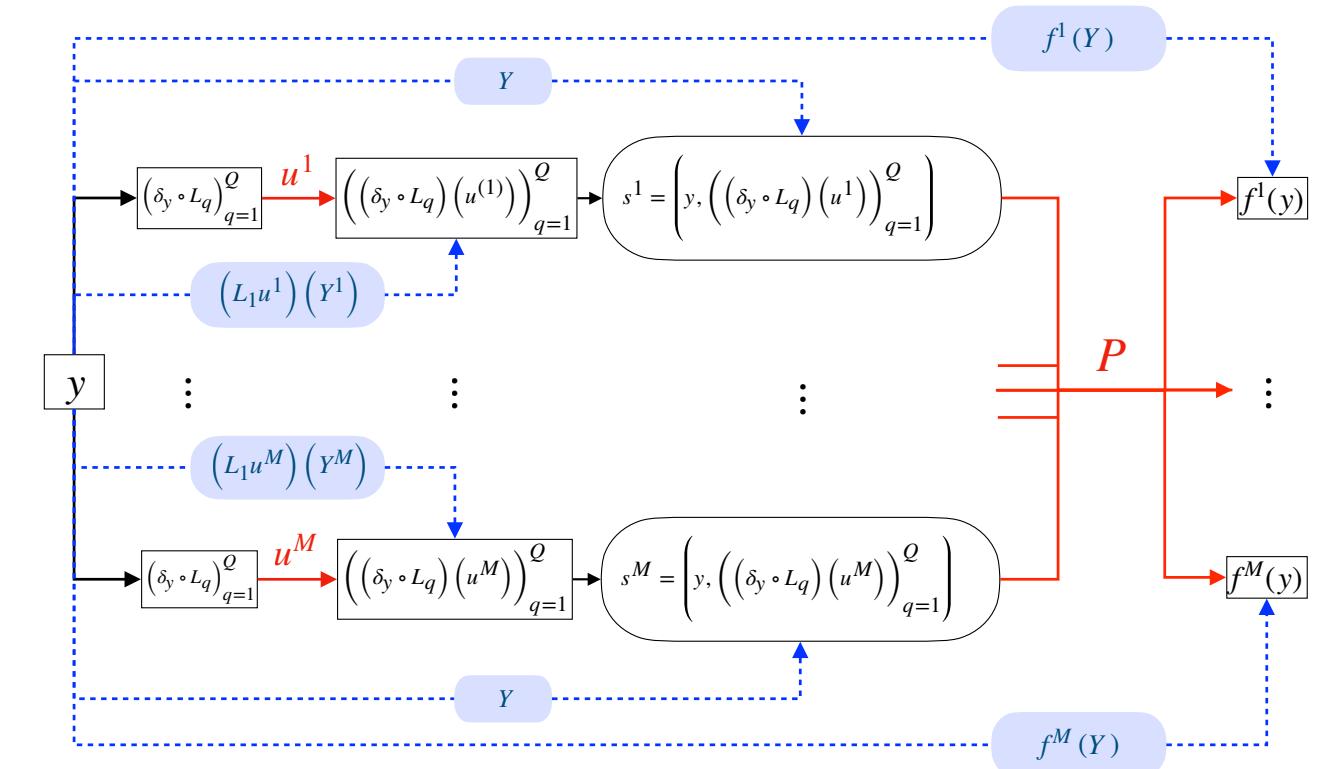




One-step learning

- introduce latent variables Z_N^m such that $Z_N^m = \phi_N^m$,
- rewrite the problem as

$$\left\{ \begin{array}{ll} \text{minimize}_{Q \in \mathcal{H}_\Gamma, v^m \in \mathcal{H}_K, Z_N^m} & \|Q\|_\Gamma^2 + \lambda \sum_{m=1}^M \|v^m\|_K^2 \\ \text{s.t.} & v^m(X_N) = u^m(X_N), \quad \text{for } m = 1, \dots, M, \\ & Q(Z_N^m) = f^m(X_N), \quad \text{for } m = 1, \dots, M, \\ & Z_N^m = \phi_N^m, \quad \text{for } m = 1, \dots, M. \end{array} \right.$$



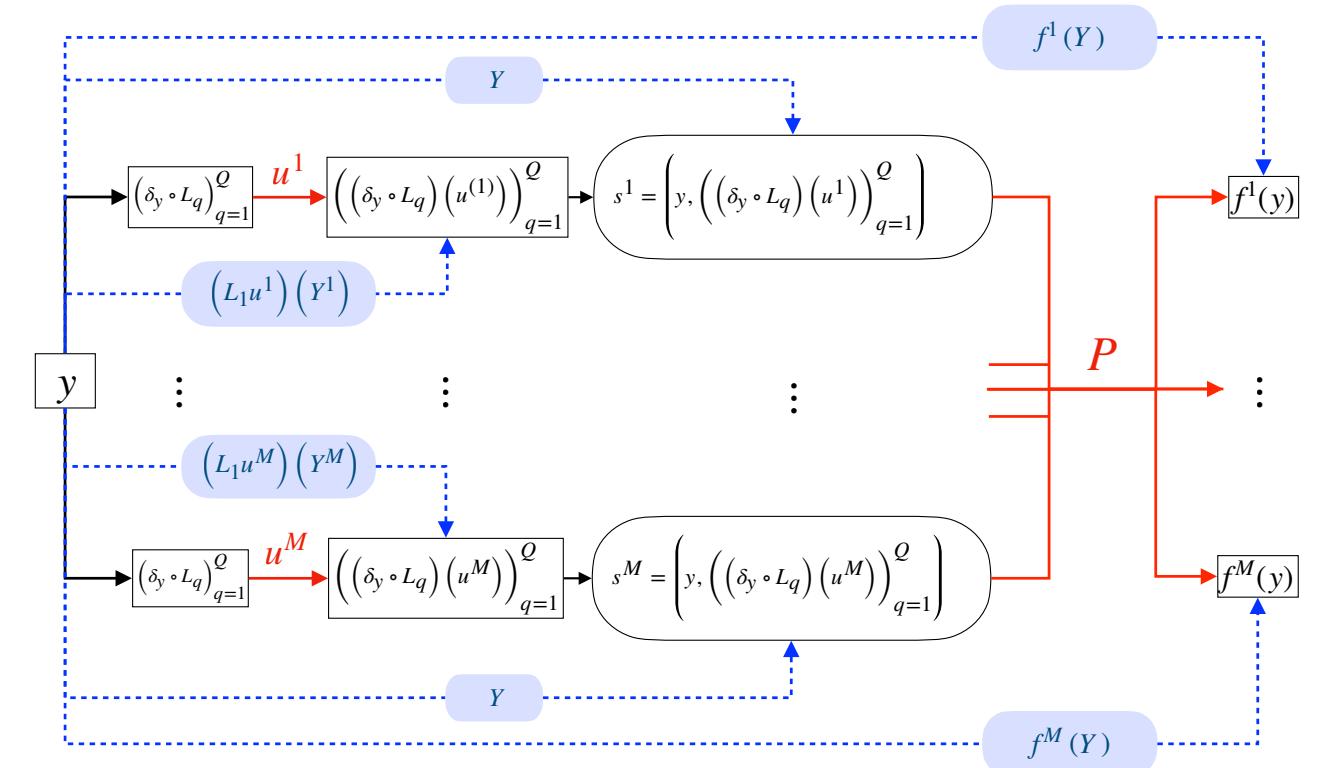


One-step learning

- **define $S_N^m := (X_N, Z_N^m)$,**
- **using representer theorem, reduce the problem to**

$$\begin{cases} \text{minimize}_{Z_N^m} & f(X_N)^\top \Gamma(S_N, S_N)^{-1} f(X_N) + \lambda \sum_{m=1}^M (Z_N^m)^\top K(\phi_N^m, \phi_N^m)^{-1} Z_N^m \\ \text{s.t.} & Z_{N,0}^m = u^m(X_N), \quad \text{for } m = 1, \dots, M. \end{cases}$$

- **solve numerically using a Levenberg–Marquardt-type algorithm**



Theorem (convergence guarantee)

Assumptions:

- the kernel $K : \Omega \times \Omega \rightarrow \mathbb{R}$ is such that $\mathcal{H}_K \subset H^s(\Omega)$ for some $s > d/2 + \text{order}_P$
- the kernel $\Gamma : \mathbb{R}^{d+q} \times \mathbb{R}^{d+q} \rightarrow \mathbb{R}$ be such that $\mathcal{H}_\Gamma \subset H^{s'}(\mathbb{R}^{d+q})$ for some $s' > \frac{q}{2}$
- the fill distance between $x_j \in \Omega$ goes to zero as $N \rightarrow \infty$
- the fill distance $h_{S_N, \mathbb{R}^{d+q}} \rightarrow 0$ as $M \rightarrow \infty$

Let $v_N^m, Q_{M,N}$ be the minimizers of the kernel interpolation problem.

$$\lim_{N \rightarrow \infty} v_N^m = u^m, \quad \text{pointwise and in } H^t(t < s) \text{ in } \Omega$$

$$\lim_{M \rightarrow \infty} \lim_{N \rightarrow \infty} Q_{M,N} = P, \quad \text{pointwise and in } H^{t'}(t' < s') \text{ in } \mathbb{R}^{d+q}$$

+ A priori Error Estimates for u^m and P



Example: Darcy Flow

Problem:

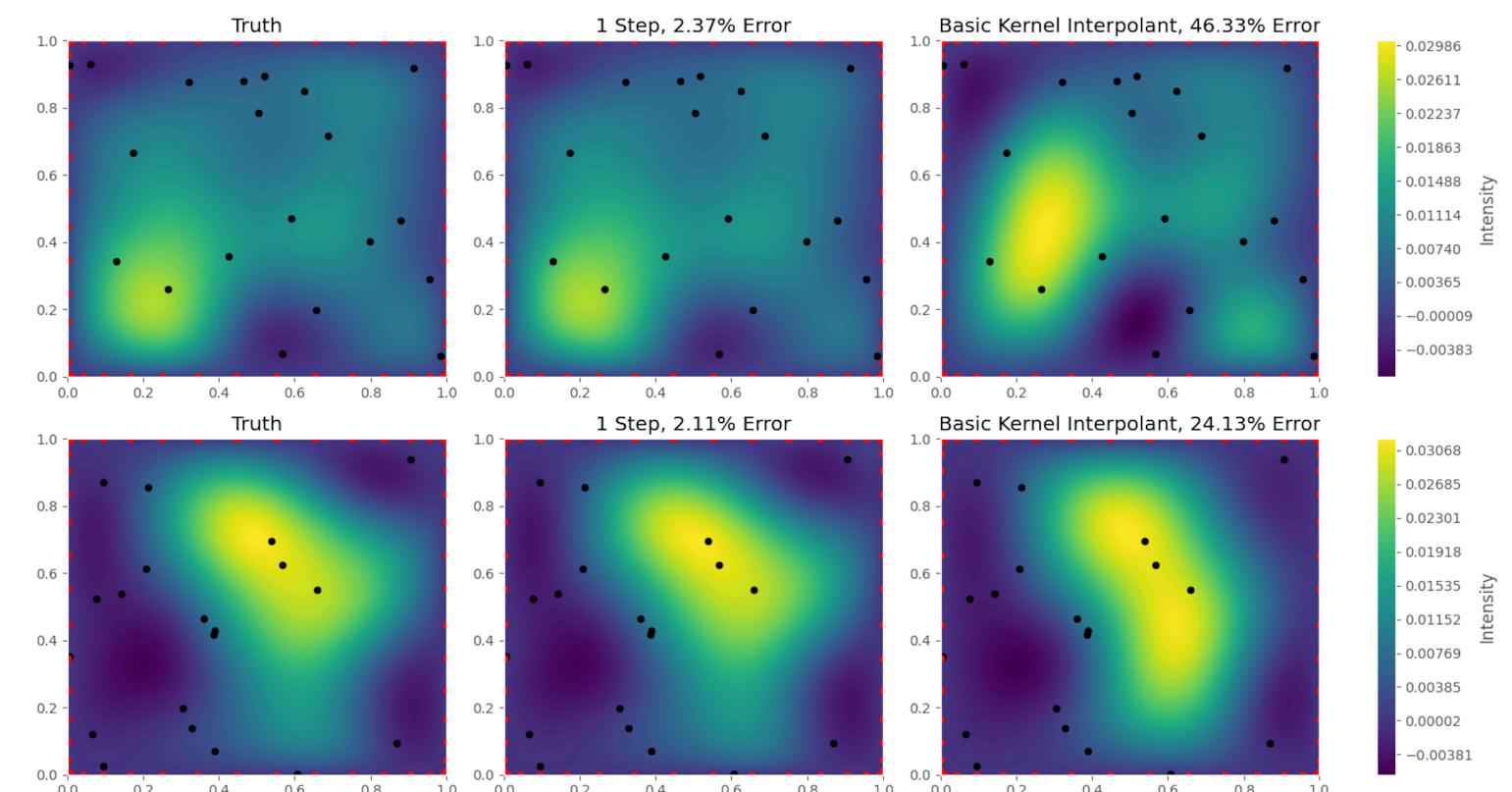
$$P(x, u(x), Du(x), D^2u(x)) = \nabla \cdot (a(x) \nabla u(x)) = f(x), \quad \forall x \in (0,1)^2$$

Data:

- $M \in \{10, 20, 30, 40, 50\}$ pairs of (u^m, f^m) , for $m \in \{1, \dots, M\}$
- prescribe functions $f^m \sim GP(0, K)$ and use solvers to get u^m
- 20×20 fine mesh of data points $X, N \in \{5, 10, 20\}$ observation points X_N^m

Objective:

learn P and u^m 's



50 pairs of functions on 20 observation points

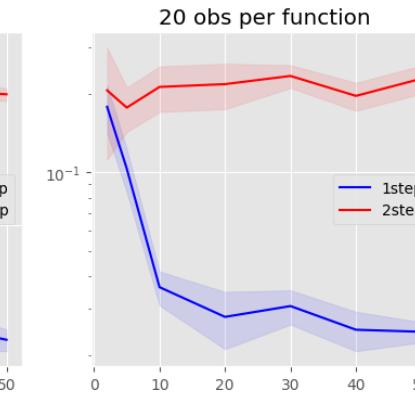
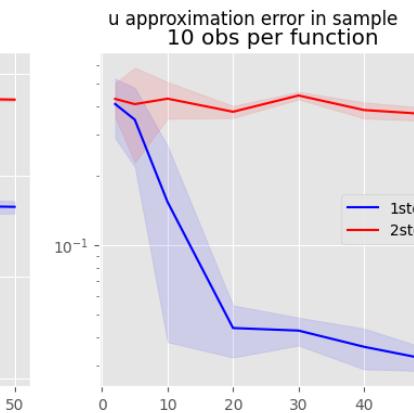
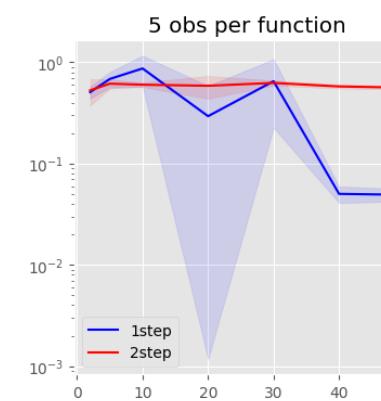


Example: Darcy Flow

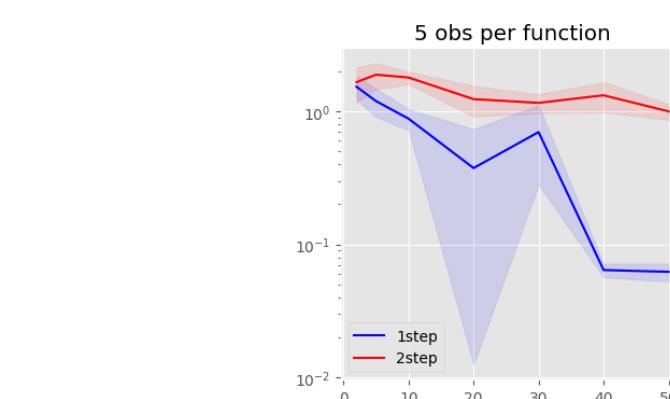
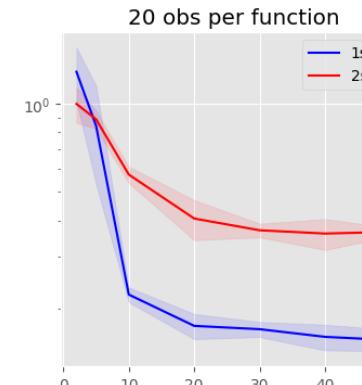
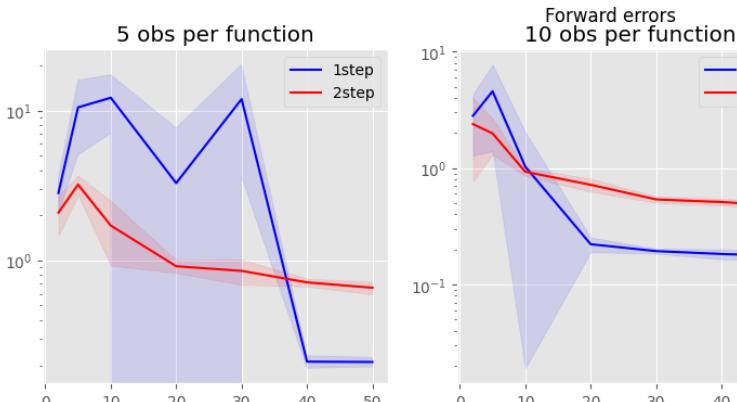
Problem:

$$P(x, u(x), Du(x), D^2u(x)) = \nabla \cdot (a(x) \nabla u(x)) = f(x), \quad \forall x \in (0,1)^2$$

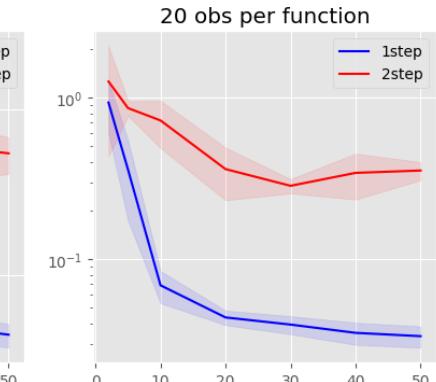
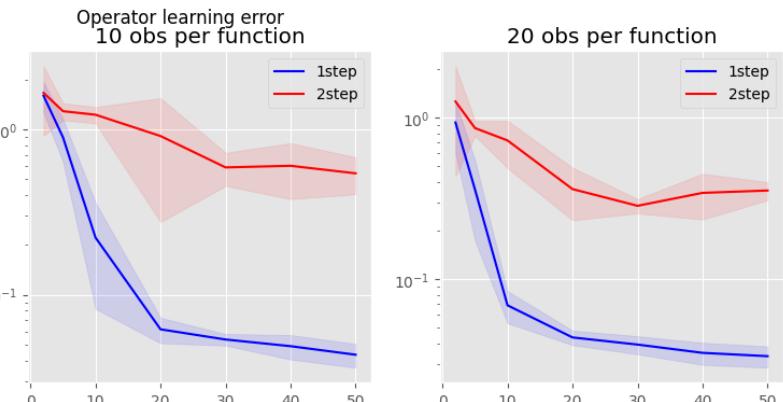
forward: $\frac{\|\hat{P}(u) - P(u)\|}{\|P(u)\|}$



in-sample: $\frac{\|\hat{u} - u\|}{\|u\|}$



operator learning: $\frac{\|u - \hat{P}^{-1}P(u)\|}{\|u\|}$



Example: Burgers' Equation

Problem:

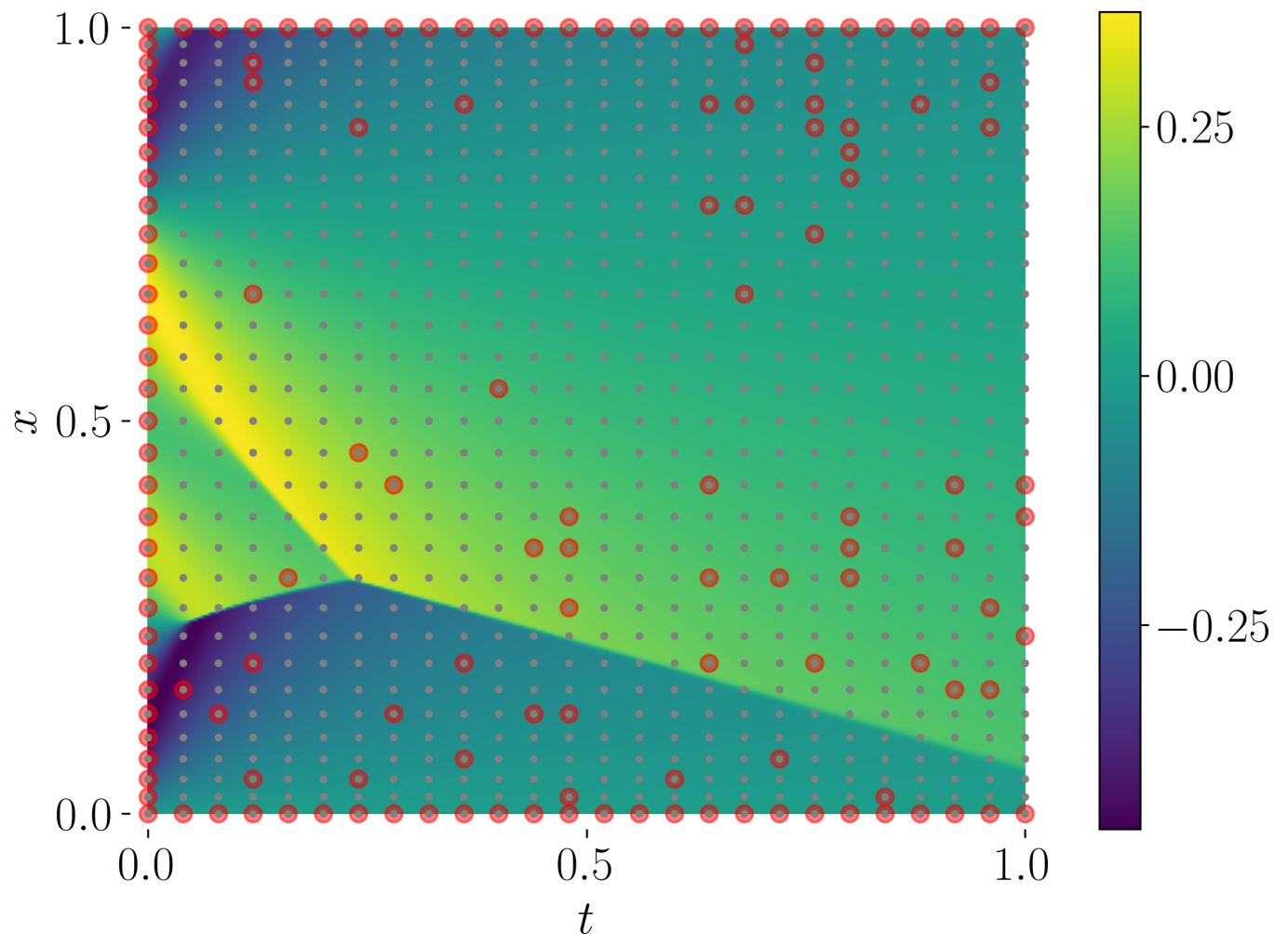
$$P(u)(y) = \partial_t u - 5u\partial_x u - 0.001\partial_{xx}u, \quad \forall y = (t, x) \in [0,1] \times (0,1)$$

Data:

- $M = 1$ pairs of (u^1, f^1)
- 806 data points $Y, N = 60$ observation points Y_N^1

Objective:

learn P and u^1

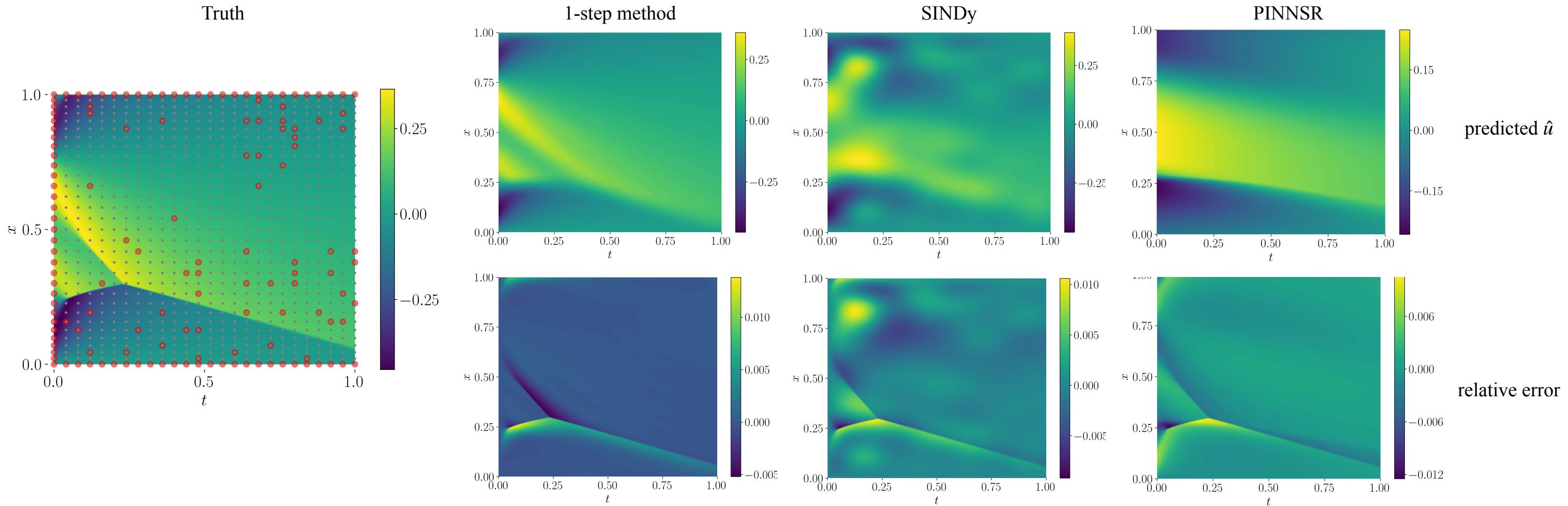




Example: Burgers' Equation

Problem:

$$P(u)(y) = \partial_t u - 5u\partial_x u - 0.001\partial_{xx}u, \quad \forall y = (t, x) \in [0,1] \times (0,1)$$



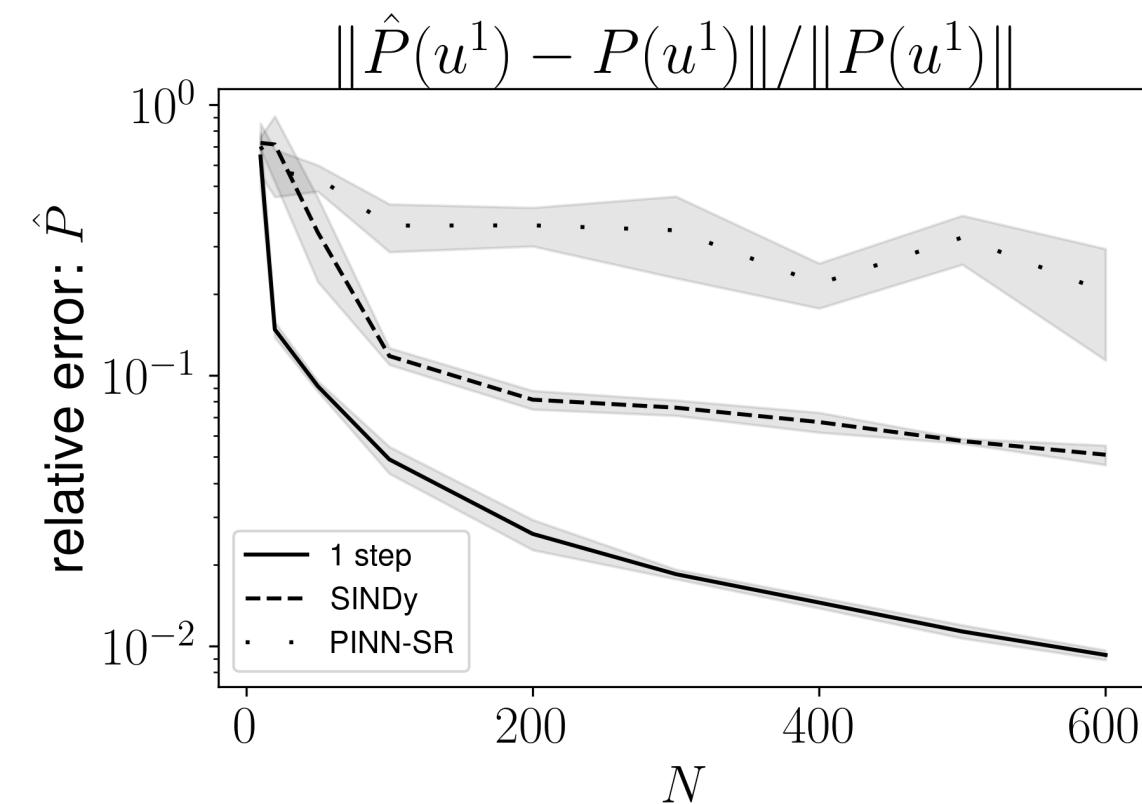
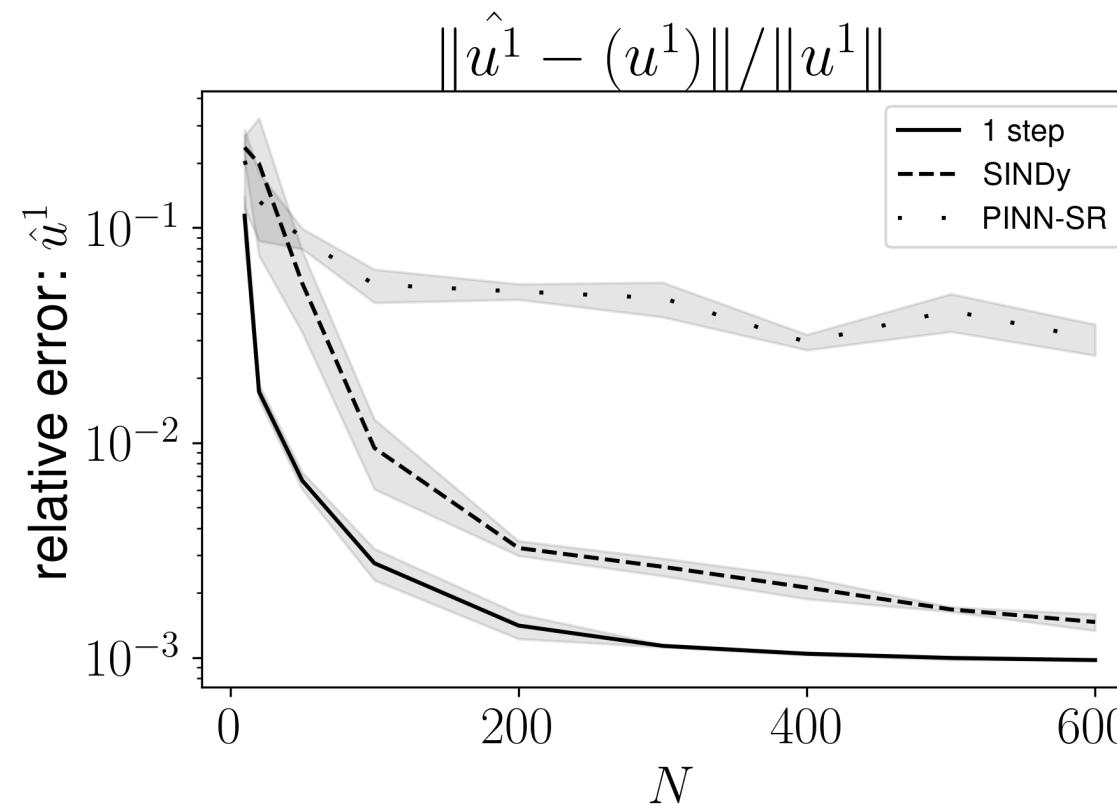


SEA-CROGS

Example: Burgers' Equation

Problem:

$$P(u)(y) = \partial_t u - 5u\partial_x u - 0.001\partial_{xx} u, \quad \forall y = (t, x) \in [0,1] \times (0,1)$$



Application: Jupiter's radiation environment

Collaboration with JPL Natural Space Environments Group

Problem:

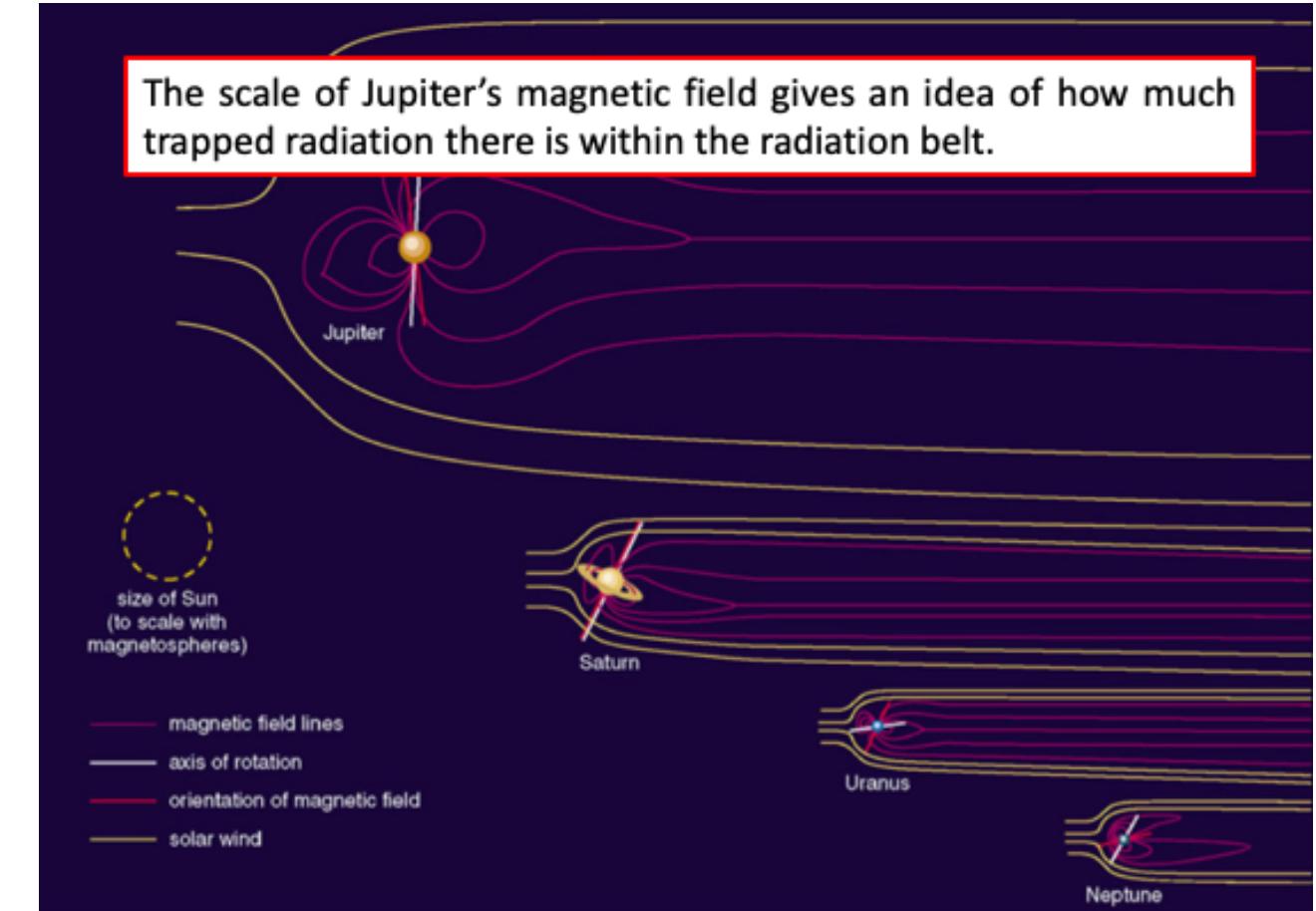
$$\partial_t f = \nabla \cdot (D \nabla f) + \nabla \cdot (cf)$$

Data:

Values of f at 6000 points in the domain

Objective:

learn f in the entire domain and learn the unknown coefficients c and D





Application: Stress Transfer & Earthquake Dynamics

Collaboration with USC Department of Chemical Engineering and Materials Science

Problem:

$$\nabla \cdot (G \nabla u_x) + \frac{G}{1-2\nu} \frac{\partial \epsilon_v}{\partial x} - \alpha \frac{\partial p}{\partial x} = 0$$

$$\nabla \cdot (G \nabla u_y) + \frac{G}{1-2\nu} \frac{\partial \epsilon_v}{\partial y} - \alpha \frac{\partial p}{\partial y} = 0$$

$$\nabla \cdot (G \nabla u_z) + \frac{G}{1-2\nu} \frac{\partial \epsilon_v}{\partial z} - \alpha \frac{\partial p}{\partial z} = \rho_b g$$

$$\frac{1}{M} \frac{\partial p}{\partial t} + \alpha \frac{\partial \epsilon_v}{\partial t} - \nabla \cdot \left(\frac{k}{\mu} \nabla p - \rho_f g \right) = 0$$

Data:

Values of u and p , each on 5 million points in the domain

Objective:

learn u and p in the entire domain and learn all the unknown coefficients

