Learning a Neural Solver for Parametric PDEs to Enhance Physics-Informed Methods

Lise Le Boudec, Emmanuel de Bezenac, Louis Serrano, Ramon Daniel Regueiro-Espino, Yuan Yin, Patrick Gallinari

TLDR; We propose to solve parametric PDEs using Physics-Informed methods by learning a dedicated optimizer that considerably accelerates convergence.

Paper

Code

1. Context & problem formulation

- PINNs have demonstrated interesting performances but remain limited by training time and poor performance in parametric settings.
- We focus on solving **parametric PDEs** from **Physical knowledge**.

$$\mathcal{N}(u;\gamma) = f \quad \text{in } \Omega,$$
$$\mathcal{B}(u) = g \quad \text{on } \partial\Omega.$$

Where $\gamma \in \Gamma$ are the PDE physical parameters, f are forcing terms and g can be initial and/or boundary conditions.

- We assume access to a dataset of pairs composed of the PDE data (γ, f, g) and the associated solution u on a grid.

2. Motivation

- PINNs losses are ill-conditioned and hard to optimize for traditional optimizers.
- They require extensive computational time and numerous iterations to compensate this aspect.

3. How to learn a Physics-informed solver?

- Global framework

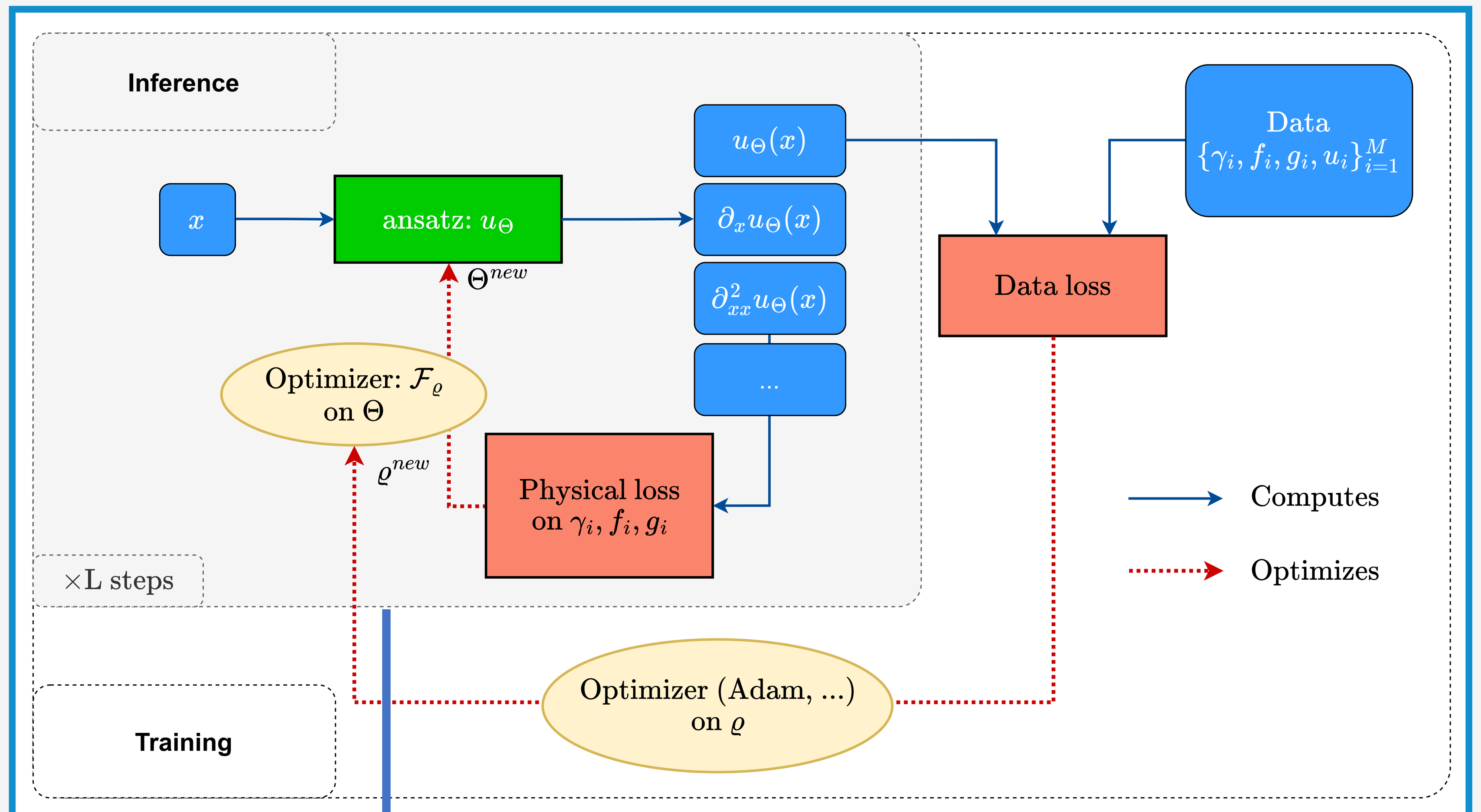

Figure 1: Optimization scheme of a physics-informed method with our framework.

Algorithm 1: Inference using the neural PDE solver.

Data: $\Theta_0 \in \mathbb{R}^n$, PDE (γ, f, g)
Result: $\Theta_L \in \mathbb{R}^n$
for l = 0...L-1 do
  $\Theta_{l+1} = \Theta_l - \eta \mathcal{F}_\varrho(\nabla \mathcal{L}_{\text{PDE}}(\Theta_l), \gamma, f, g)$
end
return Θ_L

The neural solver learns to transform the physical gradient into a more effective gradient direction that achieves fast convergence.

- Theoretical analysis in the linear case

Theorem 1. ***(Convergence rate in the linear case).*** *Given a linear ansatz $u_\Theta(x) = \sum_{i=1}^N \theta_i \phi_i(x)$, assume the conditioner $\mathcal{F}$ behaves like its linearization $P = \text{Jacobian}(\mathcal{F})$, meaning that $\mathcal{F}$ can be replaced by P at any point. Let A be the matrix derived from the PDE loss as eq. (3) for the Poisson equation or eq. (15) in the more general case. Denote by $\kappa(A)$ the condition number of the matrix A. The number of steps $N'(\varepsilon)$ required to achieve an error $\|\Theta_l - \Theta^*\|_2 \le \varepsilon$ satisfies:*

$$N'(\varepsilon) = O\left(\kappa(PA)\ln\left(\tfrac{1}{\varepsilon}\right)\right), \tag{11}$$

Moreover, if $\mathcal{F}$ minimizes $\mathcal{L}_{\text{DATA}}$ this necessarily implies $\kappa(PA) = 1 \le \kappa(A)$. Consequently, the number of steps is effectively reduced, i.e., $N'(\varepsilon) \ll N(\varepsilon)$ with $N(\varepsilon)$ the number of steps of the vanilla PINNs.

4. Results

- Quantitative evaluation: comparison with baselines.

Table 1: Results of trained models - metrics in Relative MSE on the test set. Best performances are highlighted in bold, and second best are underlined

		1d		1d+time	2d	2d+time
	Baseline	Helmholtz	Poisson	NLRD	Darcy-Flow	Heat
Supervised	*MLP + basis*	<u>4.66e-2</u>	1.50e-1	**2.85e-4**	<u>3.56e-2</u>	6.00e-1
Unsupervised	*PINNs+L-BFGS*	9.86e-1	8.83e-1	6.13e-1	9.99e-1	9.56e-1
	PINNS-multi-opt	8.47e-1	1.18e-1	7.57e-1	8.38e-1	6.10e-1
	PPINNs	9.89e-1	4.30e-2	3.94e-1	8.47e-1	1.27e-1
	P2INNs	9.90e-1	1.50e-1	5.69e-1	8,38e-1	1.78e-1
	PO-DeepONet	9.83e-1	1.43e-1	4.10e-1	8.33e-1	4.43e-1
Hybrid	*PI-DeepONet*	9.79e-1	1.20e-1	7.90e-2	2.76e-1	9.18e-1
	PINO	9.99e-1	<u>2.80e-3</u>	4.21e-4	1.01e-1	<u>9.09e-3</u>
Neural Solver	*Ours*	**2.41e-2**	**5.56e-5**	<u>2.91e-4</u>	**1.87e-2**	**2.31e-3**

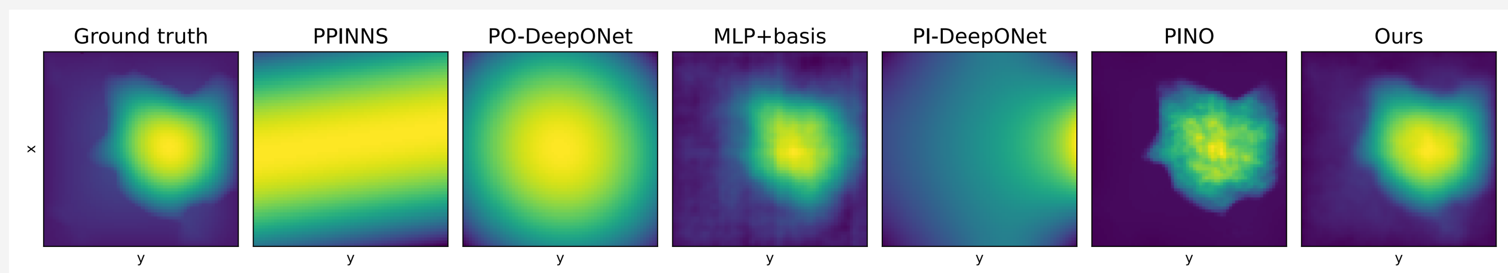

Figure 2: visual comparison of our solver's solution with baselines on the Darcy dataset.

- Solving new PDEs: comparison with optimizers.

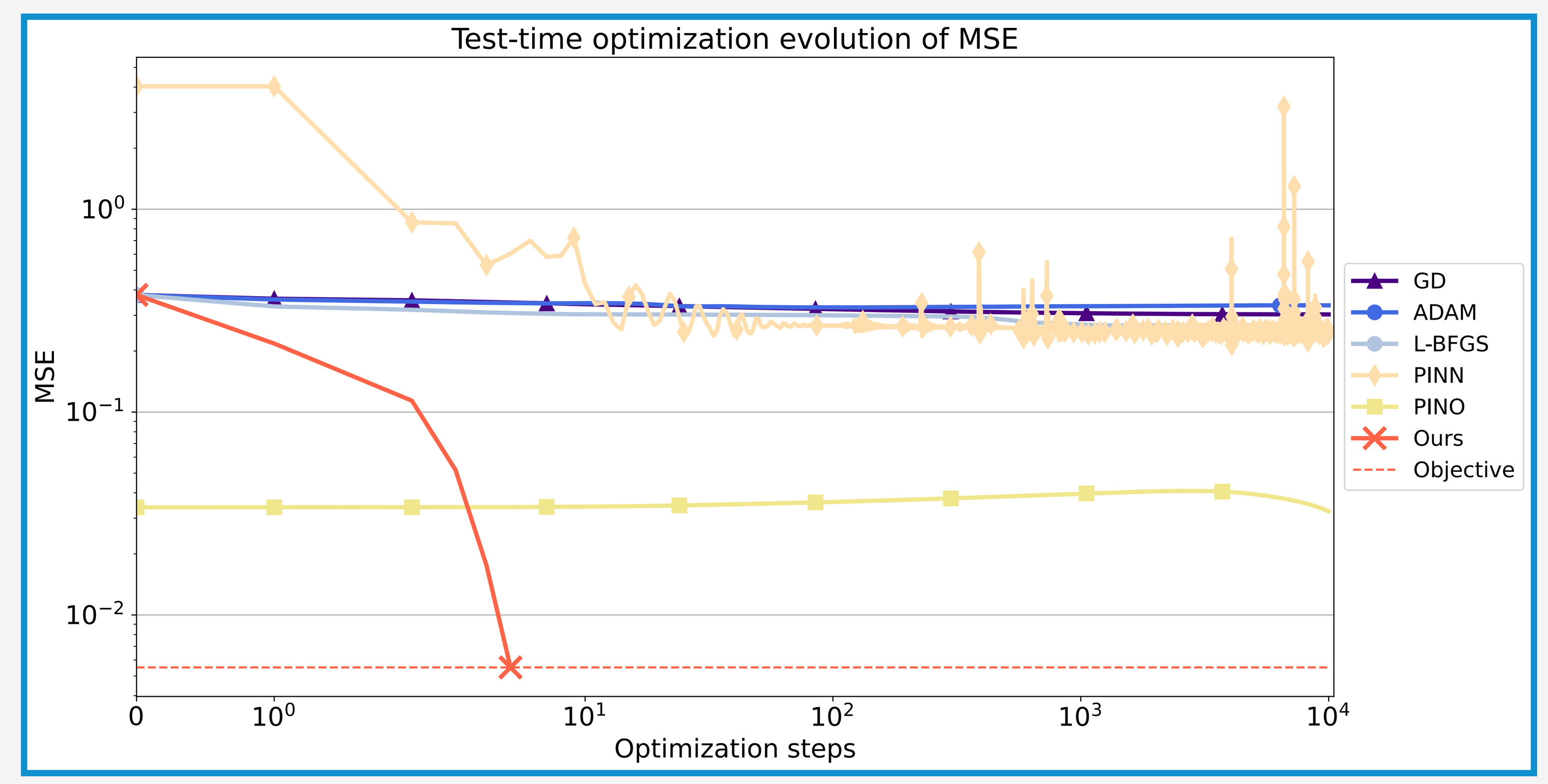

Figure 3: Test-time optimization based on the physical residual loss LPDE for new PDE on Darcy.

Evolution of the reconstruction of the solution with optimization steps.

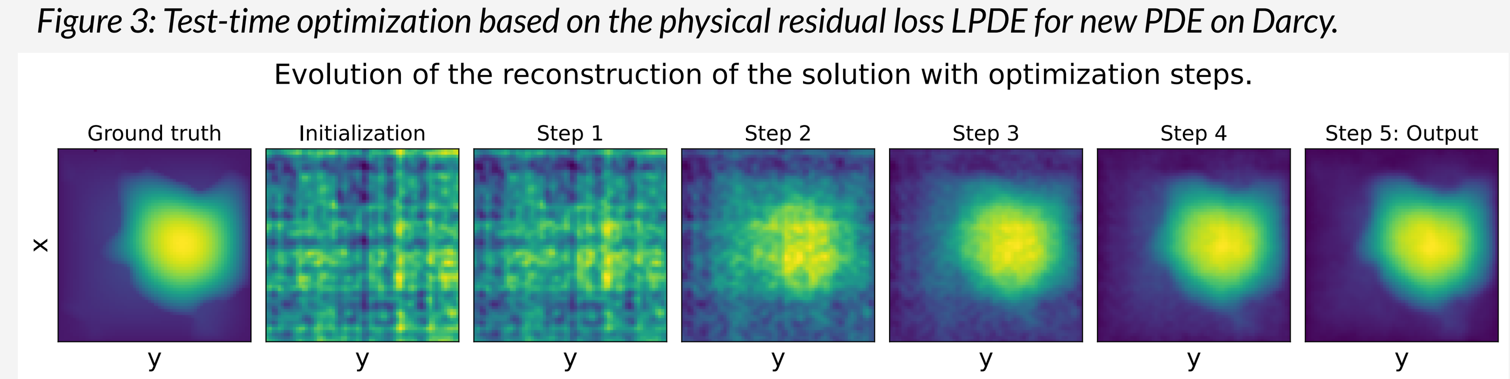

Figure 4: Visualization of the reconstruction of the solution with our method to solve a Darcy PDE.