Kernel Methods (and Some Neural Networks) for Rough PDEs

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

Workshop on digital twins for inverse problems in Earth science
CIRM 2024

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 $\xi \in H^{-s}(\Omega)$ 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 $\xi \in H^{-s}(\Omega)$ 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 $\xi \in H^{-s}(\Omega)$ 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 $\xi \in H^{-s}(\Omega)$ 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. Machine learning for rough PDEs
4. Convergence results
5. Numerical results

Table of Contents

1. Introduction and motivation
2. Machine learning for smooth PDEs
3. Machine learning for rough PDEs
4. Convergence results
5. Numerical results

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:

$$\partial_t u(t, \mathbf{x}) = \Delta u(\mathbf{x}, t) + f(u(t, \mathbf{x})) + \xi(t, \mathbf{x}) \tag{SL-SPDE}$$

where:

ξ is **stochastic forcing term** ex: space-time white noise.

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:

$$\partial_t u(t, \mathbf{x}) = \Delta u(\mathbf{x}, t) + f(u(t, \mathbf{x})) + \xi(t, \mathbf{x}) \tag{SL-SPDE}$$

where:

ξ is **stochastic forcing term** ex: space-time white noise.

Examples

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

Motivation: Solving SPDEs

Numerically solving an SPDE of the form

$$\partial_t u(t, \mathbf{x}) = \Delta u(\mathbf{x}, t) + f(u(t, \mathbf{x})) + \xi(t, \mathbf{x}) \tag{SL-SPDE}$$

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

Motivation: Solving SPDEs

Numerically solving an SPDE of the form

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

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

1. Introduction and motivation
2. Machine learning for smooth PDEs
3. Machine learning for rough PDEs
4. Convergence results
5. Numerical results

Solving PDEs with ML: the usual way

Machine learning methods solve PDEs by **minimizing a physics informed loss**:

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

Infinite data

Solving PDEs with ML: the usual way

Machine learning methods solve PDEs by **minimizing a physics informed loss**:

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

Infinite data

⇓ Discretization

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

Finite data

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 usual way

Machine learning methods solve PDEs by **minimizing a physics informed loss**:

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

$\Downarrow$ Discretization

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

Two main approaches to solving PDEs with ML

1. Physics Informed Neural networks [Raissi et al.; Tancik et al.]: the class $\mathcal{S}$ is a parametric class of deep neural networks u_θ.
2. Kernel methods/Gaussian Processes [Chen et al.]: the class $\mathcal{S}$ is a *Reproducing Kernel Hilbert Space* (RKHS).

Solving PDEs: the RKHS approach

RKHS approach [Chen et al.]: 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 et al.]: 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 et al.].
- Bayesian interpretation provides uncertainty quantification.

Table of Contents

1 Introduction and motivation

2 Machine learning for smooth PDEs

3 Machine learning for rough PDEs

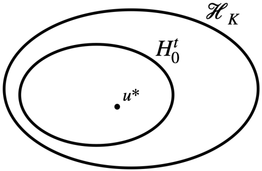
4 Convergence results

5 Numerical results

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.



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

Modified loss adapted to the roughness of the forcing term:

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

$\Downarrow$ Discretization

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

We provide a computationally efficient way to discretize the negative Sobolev norms through a test space $\Phi^N := \text{span}\{\varphi_i\}_{i=1}^N$ (ex: Fourier, finite element, Haar ...).

[1] See also [**wPINN**]

A Novel Approach to Solving Rough PDEs

Modified loss adapted to the roughness of the forcing term:

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

$\Downarrow$ Discretization

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

We provide a computationally efficient way to discretize the negative Sobolev norms through a test space $\Phi^N := \text{span}\{\varphi_i\}_{i=1}^N$ (ex: Fourier, finite element, Haar ...). This method can be interpreted as solving the PDE in weak form[1]:

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

[1]See also [**wPINN**]

Solving rough PDEs with kernel methods

New objective function:

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

- Minimized through a Gauss-Newton formulated on function space (very fast, converges in < 10 steps).
- Closed form solution for linear problems.

Solving rough PDEs with kernel methods

New objective function:

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

- Minimized through a Gauss-Newton formulated on function space (very fast, converges in < 10 steps).
- Closed form solution for linear problems.

We can also minimize this loss with a PINN u_θ:

- Minimized through gradient descent (much slower than Gauss-Newton).
- Perform poorly on linear problems but are good on non-linear problems.
- Typically uses no regularziation ($\gamma = 0$).
- Requires a random Fourier layer [Tancik et al.] to learn the high frequencies.

Table of Contents

1 Introduction and motivation

2 Machine learning for smooth PDEs

3 Machine learning for rough PDEs

4 Convergence results

5 Numerical results

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 Theorem

Theorem (Convergence to the True Solution)

Let $u^{\gamma,M,N} \in \mathcal{H}_K$ solve the approximate problem, then

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

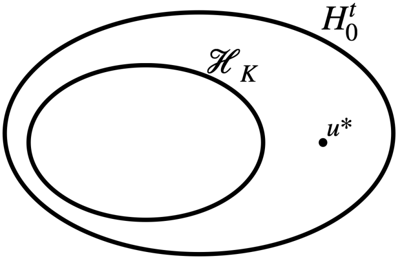


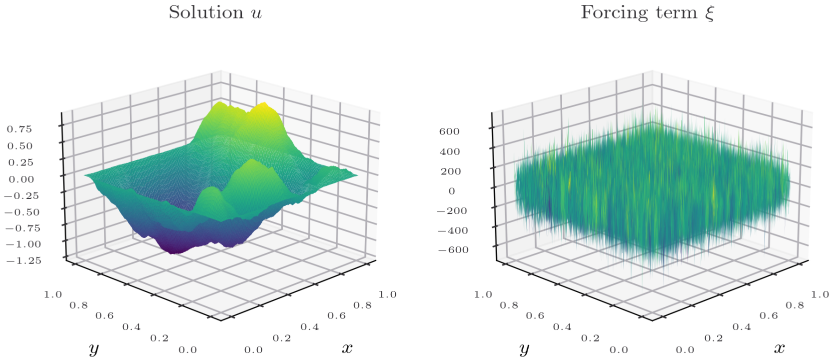
Table of Contents

1. Introduction and motivation
2. Machine learning for smooth PDEs
3. Machine learning for rough PDEs
4. Convergence results
5. Numerical results

2D semi-linear PDE

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

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



2D semi-linear PDE: pointwise loss

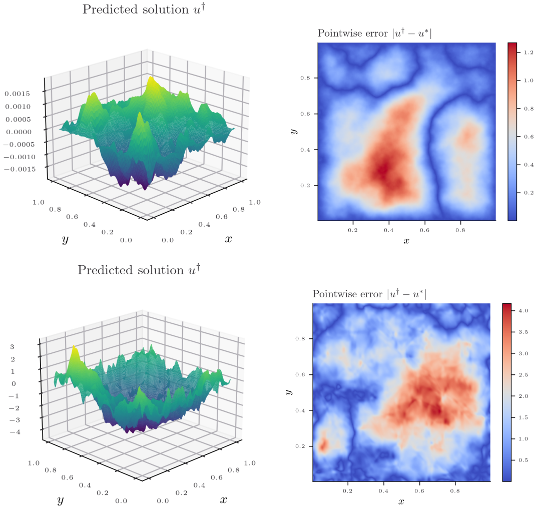


Figure: Kernel method (top), PINN (bottom) ≥ 1.00 relative L^2 error.

2D semi-linear PDE: H^{-1} loss

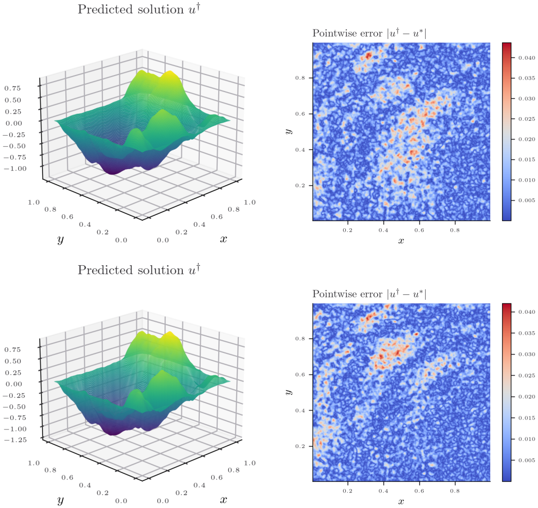


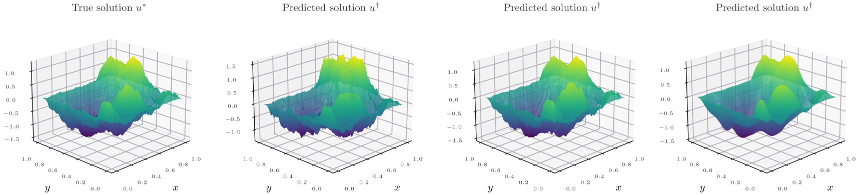
Figure: Kernel method (top), PINN (bottom) ≈ 0.02 relative L^2 error.

Choosing the right norm is very important

When $\xi \in H^{-s}, s > 0.95$.

Norm H^{-s}	$s = 0.0$ (L^2)	$s = 0.95$	$s = 0.96$	$s = 1.0$	$s = 2.0$
PINN error	0.312	0.0896	0.0469	0.0469	0.0740

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



(a) True solution (b) Unstable ($s = 0$) (c) Optimal ($s = 0.96$) (d) Overesmoothed ($s = 2$)

Figure: Effect of different norms on the recovered solution.

Time-dependent: stochastic Allen-Cahn equation

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

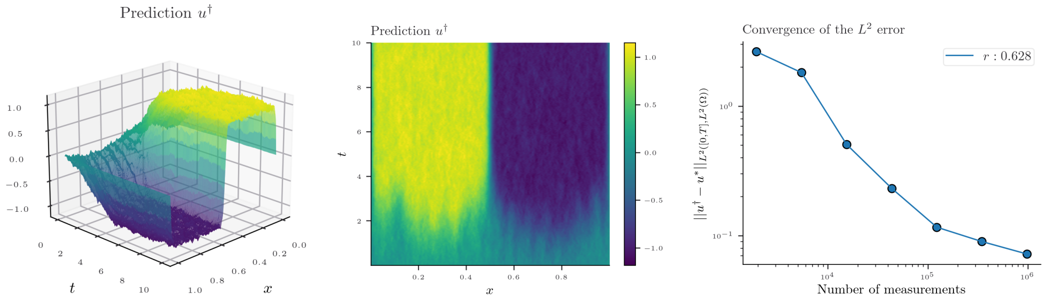


Figure: Stochastic Allen-Cahn equation.

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.

Preprint out soon

R. Baptista, E. Calvello, M. Darcy, H. Owhadi, A. M. Stuart, and X. Yang. *Kernel Methods for Solving Rough PDEs.* 2024.

mdarcy@caltech.edu

References I

Batlle, Pau et al. (2023). *Error Analysis of Kernel/GP Methods for Nonlinear and Parametric PDEs*. arXiv: 2305.04962 [math.NA].

Chen, Yifan et al. (2021). "Solving and learning nonlinear PDEs with Gaussian processes". In: *Journal of Computational Physics*.

Owhadi, Houman and Clint Scovel (Oct. 2019). *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization*. 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.

Tancik, Matthew et al. (2020). *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains*. arXiv: 2006.10739.

Table of Contents

6 Appendix: RPDE

7 Bonus: numerical method

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

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

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

$$[f, \varphi] := \left(\int f\varphi_1, \int f\varphi_2, \dots, \int f\varphi_n \right)$$

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 et al.]).

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:

$$[f, \varphi] := \left(\int f\varphi_1, \int f\varphi_2, \ldots, \int f\varphi_n \right)$$

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 et al.]).
4. Define

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

Finite dimensional problem

Finite dimensional problem

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

Weak solution

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

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

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.

Table of Contents

6 Appendix: RPDE

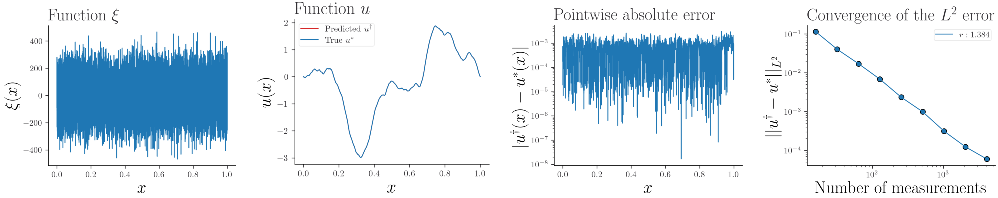
7 Bonus: numerical method

1D Poisson PDE

$$-\nu \Delta u + u = \xi \quad x \in [0,1]$$
$$u = 0 \quad x \in \{0,1\}$$

$$\xi \sim \sum_{j=1}^{\infty} \xi_j \sqrt{2} \sin(\pi j x), \quad \xi_j \sim \mathcal{N}(0,1) \text{ i.i.d.}$$

Here $\xi \in H^{-s}(\Omega)$for any $s > \frac{1}{2}$ and $u^* \in H_0^t(\Omega)$ for $t < \frac{3}{2}$.



(a) Forcing term (b) Solution (c) Pointwise error (d) L^2 error