Solving Roughly Forced Nonlinear PDEs with Kernel Methods (and Neural Networks)

Ricardo Baptista[1] Edoardo Calvello[1] **Matthieu Darcy**[1]
Houman Owhadi[1] Andrew M. Stuart[1] Xianjin Yang[1]

[1]Department of Computing and Mathematical Sciences
California Institute of Technology

IMSI - April 2025

Talk summary

The problem

We want to **solve with machine learning rough PDEs** of the form

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

- $\mathcal{P} : H_0^t(\Omega) \to H^{-s}$ is (non-linear) differential operator.
  Canonical example: $\mathcal{P}(u) = \Delta u + f(u)$.
- $u^* \in H_0^t(\Omega)$ is the **solution** and $\xi \in H^{-s}$ is the **forcing term**.

Talk summary

The problem

We want to **solve with machine learning rough PDEs** of the form

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

- $\mathcal{P} : H_0^t(\Omega) \to H^{-s}$ is (non-linear) differential operator.
  Canonical example: $\mathcal{P}(u) = \Delta u + f(u)$.
- $u^* \in H_0^t(\Omega)$ is the **solution** and $\xi \in H^{-s}$ is the **forcing term**.

Hierarchy of spaces

$$\underbrace{H_0^t(\Omega)}_{\text{Functions with } t \text{ derivatives}} \subset L^2(\Omega) \subset \underbrace{H^{-s}}_{\text{Dual space of } H_0^s}$$

Talk summary

The problem

We want to **solve with machine learning rough PDEs** of the form

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

- $\mathcal{P} : H_0^t(\Omega) \to H^{-s}$ is (non-linear) differential operator.
  Canonical example: $\mathcal{P}(u) = \Delta u + f(u)$.
- $u^* \in H_0^t(\Omega)$ is the **solution** and $\xi \in H^{-s}$ is the **forcing term**.

The difficulty

Solving (RPDE) is difficult because the forcing term is rough $\xi \notin L^2$ and, as a result, the solution u^* is also irregular/rough.

Talk summary

The problem

We want to **solve with machine learning rough PDEs** of the form

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

- $\mathcal{P} : H_0^t(\Omega) \to H^{-s}$ is (non-linear) differential operator.

  Canonical example: $\mathcal{P}(u) = \Delta u + f(u)$.
- $u^* \in H_0^t(\Omega)$ is the **solution** and $\xi \in H^{-s}$ is the **forcing term**.

The solution

We propose a kernel based method for solving rough PDEs using **negative Sobolev norms** and **weak measurements**, with **provable convergence**.

Outline

1. Introduction and motivation
2. Machine learning for smooth PDEs
3. Kernel methods for rough PDEs
4. Theory: convergence results
5. Numerical results

Table of Contents

Motivation: Stochastic Partial Differential Equations

Stochastic Partial Differential Equations (SPDEs) are PDEs that include **random fluctuations**. A general class of interest are semi-linear SPDEs with additive noise:

$$\frac{d}{dt}u(t,\mathbf{x}) = \Delta u(\mathbf{x},t) + f(u(t,\mathbf{x})) + \xi(t,\mathbf{x}) \qquad \text{(SL-SPDE)}$$

where:

$$\xi \text{ is } \textbf{space time white noise}.$$

(In general ξ could be a high frequency forcing term).

Motivation: Stochastic Partial Differential Equations

Stochastic Partial Differential Equations (SPDEs) are PDEs that include **random fluctuations**. A general class of interest are semi-linear SPDEs with additive noise:

$$\frac{d}{dt}u(t,\mathbf{x}) = \Delta u(\mathbf{x},t) + f(u(t,\mathbf{x})) + \xi(t,\mathbf{x}) \qquad \text{(SL-SPDE)}$$

where:

$$\xi \text{ is } \textbf{space time white noise}.$$

(In general ξ could be a high frequency forcing term).

Examples

- Phase field models (Allen-Cahn equation).
- Mathematical biology (Nagumo equation).
- Filtering and sampling.

Motivation: Solving SPDEs

Numerically solving an SPDE of the form

$$\frac{d}{dt}u(t,\mathbf{x}) = \Delta u(\mathbf{x},t) + f(u(t,\mathbf{x})) + \xi(t,\mathbf{x}) \qquad \text{(SL-SPDE)}$$

typically involves drawing samples of ξ and solving (SL-SPDE) for that realization.

Motivation: Solving SPDEs

Numerically solving an SPDE of the form

$$\frac{d}{dt}u(t,\mathbf{x}) = \Delta u(\mathbf{x},t) + f(u(t,\mathbf{x})) + \xi(t,\mathbf{x}) \qquad \text{(SL-SPDE)}$$

typically involves drawing samples of ξ and solving (SL-SPDE) for that realization.

Difficulty

Solving (SL-SPDE) is difficult because

1. The forcing term $\xi \notin L^2$ a.s. and is not pointwise defined.
2. The solution u^* is irregular/rough.

This motivates us to develop methods to solve PDEs with very rough forcing terms and/or irregular solutions.

Table of Contents

Solving PDEs with ML: the traditional way

Machine learning methods are a novel way of solving PDEs in a *data-driven way*

$$u^\dagger := \arg\min_{u \in \mathcal{S}} \underbrace{||\mathcal{P}(u) - \xi||^2_{L^2(\Omega)}}_{\text{PDE data}} + \underbrace{||u||^2_{L^2(\partial\Omega)}}_{\text{Boundary data}} + \underbrace{\gamma \mathcal{R}(u)}_{\text{Regularization}}$$

Continuum

Solving PDEs with ML: the traditional way

Machine learning methods are a novel way of solving PDEs in a *data-driven way*

$$u^\dagger := \arg\min_{u \in \mathcal{S}} \underbrace{||\mathcal{P}(u) - \xi||^2_{L^2(\Omega)}}_{\text{PDE data}} + \underbrace{||u||^2_{L^2(\partial\Omega)}}_{\text{Boundary data}} + \underbrace{\gamma \mathcal{R}(u)}_{\text{Regularization}} \qquad \text{Continuum}$$

$$\Downarrow$$

$$\approx \arg\min_{u \in \mathcal{S}} \underbrace{\frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} |\mathcal{P}(u)(x_i) - \xi(x_i)|^2}_{\text{PDE data approximation (pointwise)}} + \underbrace{\frac{1}{N_{\partial\Omega}} \sum_{j=1}^{N_{\partial\Omega}} |u(x_j)|^2}_{\text{Boundary approximation}} + \gamma \mathcal{R}(u) \qquad \text{Discretized}$$

Solving PDEs with ML: the traditional way

Machine learning methods are a novel way of solving PDEs in a *data-driven way*

$$u^\dagger := \arg\min_{u \in \mathcal{S}} \underbrace{||\mathcal{P}(u) - \xi||^2_{L^2(\Omega)}}_{\text{PDE data}} + \underbrace{||u||^2_{L^2(\partial\Omega)}}_{\text{Boundary data}} + \underbrace{\gamma \mathcal{R}(u)}_{\text{Regularization}} \qquad \text{Continuum}$$

$$\Downarrow$$

$$\approx \arg\min_{u \in \mathcal{S}} \underbrace{\frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} |\mathcal{P}(u)(x_i) - \xi(x_i)|^2}_{\text{PDE data approximation (pointwise)}} + \underbrace{\frac{1}{N_{\partial\Omega}} \sum_{j=1}^{N_{\partial\Omega}} |u(x_j)|^2}_{\text{Boundary approximation}} + \gamma \mathcal{R}(u) \qquad \text{Discretized}$$

Three difficulties with Rough PDEs

1. The L^2 norm is not appropriate.
2. Solves the PDE pointwise.
3. Requires that u^* may be well approximated by the class $\mathcal{S}$.

Solving PDEs with ML: the traditional way

Machine learning methods are a novel way of solving PDEs in a *data-driven way*

$$u^\dagger := \arg\min_{u \in \mathcal{S}} \underbrace{||\mathcal{P}(u) - \xi||^2_{L^2(\Omega)}}_{\text{PDE data}} + \underbrace{||u||^2_{L^2(\partial\Omega)}}_{\text{Boundary data}} + \underbrace{\gamma \mathcal{R}(u)}_{\text{Regularization}} \qquad \text{Continuum}$$

$$\Downarrow$$

$$\approx \arg\min_{u \in \mathcal{S}} \underbrace{\frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} |\mathcal{P}(u)(x_i) - \xi(x_i)|^2}_{\text{PDE data approximation (pointwise)}} + \underbrace{\frac{1}{N_{\partial\Omega}} \sum_{j=1}^{N_{\partial\Omega}} |u(x_j)|^2}_{\text{Boundary approximation}} + \gamma \mathcal{R}(u) \qquad \text{Discretized}$$

Two approaches to solving PDEs[1]

- Physics Informed Neural Networks: $\mathcal{S} = \text{NN}_\theta$.
- Kernel methods/Gaussian Processes: $\mathcal{S} = \mathcal{H}_K$ is a RKHS and $\mathcal{R}(u) = ||u||^2_K$

[1] [Raissi, Perdikaris, and Karniadakis], [Chen, Hosseini, Owhadi, and Stuart]

Solving PDEs: the RKHS approach

RKHS approach [Chen, Hosseini, Owhadi, and Stuart]: select $u^\dagger$ in a RKHS

Solve the non-linear least-squares problem:

$$u^\dagger = \underset{u \in \mathcal{H}_K}{\arg\min} \frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} |\mathcal{P}(u)(x_i) - \xi(x_i)|^2 + \frac{1}{N_{\partial\Omega}} \sum_{j=1}^{N_{\partial\Omega}} |u(x_j)|^2 + \underbrace{\gamma ||u||^2_{\mathcal{H}_K}}_{\text{RKHS regularization}}$$

Interpreted as a **MAP estimator** of a Gaussian Process conditioned on non-linear measurements.

Solving PDEs: the RKHS approach

RKHS approach [Chen, Hosseini, Owhadi, and Stuart]: select $u^\dagger$ in a RKHS

Solve the non-linear least-squares problem:

$$u^\dagger = \underset{u \in \mathcal{H}_K}{\arg\min} \frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} |\mathcal{P}(u)(x_i) - \xi(x_i)|^2 + \frac{1}{N_{\partial\Omega}} \sum_{j=1}^{N_{\partial\Omega}} |u(x_j)|^2 + \underbrace{\gamma ||u||^2_{\mathcal{H}_K}}_{\text{RKHS regularization}}$$

Interpreted as a **MAP estimator** of a Gaussian Process conditioned on non-linear measurements.

The kernel approach has strong theoretical guarantees:

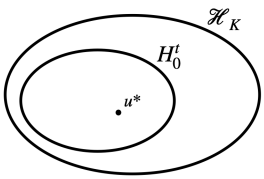
- Provable convergence (**under the condition that** $u^* \in \mathcal{H}_K$).
- Error estimates [Batlle, Chen, Hosseini, Owhadi, and Stuart].
- Bayesian interpretation provides uncertainty quantification.

Table of Contents

Challenges and solutions

Three difficulties with Rough PDEs

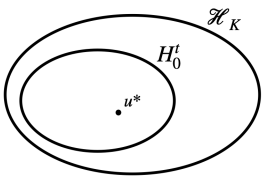
1. The L^2 norm is not appropriate.
2. Solves the PDE pointwise.
3. Requires $u^* \in \mathcal{H}_K$ for the convergence theory (misspecification).



Challenges and solutions

Three difficulties with Rough PDEs

1. The L^2 norm is not appropriate.
2. Solves the PDE pointwise.
3. Requires $u^* \in \mathcal{H}_K$ for the convergence theory (misspecification).



Solutions

Our main contributions are to solve the three main difficulties:

1. Using a **negative Sobolev norm** H^{-s} instead of the usual L^2 norm.
2. **Efficient approximation** of the H^{-s} norm with weak measurements.
3. **Convergence results** of the method without $u^* \in \mathcal{H}_k$.

A Novel Approach to Solving Rough PDEs

We propose a modified version of the machine learning framework adapted to the roughness of the forcing term:

$$u^\gamma = \underset{u \in \mathcal{H}_K}{\arg\min} \|\mathcal{P}(u) - \xi\|^2_{H^{-s}} + ||u||^2_{L^2(\partial\Omega)} + \gamma\|u\|^2_K \qquad \text{Continuum}$$

A Novel Approach to Solving Rough PDEs

We propose a modified version of the machine learning framework adapted to the roughness of the forcing term:

$$u^{\gamma} = \underset{u \in \mathcal{H}_K}{\arg\min} \, \|\mathcal{P}(u) - \xi\|^2_{H^{-s}} + ||u||^2_{L^2(\partial\Omega)} + \gamma \|u\|^2_K \qquad \text{Continuum}$$

$$\Downarrow$$

$$u^{\gamma,N,M} = \underset{u \in \mathcal{H}_K}{\arg\min} \, |\mathcal{P}(u) - \xi|^2_{\Phi^N} + \sum_{i=1}^{m} |u(x_i)|^2 + \gamma \|u\|^2_K \qquad \text{Discretized}$$

(Question: does a minimizer exist?)

A Novel Approach to Solving Rough PDEs

We propose a modified version of the machine learning framework adapted to the roughness of the forcing term:

$$u^{\gamma} = \underset{u \in \mathcal{H}_K}{\arg\min} \|\mathcal{P}(u) - \xi\|^2_{H^{-s}} + ||u||^2_{L^2(\partial\Omega)} + \gamma\|u\|^2_K \qquad \text{Continuum}$$

$$\Downarrow$$

$$u^{\gamma,N,M} = \underset{u \in \mathcal{H}_K}{\arg\min} |\mathcal{P}(u) - \xi|^2_{\Phi^N} + \sum_{i=1}^{m} |u(x_i)|^2 + \gamma\|u\|^2_K \qquad \text{Discretized}$$

(Question: does a minimizer exist?)

We provide a computationally efficient way to discretize the negative Sobolev norms by integrating against a test space $\Phi^N := \text{span}\{\varphi_i\}_{i=1}^N$

$$|\mathcal{P}(u) - \xi|^2_{\Phi^N} = \sum_i^N \sum_j^N A^{-1}_{ij} \int (\mathcal{P}(u) - \xi)\varphi_i \int (\mathcal{P}(u) - \xi)\varphi_j$$

Weak form of the PDE

$$|\mathcal{P}(u) - \xi|^2_{\Phi^N} = \sum_i^N \sum_j^N A^{-1}_{ij} \int (\mathcal{P}(u) - \xi)\varphi_i \int (\mathcal{P}(u) - \xi)\varphi_j$$

This method can be interpreted as solving the PDE in weak form[2]:

$$\underbrace{\mathcal{P}(u)(x) = \xi(x)}_{\text{Pointwise 'data'}} \longrightarrow \underbrace{\int \mathcal{P}(u)\varphi = \int \xi\varphi}_{\text{Weak measurement 'data'}}$$

[2] See also [Ryck, Mishra, and Molinaro].

[3] Can be efficiently computed [Owhadi and Scovel].

Weak form of the PDE

$$|\mathcal{P}(u) - \xi|^2_{\Phi^N} = \sum_i^N \sum_j^N A^{-1}_{ij} \int (\mathcal{P}(u) - \xi)\varphi_i \int (\mathcal{P}(u) - \xi)\varphi_j$$

This method can be interpreted as solving the PDE in weak form[2]:

$$\underbrace{\mathcal{P}(u)(x) = \xi(x)}_{\text{Pointwise 'data'}} \longrightarrow \underbrace{\int \mathcal{P}(u)\varphi = \int \xi\varphi}_{\text{Weak measurement 'data'}}$$

However, the method not only uses weak measurements but also a negative Sobolev norm [3]:

$$A_{ij} := \int \varphi_i (-\Delta)^s \varphi_j$$

[2] See also [Ryck, Mishra, and Molinaro].

[3] Can be efficiently computed [Owhadi and Scovel].

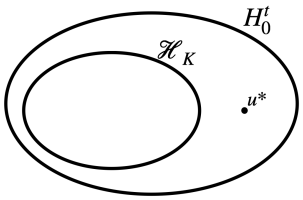
Table of Contents

Theory: goals and main difficulties

We want our numerical solution $u^{\gamma,M,N}$ to converge (in some appropriate sense) to the true solution u^* as the number of measurements $N, M \to \infty$.

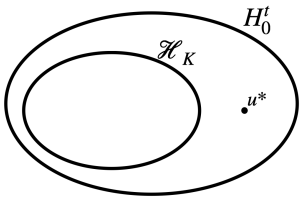
Theory: goals and main difficulties

We want our numerical solution $u^{\gamma,M,N}$ to converge (in some appropriate sense) to the true solution u^* as the number of measurements $N, M \to \infty$. The main difficulty is the problem of *misspecification*: the solution $u^* \notin \mathcal{H}_K$.



Theory: goals and main difficulties

We want our numerical solution $u^{\gamma,M,N}$ to converge (in some appropriate sense) to the true solution u^* as the number of measurements $N, M \to \infty$. The main difficulty is the problem of *misspecification*: the solution $u^* \notin \mathcal{H}_K$.



Two questions

1. Does our optimization problem have a minimizer?
2. Does the minimizer converge (and in which sense) to u^*?

Convergence: Assumptions

Assumptions on the PDE

- The operator $\mathcal{P} : H_0^t \to H^{-s}$ is continuous.
- The solution operator is (locally) stable

$$\|u - u^*\|_{H_0^t} \leq C\|\mathcal{P}(u) - \xi\|_{H^{-s}}$$

Assumptions on the method

- The space Φ^N is dense in H^{-s} as $N \to \infty$.
- The fill distance on the boundary goes to 0 as $M \to \infty$.
- $\mathcal{H}_K \hookrightarrow H_0^t(\Omega)$ and $\mathcal{H}_K$ is dense in $H_0^t(\Omega)$ (satisfied for Matérn kernels).

Convergence: Main Theorems

Theorem (Existence of a minimizer)

The continuum and discretized problems admit a minimizer.

Theorem (Convergence to the True Solution)

The numerical solutions $u^{\gamma,M,N} \in \mathcal{H}_K$ converges to the truth u^* in H_0^t:

$$\lim_{\gamma \to 0} \lim_{M \to \infty} \lim_{N \to \infty} \left\| u^{M,N,\gamma} - u^* \right\|_{H_0^t} = 0.$$

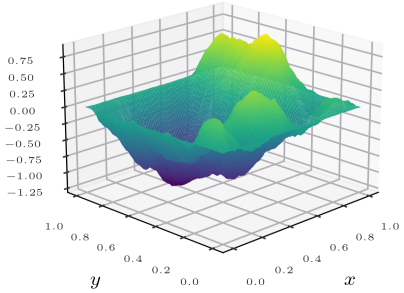
Table of Contents

2D semi-linear PDE

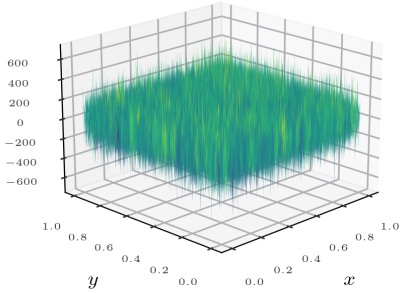
$$-\nu\Delta u + u + \sin(\pi u) = \xi \quad x \in \Omega = (0,1)\times(0,1)$$
$$u = 0 \quad x \in \partial\Omega$$

$$u^* \sim \sum_{i,j=1}^{\infty} \frac{u_{ij}}{(i^2+j^2)} 2\sin(i\pi x)\sin(j\pi y), \quad u_{ij} \sim \mathcal{N}(0,1) \text{ i.i.d.} \quad \xi \in H^{-1^-}(\Omega)$$

Solution u



Forcing term ξ



2D semi-linear PDE: pointwise vs Sobolev loss

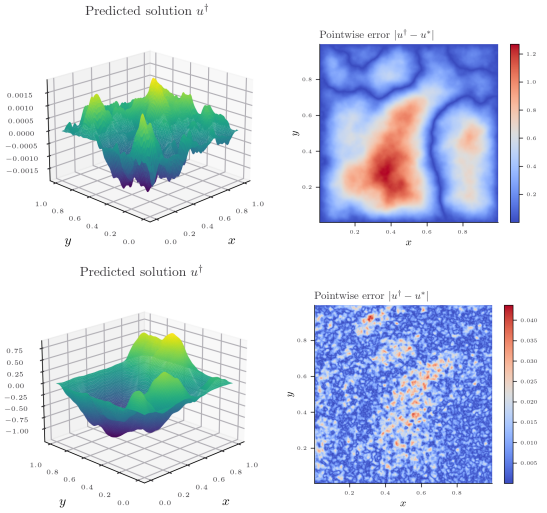
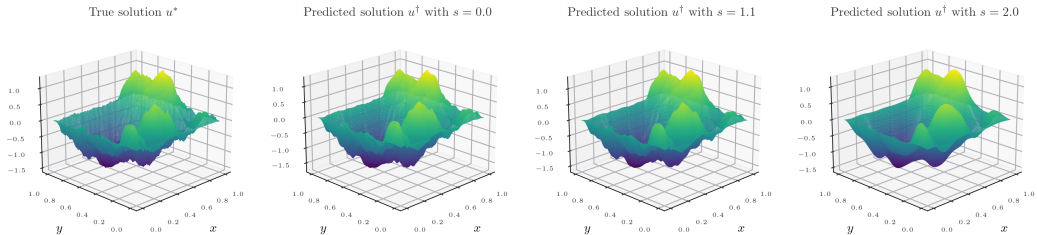


Figure: Pointsiwe loss (top) ≥ 1.00 relative error, $H^{-1.1}$ loss (bottom) ≈ 0.02 relative error.

2D semi-linear PDE: comparison of Sobolev norms

Norm H^{-s}	$s = 0.0$	$s = 0.5$	$s = 1.0$	$s = 1.1$	$s = 2.0$
Kernel method	0.121	0.116	0.046	0.041	0.079
Neural network	0.514	0.434	0.116	0.046	0.082

Table: Relative L^2 error for different choices of Sobolev norms as the loss function



Impact of norm selection on the recovered solution: true solution, unstable, well-specified and oversmoothed.

Time-dependent: stochastic Allen-Cahn equation

$$\partial_t u = \nu \Delta u + u - u^3 + \sigma \xi \quad \text{in } (0,1)$$

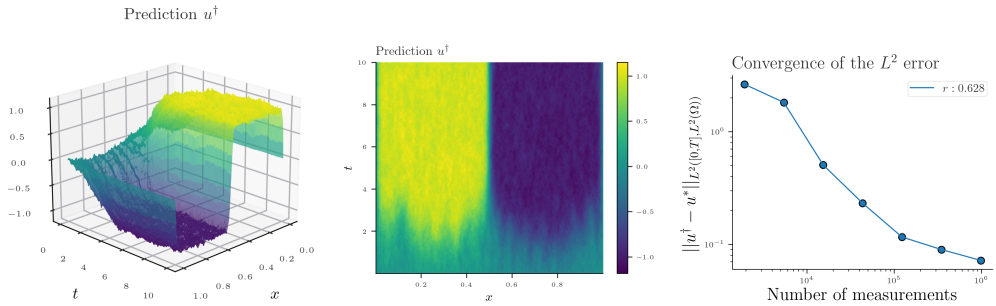


Figure: Stochastic Allen-Cahn equation.

Conclusion

We propose a kernel-based framework for solving PDEs with irregular forcing terms.
Our **theoretical contributions**:

- We extend machine learning-based solvers to PDEs with weaker norms than L^2 (solving the PDE in weak form).
- We leverage the RKHS structure to provide theoretical guarantees of convergence.
- Applies to linear and non-linear PDEs.

Conclusion

We propose a kernel-based framework for solving PDEs with irregular forcing terms.
Our **theoretical contributions**:

- We extend machine learning-based solvers to PDEs with weaker norms than L^2 (solving the PDE in weak form).
- We leverage the RKHS structure to provide theoretical guarantees of convergence.
- Applies to linear and non-linear PDEs.

Our **computational contributions**:

- We provide an efficient approximation of the negative Sobolev norm.
- We show numerically that this approach is effective for kernel methods and PINNs.
- We provide empirical (and some theoretical) error rates.

Ricardo Baptista, Edoardo Calvello, Matthieu Darcy, Houman Owhadi, Andrew M. Stuart, and Xianjin Yang. *Solving Roughly Forced Nonlinear PDEs via Misspecified Kernel Methods and Neural Networks*. 2025. arXiv: 2501.17110

mdarcy@caltech.edu

References I

[1] Ricardo Baptista, Edoardo Calvello, Matthieu Darcy, Houman Owhadi, Andrew M. Stuart, and Xianjin Yang. *Solving Roughly Forced Nonlinear PDEs via Misspecified Kernel Methods and Neural Networks*. 2025. arXiv: 2501.17110.

[2] Pau Batlle, Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. *Error Analysis of Kernel/GP Methods for Nonlinear and Parametric PDEs*. 2023. arXiv: 2305.04962 [math.NA].

[3] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M. Stuart. "Solving and learning nonlinear PDEs with Gaussian processes". In: *Journal of Computational Physics* (2021).

[4] Houman Owhadi and Clint Scovel. *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization*. Cambridge University Press, Oct. 2019.

References II

[5] M. Raissi, P. Perdikaris, and G.E. Karniadakis. "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 (2019), pp. 686–707.

[6] Tim De Ryck, Siddhartha Mishra, and Roberto Molinaro. *wPINNs: Weak Physics informed neural networks for approximating entropy solutions of hyperbolic conservation laws*. 2022. arXiv: 2207.08483 [math.NA]. URL: https://arxiv.org/abs/2207.08483.

Table of Contents

Bonus: Negative Sobolev Norm approximation

We need to compute an approximation of the negative Sobolev norm:

$$\|f\|_{H^{-s}} \approx \|f\|_{\Phi^N}$$

Bonus: Negative Sobolev Norm approximation

We need to compute an approximation of the negative Sobolev norm:

$$\|f\|_{H^{-s}} \approx \|f\|_{\Phi^N}$$

The recipe:

1. Choose a test space $\Phi^N := \text{span}\{\varphi_i\}_{i=1}^N$ to approximate H^s (ex: Haar basis).

Bonus: Negative Sobolev Norm approximation

We need to compute an approximation of the negative Sobolev norm:

$$\|f\|_{H^{-s}} \approx \|f\|_{\Phi^N}$$

The recipe:

1. Choose a test space $\Phi^N := \text{span}\{\varphi_i\}_{i=1}^N$ to approximate H^s (ex: Haar basis).
2. Measure f against the test space:

$$[f, \boldsymbol{\varphi}] := \Big(\int f\varphi_1, \int f\varphi_2, \ldots, \int f\varphi_n \Big)$$

Bonus: Negative Sobolev Norm approximation

We need to compute an approximation of the negative Sobolev norm:

$$\|f\|_{H^{-s}} \approx \|f\|_{\Phi^N}$$

The recipe:

1. Choose a test space $\Phi^N := \text{span}\{\varphi_i\}_{i=1}^N$ to approximate H^s (ex: Haar basis).
2. Measure f against the test space:

$$[f, \boldsymbol{\varphi}] := \Big(\int f\varphi_1, \int f\varphi_2, \ldots, \int f\varphi_n \Big)$$

3. Compute the stiffness matrix $A \in \mathbb{R}^{N\times N}, A_{i,j} := \int_\Omega \varphi_i(-\Delta)^s\varphi_j$ and it's inverse A^{-1} (can be done efficiently [Owhadi and Scovel]).

Bonus: Negative Sobolev Norm approximation

We need to compute an approximation of the negative Sobolev norm:

$$\|f\|_{H^{-s}} \approx \|f\|_{\Phi^N}$$

The recipe:

1. Choose a test space $\Phi^N := \text{span}\{\varphi_i\}_{i=1}^N$ to approximate H^s (ex: Haar basis).
2. Measure f against the test space:

$$[f, \boldsymbol{\varphi}] := \Big(\int f\varphi_1, \int f\varphi_2, \ldots, \int f\varphi_n \Big)$$

3. Compute the stiffness matrix $A \in \mathbb{R}^{N\times N}, A_{i,j} := \int_\Omega \varphi_i(-\Delta)^s\varphi_j$ and it's inverse A^{-1} (can be done efficiently [Owhadi and Scovel]).
4. Define

$$\|f\|_{\Phi^N} := \sqrt{[f, \boldsymbol{\varphi}]^\intercal A^{-1}[f, \boldsymbol{\varphi}]}.$$

Finite dimensional problem

Finite dimensional problem

$$u^{\gamma,N,M} = \underset{u\in\mathcal{H}_K}{\arg\min}[\mathcal{P}(u) - \xi, \varphi]A^{-1}[\mathcal{P}(u) - \xi, \varphi] + \sum_{j=1}^{m} |u(x_j)|^2 + \gamma\|u\|^2_{\mathcal{H}_K}$$

Weak solution

This method can be interpreted as solving the PDE in weak form:

$$[\mathcal{P}(u), \varphi_i] = [\xi, \varphi_i] \quad i = 1, \dots M.$$

This problem can be solved efficiently with the representer theorem and a non-linear least squares optimization techniques such as a variant of the Gauss-Newton algorithm.