Neural Network Parametrization for PDE-Constrained Optimization

Mark Miller†, Pritam Kayal†, Yifan Yang†, PJ Clementson†
Faculty Mentor: Prof. Yukun Yue, Martin Guerra.
†University of Wisconsin - Madison

An Important Problem

Consider any PDE with certain boundary conditions like the following:

$$\partial_t - \partial_x^2 u = f \text{ in } \Omega \times I$$

$$u = u_0 \text{ in } \partial\Omega \times I$$

Forward Problem: Given f and u_0, solve the PDE to get u.
Inverse Problem: Given some (partial or noisy) observations about u, which we denote u^{obs}, find f.

We wish to solve the inverse problem by framing it as a PDE-constrained optimization problem [1]. PDE-constrained optimization is a special class of optimization problems where the optimization objective is constrained by partial differential equations (PDEs).

General Form

$$\min_f \; J(u, f) \quad s.t. \quad \mathcal{L}(u, f) = 0$$

f: control variable; u: state variable; J: objective function; $\mathcal{L}$: PDE operator

Fix some $\lambda > 0$. We reformulate our inverse problem as

$$\min_f \frac{1}{2}||u^{\text{obs}} - u||_2^2 + \frac{\lambda}{2}||f||_2^2$$

$$s.t.\ \partial_t u - \partial_x^2 u = f \text{ in } \Omega \times I$$

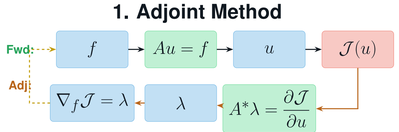
$$u = u_0 \text{ in } \partial\Omega \times I$$

Key Idea: We can parameterize f via a neural network to tackle any hard constraints on f (like non-negativity). By choosing a suitable activation functions (like ReLU), we can convert a traditionally constrained problem into an unconstrained one over the network parameters. Hence, our problem becomes (with the same PDE constraints)

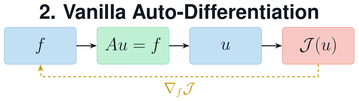
$$\min_\theta \frac{1}{2}||u^{\text{obs}} - u||_2^2 + \frac{\lambda}{2}||f_\theta||_2^2$$

Methods

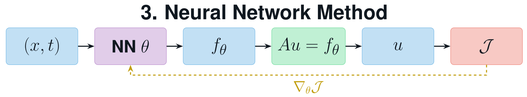
1. Adjoint Method



2. Vanilla Auto-Differentiation



3. Neural Network Method



- **Adjoint method**: Computes gradients by solving a single adjoint (dual) PDE; computational cost is essentially independent of the dimension of the control or parameter space.
- **Vanilla Auto-Differentiation**: Gradients are taken with respect to f (discretized); straightforward to implement but memory and time intensive.
- **Neural Network method**: Gradients are taken with respect to network parameters θ

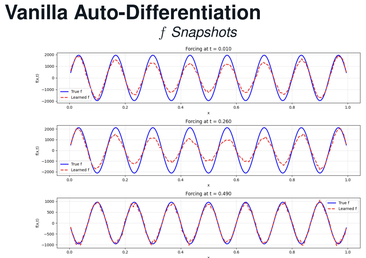
Results

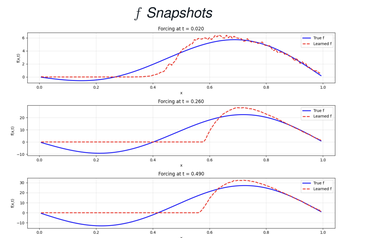
Problem 1: High Oscillation **Problem 2: Non-Negative Constraint**

$$\text{MSE} = \frac{1}{N}\sum_i (u_i^{\text{learned}} - u_i^{\text{true}})^2$$

Vanilla Auto-Differentiation

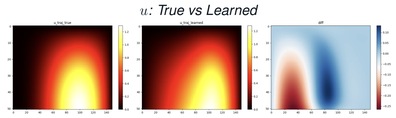
f Snapshots





u: True vs Learned

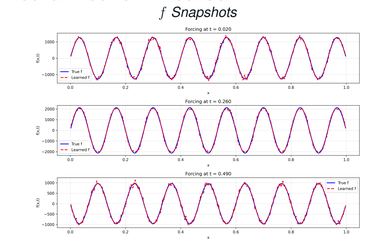
MSE: 0.0727143,
10k epochs, 200×50 grid

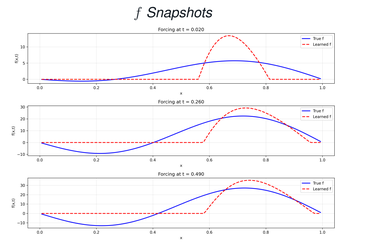


MSE: 0.006551,
5k epochs, 150×50 grid

Neural Network Method

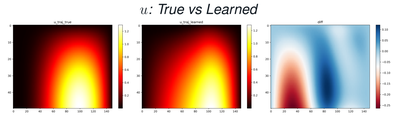
f Snapshots





u: True vs Learned

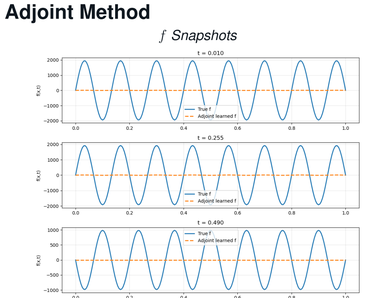
MSE: 0.000012,
10k epochs, 200×50 grid

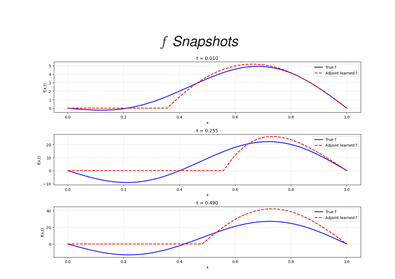


MSE: 0.006767,
5k epochs, 150×50 grid

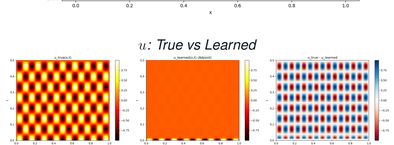
Adjoint Method

f Snapshots

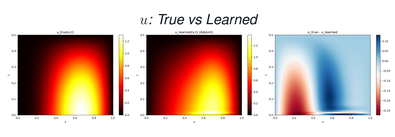




u: True vs Learned



Method Fails



MSE: 0.006481

Observations

- In the highly oscillating case, where $u(x,t) = \sin(15\pi x)\cos(15\pi t)$, the neural network, using Fourier features [2], outperforms the other two methods. The vanilla auto-differentiation method requires an order of magnitude more iterations to achieve similar performance. The given plots are for 5000 iterations for both methods. The adjoint-based approach fails to achieve any noticeable improvement.
- In the case with the non-negative constraint on f, where $u(x,t) = \big(\sin(\pi x) - 0.5\sin(2\pi x)\big)\sin(\pi t)$, all three methods achieve a very similar mean square error (MSE) between the true and learned u. The adjoint method does slightly better.

Future Work

We wish to study reformulations in Reproducing Kernel Hilbert Spaces (RKHS) too. For the non-negativity constraint, projected gradient descent (on α_{ij} after parameterizing $f(x,t) = \sum_i \sum_j \alpha_{ij} K(x, x_i, t, t_j)$) with a non-negative kernel K yields the following result

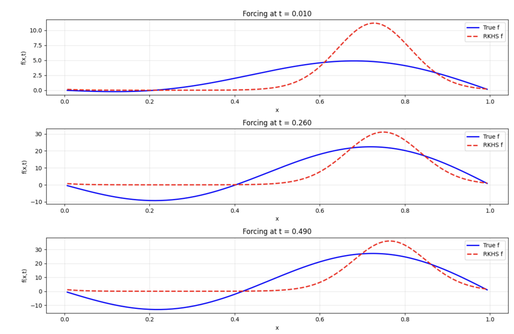


Fig. 1: f learned by Vanilla Grad Method

Specifically, we used the kernel given by

$$K(x, x', t, t') = \exp\left(-\frac{2\sin^2(\pi|x - x'|/p)}{l^2}\right)\exp\left(-\frac{2\sin^2(\pi|t - t'|/p)}{l^2}\right)$$

Besides other reformulations (for comparing with neural networks), we wish to study more complex constraints that can be effectively handled by other activation functions in neural networks. We also wish to study more complex PDEs with nonlinearities.

Acknowledgments

We would like to thank our instructor, Yukun Yue, for his guidance and support throughout this project. We are also grateful to our graduate mentor, Martin Guerra, for his valuable insights, discussions, and continuous encouragement.

References

[1] Denis Khimin et al. “Optimal control of partial differential equations in PyTorch using automatic differentiation and neural network surrogates”. In: *arXiv preprint arXiv:2408.12404* (2024).
[2] Matthew Tancik et al. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 7537–7547.