Machine Learning for Partial Differential Equations: Solvers and Operator Learning

Matthieu Darcy[1]

[1]California Institute of Technology

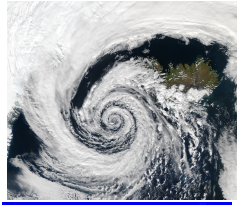
University of Bern - Institute of Mathematical Statistics and Actuarial Science
January, 2026

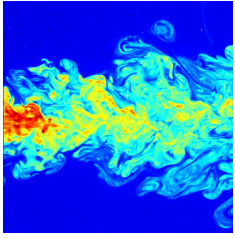
Table of Contents

Introduction: PDEs and Machine learning

Many physical phenomena are modeled by PDEs:

- Fluid mechanics
- Weather prediction
- Geophysics
- ...



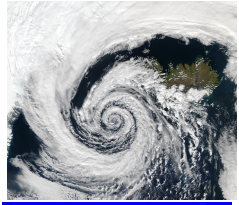


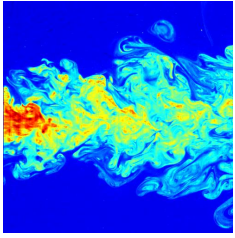
Introduction: PDEs and Machine learning

Many physical phenomena are modeled by PDEs:

- Fluid mechanics
- Weather prediction
- Geophysics
- ...

What are the current applications of machine learning in the context of PDEs?





Applications of machine learning to differential equations

What are the current applications of machine learning in the context of PDEs?

- **Solve differential equations:** provide a solution to the PDE.
- **Learn differential equations:** recover unknowns of the system or learn the functional form.
- **Accelerate computations:** replace or improve existing solvers.

Introduction: partial differential equations

Partial differential equations model systems and encode laws:

$$\underbrace{\frac{\partial v}{\partial t}}_{\text{Time evolution}} = \underbrace{\nabla \cdot (a \nabla v)}_{\text{Diffusion}} + \underbrace{R(v)}_{\text{Reaction}} + \underbrace{f}_{\text{External forcing}} + \dots$$

- v is the "quantity of interest" $\rightarrow$ **solution to the PDE**
- a is a property of the medium $\rightarrow$ **coefficient**
- ...

Introduction: partial differential equations

Partial differential equations model systems and encode laws:

$$\underbrace{\frac{\partial v}{\partial t}}_{\text{Time evolution}} = \underbrace{\nabla \cdot (a \nabla v)}_{\text{Diffusion}} + \underbrace{R(v)}_{\text{Reaction}} + \underbrace{f}_{\text{External forcing}} + \dots$$

- v is the "quantity of interest" $\rightarrow$ **solution to the PDE**
- a is a property of the medium $\rightarrow$ **coefficient**
- ...

Solving the PDE gives the solution v which describes the system, but...

- How do we solve the PDE?
- What if $a, f, \dots$ are unknown?
- What if solving the system is too expensive?
- What if we don't know the PDE to begin with?

Introduction: partial differential equations

Partial differential equations model systems and encode known laws. Abstractly:

$$\mathcal{F}(v; a)(x) = f(x) \quad x \in \Omega$$
$$\mathcal{B}(v; a)(x) = g(x) \quad x \in \partial\Omega$$

Introduction: partial differential equations

Partial differential equations model systems and encode known laws. Abstractly:

$$\mathcal{F}(v; a)(x) = f(x) \quad x \in \Omega$$
$$\mathcal{B}(v; a)(x) = g(x) \quad x \in \partial\Omega$$

Terminology:

$\mathcal{F}$ differential operator
$\mathcal{B}$ boundary operator

Unified as a single operator $\mathcal{F}$ (laws, conserved quantities...)

v solution (want to find)

a coefficient, f forcing, g boundary/initial condition... (data, properties...)

geometry?

...

Two classes of PDE problems

Forward problem: given $\mathcal{F}, a, f, g$...find v
Inverse problem: given $\mathcal{F}$ ($+$ partial information) find $v, a \ldots$
Learning differential equations: learn $\mathcal{F}$ from observations

} Solving and learning DEs

Machine learning solvers

- Gaussian processes [Chen, Hosseini, Owhadi, and Stuart 2021]
- Neural networks [Raissi, Perdikaris, and G. Karniadakis 2019]

parametrize a **single** function: solution v, coefficient a...

"Learn and predict a single system."

Two classes of PDE problems

Given an operator $\mathcal{S}$ associated to the PDE
and snapshots $(u_i, \mathcal{S}(u_i))_{i=1}^N \to$ learn $\mathcal{S}$
} Operator learning

Neural networks

- Fourier Neural operators [Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, and Anandkumar 2020]
- DeepONet [Lu, Jin, Pang, Zhang, and G. E. Karniadakis 2021]

parametrize an **operator** that can produce **many** solutions.

"Learn and predict across systems."

Example: Darcy flow

Example

$$-\nabla \cdot (a(x)\nabla v(x)) = f(x)$$
$$v(x) = g(x)$$

Forward problem: given a single $a, f, g \cdots$ recover v.
Inverse problem: given f, g and data on v, recover a, v

Example: Darcy flow

Example

$$-\nabla \cdot (a(x)\nabla v(x)) = f(x)$$
$$v(x) = g(x)$$

Forward problem: given a single $a, f, g \cdots$ recover v.
Inverse problem: given f, g and data on v, recover a, v

In this talk:
Operator learning problem: learn the solution operator $\mathcal{S} : (a, f, g \dots) \mapsto v$.

Table of Contents

Outline

Gaussian processes/kernels for operator learning

Relevant publication: Batlle, D, Hosseini, and Owhadi "Kernel methods are competitive for operator learning", *Journal of Computational Physics*, 2024

The operator learning problem

The operator learning problem (abstract version)

Let $\{u_i, v_i\}_{i=1}^N$ be N elements of $\mathcal{U} \times \mathcal{V}$ such that

$$\mathcal{S}(u_i) = v_i, \quad \text{for } i = 1, \dots, N\,.$$

The data driven operator learning problem is summarized as :

Given the data $\{u_i, v_i\}_{i=1}^N$ approximate $\mathcal{S}$.

$\mathcal{U}$ is a space of functions $u : \Omega \to \mathbb{R}$

$\mathcal{V}$ is a space of functions $v : D \to \mathbb{R}$

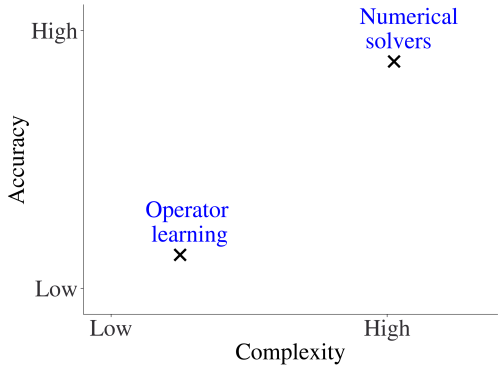
The functions u, v are *continuous, infinite* dimensional objects.

Applications of data-driven operator learning.

- **Problem 1**: unknown operator (unknown/incomplete physics, no model $\mathcal{F}$): learn from data. *Accuracy* is the relevant metric.

Applications of data-driven operator learning.

- **Problem 1**: unknown operator (unknown/incomplete physics, no model $\mathcal{F}$): learn from data. *Accuracy* is the relevant metric.
- **Problem 2** available but expensive model: accelerate computations. *Complexity-accuracy* is the relevant metric.



Operator learning

Past work has focused on Operator Neural Networks that generalize Neural Networks to functional inputs and outputs:

- FNO [Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, and Anandkumar 2020]
- PCA-net[Bhattacharya, Hosseini, Kovachki, and Stuart 2021]
- DeepOnet [Lu, Jin, Pang, Zhang, and G. E. Karniadakis 2021]
- ...

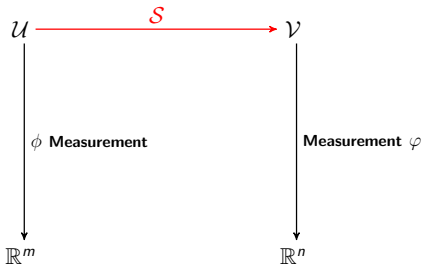
Our contribution

We propose a family of kernel/Gaussian process based-methods that are **simple, fast** and **competitive in accuracy.**

General framework

In practice we only have access to finite dimensional measurements of (u, v) given by $(\phi(u), \varphi(v))$

Encoder ϕ, φ



General framework

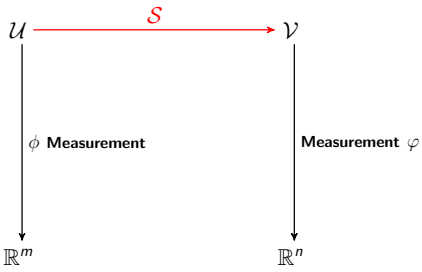
In practice we only have access to finite dimensional measurements of (u, v) given by $(\phi(u), \varphi(v))$

Encoder ϕ, φ

Common examples are:

$\phi : u \mapsto (u(x_1), \ldots, u(x_m))$ Pointwise values.

$\phi : u \mapsto (\langle u, e_1 \rangle, \ldots, \langle u, e_m \rangle)$ Projection in a basis (PCA/POD, Fourier...)



General framework

In practice we only have access to finite dimensional measurements of (u, v) given by $(\phi(u), \varphi(v))$

Encoder ϕ, φ

Common examples are:

$\phi : u \mapsto (u(x_1), \ldots, u(x_m))$ Pointwise values.

$\phi : u \mapsto (\langle u, e_1 \rangle, \ldots, \langle u, e_m \rangle)$ Projection in a basis (PCA/POD, Fourier...)

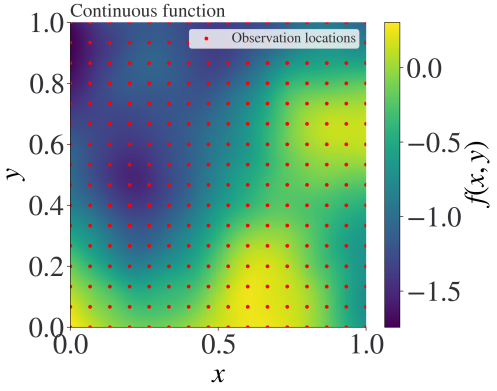


Figure: Continuum and observations

General framework

We have access to finite dimensional measurements of (u, v) given by $(\phi(u), \varphi(v))$:

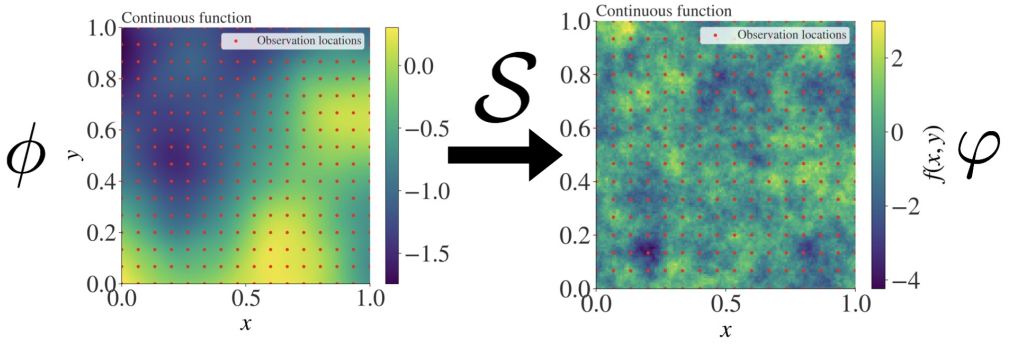
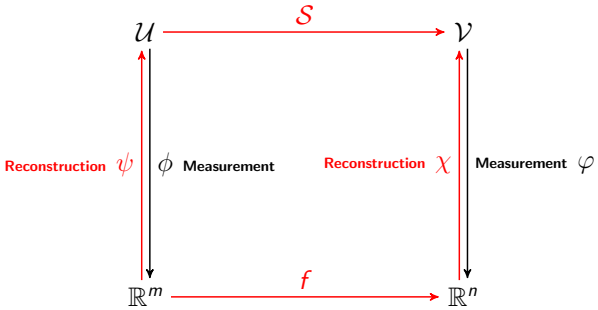


Figure: Continuum and observations

General framework

Encoder ϕ, φ

Decoder ψ, χ

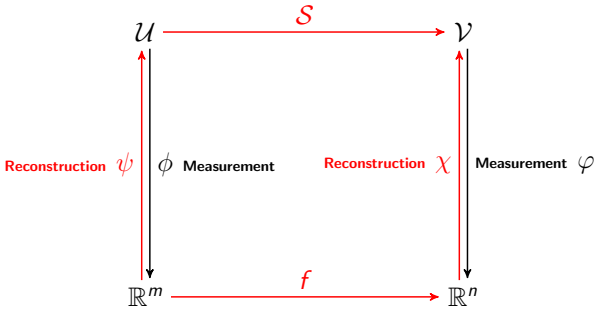


General framework

Encoder ϕ, φ

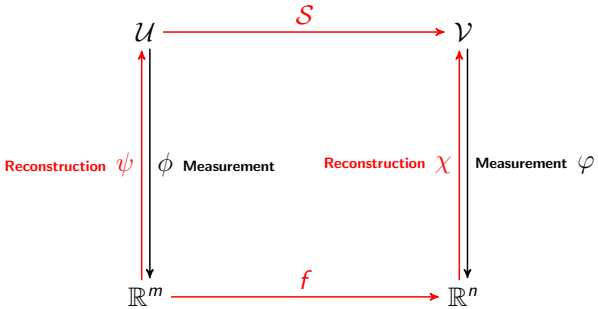
Decoder ψ, χ

Finite dimensional map $f := \varphi \circ \mathcal{S} \circ \psi$



General framework

Encoder	ϕ, φ
Decoder	ψ, χ
Finite dimensional map	$f := \varphi \circ \mathcal{S} \circ \psi$



Summary of our method

1. Define the reconstructions ψ and χ as optimal recovery maps.
2. Approximate the function f using a kernel method/GP (coordinate wise).

Step 1: Optimal recovery

We assume that $\mathcal{U}$ and $\mathcal{V}$ are RKHSs arising from kernels Q and K respectively. The reconstruction operators are defined as optimal recovery maps

$$\psi(\phi(u)) := \underset{w \in \mathcal{U}}{\arg\min} \|w\|_Q \quad \text{s.t.} \quad \phi(w) = \phi(u),$$
$$\chi(\varphi(v)) := \underset{w \in \mathcal{V}}{\arg\min} \|w\|_K \quad \text{s.t.} \quad \varphi(w) = \varphi(v),$$

The maps are the minmax optimal recovery of u and v respectively [Owhadi and Scovel].

Step 1: Optimal recovery

We assume that $\mathcal{U}$ and $\mathcal{V}$ are RKHSs arising from kernels Q and K respectively. The reconstruction operators are defined as optimal recovery maps

$$\psi(\phi(u)) := \underset{w \in \mathcal{U}}{\arg\min} \|w\|_Q \quad \text{s.t.} \quad \phi(w) = \phi(u),$$
$$\chi(\varphi(v)) := \underset{w \in \mathcal{V}}{\arg\min} \|w\|_K \quad \text{s.t.} \quad \varphi(w) = \varphi(v),$$

The maps are the minmax optimal recovery of u and v respectively [Owhadi and Scovel].
Optimal recovery maps can be expressed by solving a linear system

$$\psi(\phi(u))(x) = Q(x, X)Q(X, X)^{-1}\phi(u) \quad \text{and} \quad \chi(\varphi(v))(y) = K(y, Y)K(Y, Y)^{-1}\varphi(v).$$

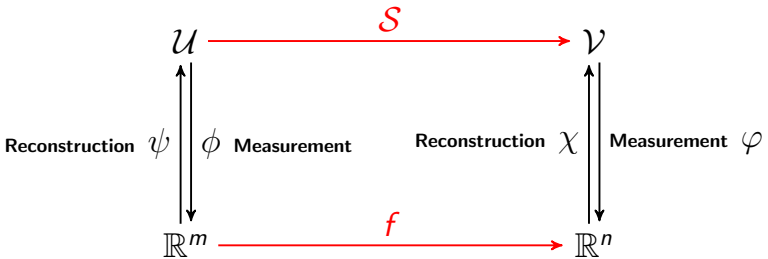
Step 2: Recovery of f

Once the reconstruction operators ψ and χ are defined, our best strategy is to reconstruct f in the diagram:

$$f^\dagger \approx f := \varphi \circ \mathcal{S} \circ \psi$$

and to approximate the operator $\mathcal{S}$ with the operator

$$\mathcal{S}^\dagger := \chi \circ f^\dagger \circ \phi\,.$$



A simple Gaussian process method for $f^\dagger$

Given a kernel S, we approximate $f : \mathbb{R}^m \to \mathbb{R}^n$ via optimal recovery **independently component wise**:

$$f_j^\dagger := \underset{h \in \mathcal{H}_S}{\arg\min} \|h\|_S \quad \text{s.t.} \qquad h(\phi(u_i)) = (\varphi(v_i))_j \qquad \text{for} \quad i = 1, \ldots, N.$$

A simple Gaussian process method for $f^\dagger$

Given a kernel S, we approximate $f : \mathbb{R}^m \to \mathbb{R}^n$ via optimal recovery **independently component wise**:

$$f_j^\dagger := \operatorname*{arg\,min}_{h \in \mathcal{H}_S} \|h\|_S \quad \text{s.t.} \qquad h(\phi(u_i)) = (\varphi(v_i))_j \qquad \text{for} \quad i = 1, \ldots, N.$$

This also has a closed-form solution given a linear solve:

$$f_j^\dagger(\boldsymbol{u}) = S(\boldsymbol{u}, U) S(U, U)^{-1} \boldsymbol{v}_j.$$

where $U_i := \phi(u_i)$ and $V_i := \varphi(v_i)$.
Can be solved in parallel (or even better).

Measurement invariance

Mesh invariance: translate between meshes.

$$h := \tilde{\varphi} \circ \chi \circ f \circ \phi \circ \tilde{\psi} \equiv \tilde{\varphi} \circ \mathcal{S} \circ \tilde{\psi}.$$

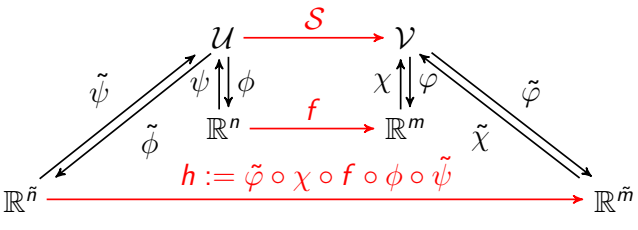


Figure: Mesh invariance of the method.

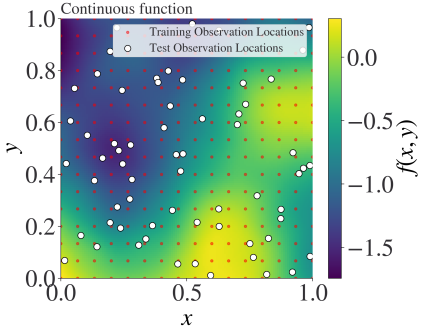


Figure: Change of mesh

Summary of the method

Summary of our method

1. Define the reconstructions ψ and χ as the optimal recovery map (linear solve).
2. Approximate the function f using a GP/kernel method.

Key takeways

- **Simplicity:** simplifying assumptions (independence of steps 1 and 2, coordinate wise regression of $f \dots$) for improved complexity.
- **UQ:** Gaussian process interpretation provides UQ.
- **Theory:** convergence guarantees (new for operator learning).
- **Experimentally:** competitive in accuracy and cost/accuracy (low-medium data regime).

Summary of results: accuracy

	Low-data regime			High-data regime			
	Burger's	**Darcy problem**	**Advection I**	**Advection II**	**Hemholtz**	**Structural Mechanics**	**Navier Stokes**
DeepONet	2.15%	2.91%	0.66%	15.24%	5.88%	5.20%	3.63%
POD-DeepONet	1.94%	2.32%	0.04%	n/a	n/a	n/a	n/a
FNO	1.93%	2.41%	0.22%	13.49%	1.86%	4.76%	0.26%
PCA-Net	n/a	n/a	n/a	12.53%	2.13%	4.67%	2.65%
PARA-Net	n/a	n/a	n/a	16.64%	12.54%	4.55%	4.09%
Linear	36.24%	6.74%	2.15×10^{-13}%	11.28%	10.59%	27.11%	5.41%
Best of Matérn/RQ	2.15%	2.75%	2.75×10^{-3}%	11.44%	1.00%	5.18%	0.12%

Table: Summary of numerical results: L^2 relative test error.

We compare the performance against several neural operator frameworks on a range of problems taken from two benchmarking studies [Hoop, Huang, Qian, and Stuart 2022, Lu, Meng, Cai, Mao, Goswami, Zhang, and G. E. Karniadakis 2022] and find competitive performance.

Experimental results: Navier-Stokes

Vorticity-stream formulation of the incompressible Navier-Stokes equations:

$$\begin{aligned} \frac{\partial v}{\partial t} + (w \cdot \nabla) v - \nu \Delta v &= f \\ v &= -\Delta \psi \\ \int_D \psi &= 0 \\ w &= \left(\frac{\partial \psi}{\partial x_2}, -\frac{\partial \psi}{\partial x_1} \right) \end{aligned}$$

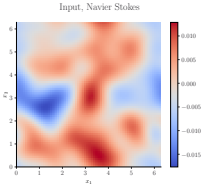
Learn the map from the forcing term f to the vorticity field v at $t = T$:

$$\mathcal{S} : f \mapsto v(\cdot, T)$$

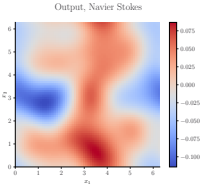
Navier-Stokes

Method	Accuracy
DeepONet	3.63 %
FNO	0.26 %
PCA-Net	2.32 %
Linear method	5.41 %
Kernel method	0.12%

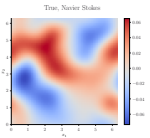
Table: L^2 relative error on Navier-Stokes.



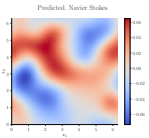
(a) Input



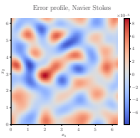
(b) Output



(c) True



(d) Predicted



(e) Error

Empirical results

Competitive in complexity-accuracy:

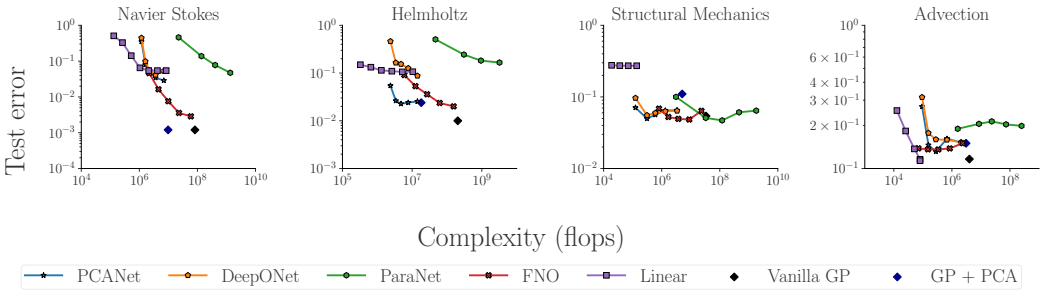


Figure: Cost-accuracy tradeoff at prediction.

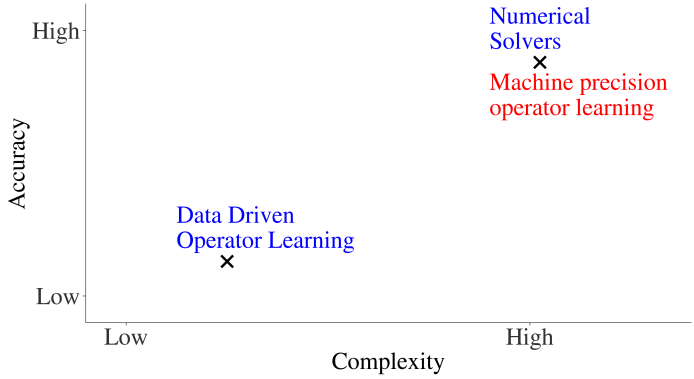
Outline

High accuracy operator learning

- The presented framework is well suited for **Problem 1**: unintrusive, doesn't use any information about $\mathcal{S}$.

High accuracy operator learning

- The presented framework is well suited for **Problem 1**: unintrusive, doesn't use any information about $\mathcal{S}$.
- **Problem 2**, can operator learning match the accuracy of traditional solvers?



Operator learning at machine precision

If we know the underlying PDE operator $\mathcal{F}$ and have access to an underlying solver, can we achieve the same high/machine precision?

Operator learning at machine precision

If we know the underlying PDE operator $\mathcal{F}$ and have access to an underlying solver, can we achieve the same high/machine precision?

We propose CHONKORIS:a data-driven operator learning framework, that learns to emulate a (quasi)-Newton solvers and can achieve machine precision.

Relevant publication: Bacho, Sorokin, Yang, Bourdais, Calvello, D, Hsu, Hosseini, and Owhadi *Operator Learning at Machine Precision*, 2025

Operator learning at machine precision

Observation: the PDE defines a constraint for $\mathcal{S}$:

$$\mathcal{F}(\mathcal{S}(u), u) = 0$$

Operator learning at machine precision

Observation: the PDE defines a constraint for $\mathcal{S}$:

$$\mathcal{F}(\mathcal{S}(u), u) = 0$$

Ex:

$$\mathcal{F}(\mathcal{S}(u), u) = -\nabla \cdot (u \nabla v) + \tau(v) = 0$$
$$\mathcal{S}(u) = v \quad \text{(Coefficient to solution map)}$$

Operator learning at machine precision

Observation: the PDE defines a constraint for $\mathcal{S}$:

$$\mathcal{F}(\mathcal{S}(u), u) = 0$$

Ex:

$$\begin{aligned} \mathcal{F}(\mathcal{S}(u), u) &= -\nabla \cdot (u \nabla v) + \tau(v) = 0 \\ \mathcal{S}(u) &= v \quad \text{(Coefficient to solution map)} \end{aligned}$$

Observation: We can solve this using iterative methods (Newton-Raphson)

$$v_{n+1} = v_n - \left(\frac{\delta}{\delta v} \mathcal{F}(v_n, u) \right)^{-1} \mathcal{F}(v_n, u)$$

Key idea: it is sufficient to learn $\mathcal{S}(v, u) = \left(\frac{\delta}{\delta v} \mathcal{F}(v, u) \right)^{-1}$

Operator learning at machine precision

High level idea: the optimization algorithm defines an operator $\mathcal{S}$

$$u \text{ (PDE data)} \xrightarrow{\text{Newton flow } \mathcal{S}} v \quad \text{(output of the algorithm)}$$

Traditional operator learning learns the entire flow $\mathcal{S}$

Operator learning at machine precision

High level idea: the optimization algorithm defines an operator $\mathcal{S}$

$$u \text{ (PDE data)} \xrightarrow{\text{Newton flow } \mathcal{S}} v \quad \text{(output of the algorithm)}$$

Traditional operator learning learns the entire flow $\mathcal{S}$

$$(v_0, u) \xrightarrow{\text{Iterate } \mathcal{R}} (v_1, u) \xrightarrow{\text{Iterate } \mathcal{R}} \dots (v_n, u)$$

CHONKNORIS learns the iterates of the flow $\mathcal{R}$.

CHONKNORIS

Key takeways:

- Achieves machine precision (solver accuracy) on many problems
- In practice use Gauss-Newton (regularized Newton).
- Can be combined with previous operator learning approaches for the initial guess.
- Enables a foundation model that can generalize to unseen PDEs.

CHONKNORIS

Key takeways:

- Achieves machine precision (solver accuracy) on many problems
- In practice use Gauss-Newton (regularized Newton).
- Can be combined with previous operator learning approaches for the initial guess.
- Enables a foundation model that can generalize to unseen PDEs.

Limitations

- Intrusive: requires a solver and its intermediate states.
- Still slower than the original solver.
- Limited to low dimensional examples (forming and inverting the Hessian is bad in high dimensions).
- Inverse problems are harder du to ill-conditioning.

Example

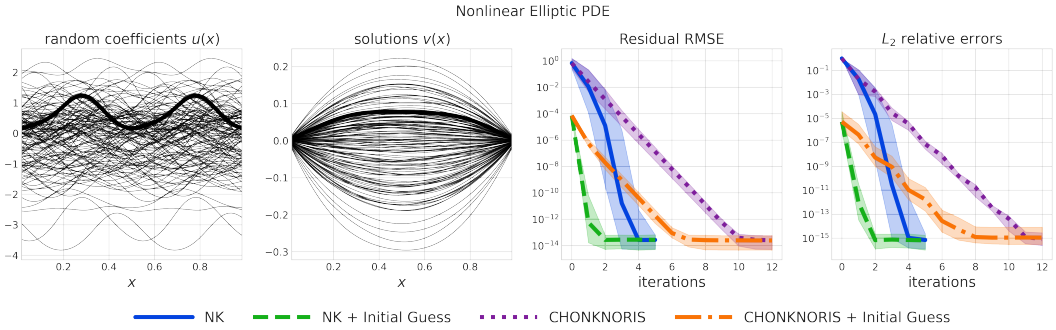


Figure: Nonlinear elliptic PDE: $10^{-5} \to 10^{-15}$ error

Summary and outlook

We proposed two methods for operator learning:

- A general purpose Gaussian process approach for inintrusive, low-accuracy data-driven operator learning.
- An intrusive operator learning framework for high-accuracy operator learning.

Summary and outlook

We proposed two methods for operator learning:

- A general purpose Gaussian process approach for inintrusive, low-accuracy data-driven operator learning.
- An intrusive operator learning framework for high-accuracy operator learning.

Outlook and future work:

- Is it possiblle to adress operator learning between **Problem 1** and **Problem 2**, i.e. partial knowledge of the physics (conserved quantities, mathematical properties...).
- How can we include geometry?
- Foundation models for PDEs: how doe we enable cross-pde learning at scale?
- Can we do this while preserving convergence guarantees?

Questions? Comments?

(a) Operator learning with Gaussian processes

(b) Machine precision operator learning

References I

- Bacho, Aras et al. [2025]. *Operator Learning at Machine Precision*. arXiv: 2511.19980 [cs.LG]. URL: https://arxiv.org/abs/2511.19980.
- Baptista, Ricardo et al. [2025]. *Solving Roughly Forced Nonlinear PDEs via Misspecified Kernel Methods and Neural Networks*. arXiv: 2501.17110 [math.NA]. URL: https://arxiv.org/abs/2501.17110.
- Batlle, Pau et al. [2024]. "Kernel methods are competitive for operator learning". In: *Journal of Computational Physics* 496.
- Bhattacharya, Kaushik et al. [2021]. "Model Reduction And Neural Networks For Parametric PDEs". en. In: *The SMAI Journal of computational mathematics* 7, pp. 121–157. DOI: 10.5802/smai-jcm.74. URL: https://smai-jcm.centre-mersenne.org/articles/10.5802/smai-jcm.74/.
- Chen, Yifan et al. [2021]. "Solving and learning nonlinear PDEs with Gaussian processes". In: *Journal of Computational Physics* 447, p. 110668. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2021.110668. URL: https://www.sciencedirect.com/science/article/pii/S0021999121005635.

References II

- D, Matthieu et al. [2023]. "One-shot learning of stochastic differential equations with data adapted kernels". In: *Physica D: Nonlinear Phenomena* 444, p. 133583. ISSN: 0167-2789. DOI: https://doi.org/10.1016/j.physd.2022.133583. URL: https://www.sciencedirect.com/science/article/pii/S0167278922002871.
- Hoop, Maarten V. de et al. [2022]. *The Cost-Accuracy Trade-Off In Operator Learning With Neural Networks*.
- Li, Zongyi et al. [2020]. "Fourier Neural Operator for Parametric Partial Differential Equations". In: *CoRR* abs/2010.08895. arXiv: 2010.08895. URL: https://arxiv.org/abs/2010.08895.
- Lu, Lu et al. [Mar. 2021]. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature Machine Intelligence* 3.3, pp. 218–229.

References III

- Lu, Lu et al. [2022]. "A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data". In: *Computer Methods in Applied Mechanics and Engineering* 393, p. 114778. ISSN: 0045-7825.
- Owhadi, Houman and Clint Scovel [2019]. *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.
- Raissi, M., P. Perdikaris, and G.E. Karniadakis [2019]. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378, pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Table of Contents

Theory: assumptions

Suppose that

$$\mathcal{U} \text{ is an RKHS of functions } u : \Omega \to \mathbb{R}$$
$$\mathcal{V} \text{ is an RKHS of functions } v : D \to \mathbb{R}.$$

Assumption (Reconstruction operators - informal)

- *Regularity of the domains Ω and D: compactness and Lipschitz boundary.*
- *Regularity of the kernels Q and K.*
- *Space filling property of collocation points.*

Assumption (Approximation of $\mathcal{S}$ - informal)

- *Regularity of the operator $\mathcal{S}$: continuity and bounded Fréchet derivatives.*
- *Regularity of the kernels S^n.*
- *Resolution and space-filling property of the data.*

Theory: convergence result

Under a set of technical assumptions, we have the following theorem:

Theorem (Condensed version of Main Theorem)

For all $t' \in (0, t)$,

$$\lim_{n,m\to\infty} \lim_{N\to\infty} \sup_{u\in B_R(\mathcal{H}_Q)} \|\mathcal{S}(u) - \chi^m \circ f_N^{\dagger,m,n} \circ \phi^n(u)\|_{H^{t'}(D)} \to 0\,.$$

Table of Contents

Outline

Learning SODEs

Relevant publication: D, Hamzi, Livieri, Owhadi, and Tavallali "One-shot learning of stochastic differential equations with data adapted kernels", *Physica D: Nonlinear Phenomena*, 2023

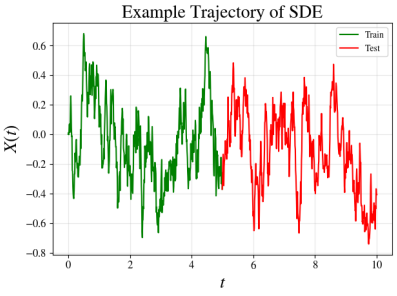
Learning SDEs

The problem

Learn the unknown drift f and the diffusion σ of a generic SODE:

$$dX_t = f(X_t)\,dt + \sigma(X_t)\,dW_t$$

from a finite number of samples $X := (X_{t_n})_{n=1}^{N}$.
Learn the evolution of the system from samples.



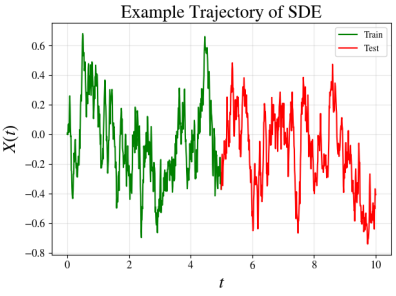
Learning SDEs

The problem

Learn the unknown drift f and the diffusion σ of a generic SODE:

$$dX_t = f(X_t)\,dt + \sigma(X_t)\,dW_t$$

from a finite number of samples $X := (X_{t_n})_{n=1}^{N}$.
Learn the evolution of the system from samples.



The difficulty

- The observations X come from a *single* trajectory with N *small* (sparse data).
- We want a flexible method: few assumptions on f and σ.
- A large sampling time-steps Δt introduces a discretization error.

Summary of the method

Our approach

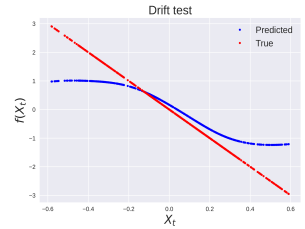
1. Place a Gaussian Process prior on f, σ.
2. Formulate and solve an optimization problem using Maximum A Posteriori (MAP) Estimation.
3. Optimize hyper-parameters using cross-validation $\rightarrow$ enables generalization in sparse data regimes!

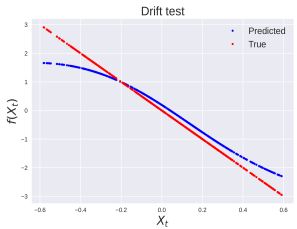
Example

Non learned kernel

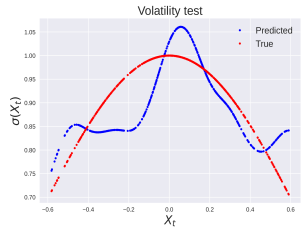
Learned kernel

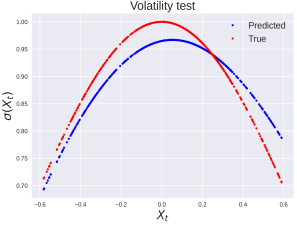
Drift f





Diffusion σ





Learning the right kernel makes a big difference in performance!

Outline

Solving PDEs with stochasticity and high frequency

Relevant publication: Baptista, Calvello, D, Owhadi, Stuart, and Yang *Solving Roughly Forced Nonlinear PDEs via Misspecified Kernel Methods and Neural Networks*, 2025

Solving rough PDEs

The problem

We want to **develop machine learning solvers for rough PDEs** of the form

$$\begin{aligned} \mathcal{F}(v) &= \xi, \quad x \in \Omega, \\ v &= 0, \quad x \in \partial\Omega \end{aligned} \tag{RPDE}$$

Solving rough PDEs

The problem

We want to **develop machine learning solvers for rough PDEs** of the form

$$\begin{aligned} \mathcal{F}(v) &= \xi, \quad x \in \Omega, \\ v &= 0, \quad x \in \partial\Omega \end{aligned}$$

The difficulty

Solving (RPDE) is difficult because the forcing term is 'rough':

- ξ is stochastic/high frequency $\to \xi \in H^{-s} \neq L^2$
- v has low regularity.

Machine learning solvers struggle to capture high frequency.

Summary of the method

We modify the loss to adapt to the roughness/high frequency:

$$\arg\min_{u} \|\mathcal{F}(u) - \xi\|_{L^2}^2 + ||u||_{L^2(\partial\Omega)}^2 + \gamma\mathcal{R}(u) \qquad \text{Physics Informed Loss}$$

Summary of the method

We modify the loss to adapt to the roughness/high frequency:

$$\arg\min_u \|\mathcal{F}(u) - \xi\|^2_{L^2} + ||u||^2_{L^2(\partial\Omega)} + \gamma\mathcal{R}(u) \qquad \text{Physics Informed Loss}$$

$$\Downarrow$$

$$\arg\min_u \|\mathcal{F}(u) - \xi\|^2_{H^{-s}} + ||u||^2_{L^2(\partial\Omega)} + \gamma\mathcal{R}(u) \qquad \text{Negative Sobolev Loss}$$

Summary of the method

We modify the loss to adapt to the roughness/high frequency:

$$\arg\min_u \|\mathcal{F}(u) - \xi\|^2_{L^2} + ||u||^2_{L^2(\partial\Omega)} + \gamma\mathcal{R}(u) \qquad \text{Physics Informed Loss}$$

$$\Downarrow$$

$$\arg\min_u \|\mathcal{F}(u) - \xi\|^2_{H^{-s}} + ||u||^2_{L^2(\partial\Omega)} + \gamma\mathcal{R}(u) \qquad \text{Negative Sobolev Loss}$$

$$\Downarrow$$

$$\arg\min_u |\mathcal{F}(u) - \xi|^2_{\Phi^N} + \sum_{i=1}^{m} |u(x_i)|^2 + \gamma\mathcal{R}(u) \qquad \text{Discretized loss}$$

Illustration of results

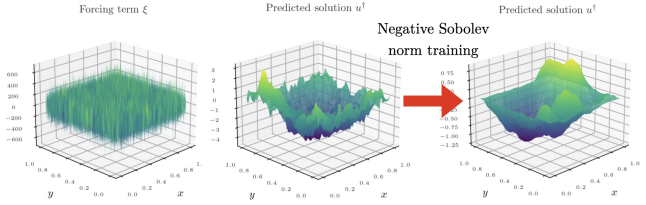


Figure: Negative Sobolev training

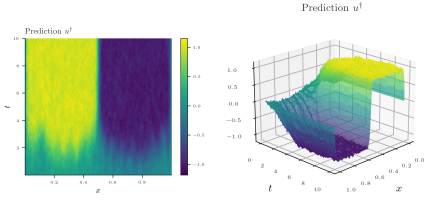


Figure: Time dependent: Stochastic Allen-Cahn equation.

Key takeways

Theoretical contributions:

- Proposes a negative Sobolev norm (related to weak solutions).
- Prove the convergence of the kernel method (even with misspecification).

Computational contributions:

- Provide an efficient numerical discretization of the Sobolev norm.
- Provide an efficient numerical method to minimize the resulting loss.
- Demonstrate the framework with neural networks and kernels.

Summary and outlook

I have worked on two learning and solving differential equations involving stochasticity:

- Learning SDEs from sparse data.
- Solving PDEs driven by rough forcing.

Summary and outlook

I have worked on two learning and solving differential equations involving stochasticity:

- Learning SDEs from sparse data.
- Solving PDEs driven by rough forcing.

Outlook and future work:

- Can we use some of these techniques for operator learning in sparse data regime
- Can we use some of these insights for machine learning in the presence of roughness/shocks/discontinuity?
- Can we provide better probabilistic predictions and more scalable approaches for GPs (work in progress).

Optimal recovery

We will assume that $\mathcal{U}$ and $\mathcal{V}$ are RKHSs arising from kernels Q and K respectively. The reconstruction operators are defined as optimal recovery maps

$$\psi(\phi(u)) := \arg\min_{w \in \mathcal{U}} \|w\|_Q \quad \text{s.t.} \quad \phi(w) = \phi(u),$$
$$\chi(\varphi(v)) := \arg\min_{w \in \mathcal{V}} \|w\|_K \quad \text{s.t.} \quad \varphi(w) = \varphi(v),$$

The maps are the minmax optimal recovery of u and v respectively [Owhadi and Scovel].

Optimal recovery

We will assume that $\mathcal{U}$ and $\mathcal{V}$ are RKHSs arising from kernels Q and K respectively. The reconstruction operators are defined as optimal recovery maps

$$\psi(\phi(u)) := \underset{w \in \mathcal{U}}{\arg\min} \|w\|_Q \quad \text{s.t.} \quad \phi(w) = \phi(u),$$
$$\chi(\varphi(v)) := \underset{w \in \mathcal{V}}{\arg\min} \|w\|_K \quad \text{s.t.} \quad \varphi(w) = \varphi(v),$$

The maps are the minmax optimal recovery of u and v respectively [Owhadi and Scovel].
Optimal recovery maps can be expressed by solving a linear system

$$\psi(\phi(u))(x) = Q(x, X)Q(X, X)^{-1}\phi(u) \quad \text{and} \quad \chi(\varphi(v))(y) = K(y, Y)K(Y, Y)^{-1}\varphi(v).$$

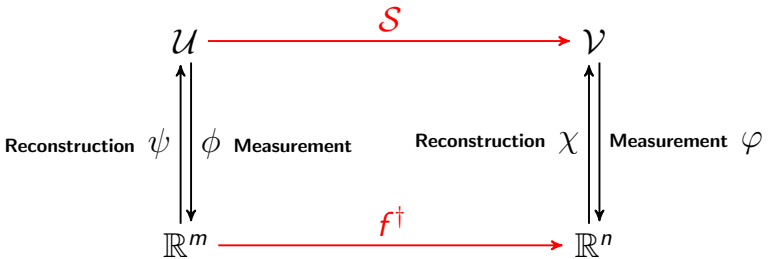
Recovery of $f^\dagger$

Once the reconstruction operators ψ and χ are defined, our best strategy is to reconstruct $f^\dagger$ in the diagram:

$$\bar{f} \approx f^\dagger := \varphi \circ \mathcal{S} \circ \psi$$

and to approximate the operator $\mathcal{S}$ with the operator

$$\bar{\mathcal{S}} := \chi \circ \bar{f} \circ \phi\,.$$



A simple kernel method for $f^\dagger$

Given a kernel S, we approximate $f : \mathbb{R}^m \to \mathbb{R}^n$ via optimal recovery **independently component wise**:

$$\bar{f}_j := \underset{h \in \mathcal{H}_S}{\arg\min} \|h\|_S \quad \text{s.t.} \qquad h(\phi(u_i)) = (\varphi(v_i))_j \qquad \text{for} \quad i = 1, \ldots, N.$$

A simple kernel method for $f^\dagger$

Given a kernel S, we approximate $f : \mathbb{R}^m \to \mathbb{R}^n$ via optimal recovery **independently component wise**:

$$\bar{f}_j := \underset{h \in \mathcal{H}_S}{\arg\min} \|h\|_S \quad \text{s.t.} \qquad h(\phi(u_i)) = (\varphi(v_i))_j \qquad \text{for} \quad i = 1, \ldots, N.$$

This also has a closed-form solution given a linear solve:

$$\bar{f}_j(\boldsymbol{u}) = S(\boldsymbol{u}, U) S(U, U)^{-1} \boldsymbol{v}_j.$$

where $U_i := \phi(u_i)$ and $V_i := \varphi(v_i)$.
Can be solved in parallel (or even better).

Advantages the simple approach

The kernel S can be a standard kernel such as the linear[1], squared exponential or Matérn kernel. This simple choice already offers several advantages:

1. Low cost in training (< 5 seconds on a workstation) and at prediction (in the low-medium data regime).
2. Competitive accuracy.
3. Empirically robust to choice of hyper-parameters/kernels.
4. Simple to implement: several libraries solve this problem out of the box.
5. The Gaussian process interpretation provides uncertainty quantification.
6. Convergence guarantees.

If you do operator learning, you should try using Gaussian processes!

[1] Equivalent to doing linear regression

Complexity-accuracy tradeoff

We evaluate our operator learning methods through the **cost-accuracy tradeoff**:

- The **accuracy** is measured by the relative L^2 loss:

$$\mathcal{R}_N(\mathcal{S}) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{\left\|\mathcal{S}(u_i) - \mathcal{S}^\dagger(u_i)\right\|_{L^2}}{\|\mathcal{S}(u_i)\|_{L^2}}\right]$$

- The **complexity** depends on the **training cost** (qualitative metrics) and the **prediction cost** (measured in floating point operations - FLOPs).

We compare the test performance of our method with different choices of the kernel S using the examples from two comparison papers [Hoop, Huang, Qian, and Stuart], [Lu, Meng, Cai, Mao, Goswami, Zhang, and G. E. Karniadakis]

Navier-Stokes

Vorticity-stream formulation of the incompressible Navier-Stokes equations:

$$\begin{aligned} \frac{\partial v}{\partial t} + (v \cdot \nabla) v - \nu \Delta v &= u \\ v &= -\Delta \psi \\ \int_D \psi &= 0 \\ v &= \left(\frac{\partial \psi}{\partial x_2}, -\frac{\partial \psi}{\partial x_1} \right) \end{aligned}$$

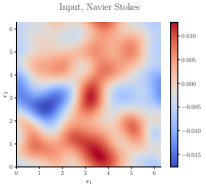
Learn the map from the forcing term f to the vorticity field v at $t = T$:

$$\mathcal{S} : u \mapsto v(\cdot, T)$$

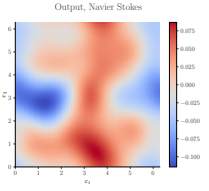
Navier-Stokes

Method	Accuracy
DeepONet	3.63 %
FNO	0.26 %
PCA-Net	2.32 %
Linear method	5.41 %
Kernel method	0.12%

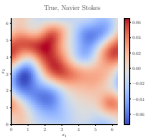
Table: L^2 relative error on Navier-Stokes.



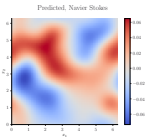
(a) Input



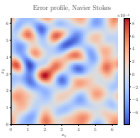
(b) Output



(c) True



(d) Predicted



(e) Error

Accuracy-complexity

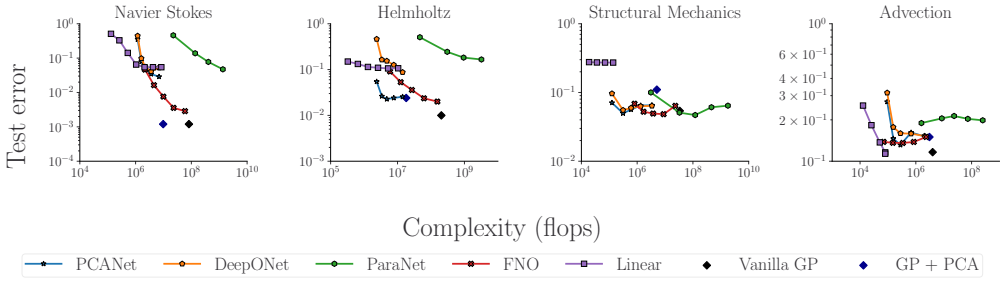


Figure: Cost-accuracy tradeoff at prediction for different operator learning frameworks.

Data taken from Hoop, Huang, Qian, and Stuart 2022.

Theory: convergence result

Under a set of technical assumptions, we have the following theorem:

Theorem (Condensed version of Main Theorem)

For all $t' \in (0, t)$,

$$\lim_{n,m\to\infty} \lim_{N\to\infty} \sup_{u\in B_R(\mathcal{H}_Q)} \|\mathcal{S}(u) - \chi^m \circ \bar{f}_N^{m,n} \circ \phi^n(u)\|_{H^{t'}(D)} \to 0\,.$$

Uses the theory of kernel interpolation + reconstruction operators.

Summary of results: accuracy

	Low-data regime			High-data regime			
	Burger's	**Darcy problem**	**Advection I**	**Advection II**	**Hemholtz**	**Structural Mechanics**	**Navier Stokes**
DeepONet	2.15%	2.91%	0.66%	15.24%	5.88%	5.20%	3.63%
POD-DeepONet	1.94%	2.32%	0.04%	n/a	n/a	n/a	n/a
FNO	1.93%	2.41%	0.22%	13.49%	1.86%	4.76%	0.26%
PCA-Net	n/a	n/a	n/a	12.53%	2.13%	4.67%	2.65%
PARA-Net	n/a	n/a	n/a	16.64%	12.54%	4.55%	4.09%
Linear	36.24%	6.74%	2.15×10^{-13}%	11.28%	10.59%	27.11%	5.41%
Best of Matérn/RQ	2.15%	2.75%	2.75×10^{-3}%	11.44%	1.00%	5.18%	0.12%

Table: Summary of numerical results: L^2 relative test error.

We compare the performance against several neural operator frameworks on a range of problems taken from two benchmarking studies [2] and find competitive performance.

[2] [Hoop, Huang, Qian, and Stuart 2022, Lu, Meng, Cai, Mao, Goswami, Zhang, and G. E. Karniadakis 2022]

Our key contributions are:

- A simple, low-cost, and competitive kernel method for operator learning.
- Theoretical guarantees for these methods.
- A general framework for doing operator learning with kernel methods.

Batlle, D, Hosseini, and Owhadi "Kernel methods are competitive for operator learning", *Journal of Computational Physics*, 2024