Multifidelity Finite Basis Physics Informed Neural Networks

Name: Damien Beecroft

Hosting Site: Pacific Northwest National Lab

Mentors: Amanda Howard and Panos Stinis

Mentor's Signature:

Abstract

Physics informed neural networks (PINNs) struggle to successfully learn solutions to differential equations that exhibit high-frequency oscillations or multi-scale behavior. Multilevel finite basis physics informed neural networks (FBPINNs) tackle this problem by recursively discretizing the solution domain and training coupled neural networks on the subdomains. High frequency structures are made less oscillatory with respect to the length scale of these smaller subdomains. This plays to the spectral bias of PINNs and enables one to learn the global solution faster. In this work we integrate multifidelity methods into multilevel FBPINNs to further improve the learning rate. This internship was virtual. I created a GitHub project to communicate what I am working on with my mentors. Anyone interested in solving differential equations could benefit from this project.

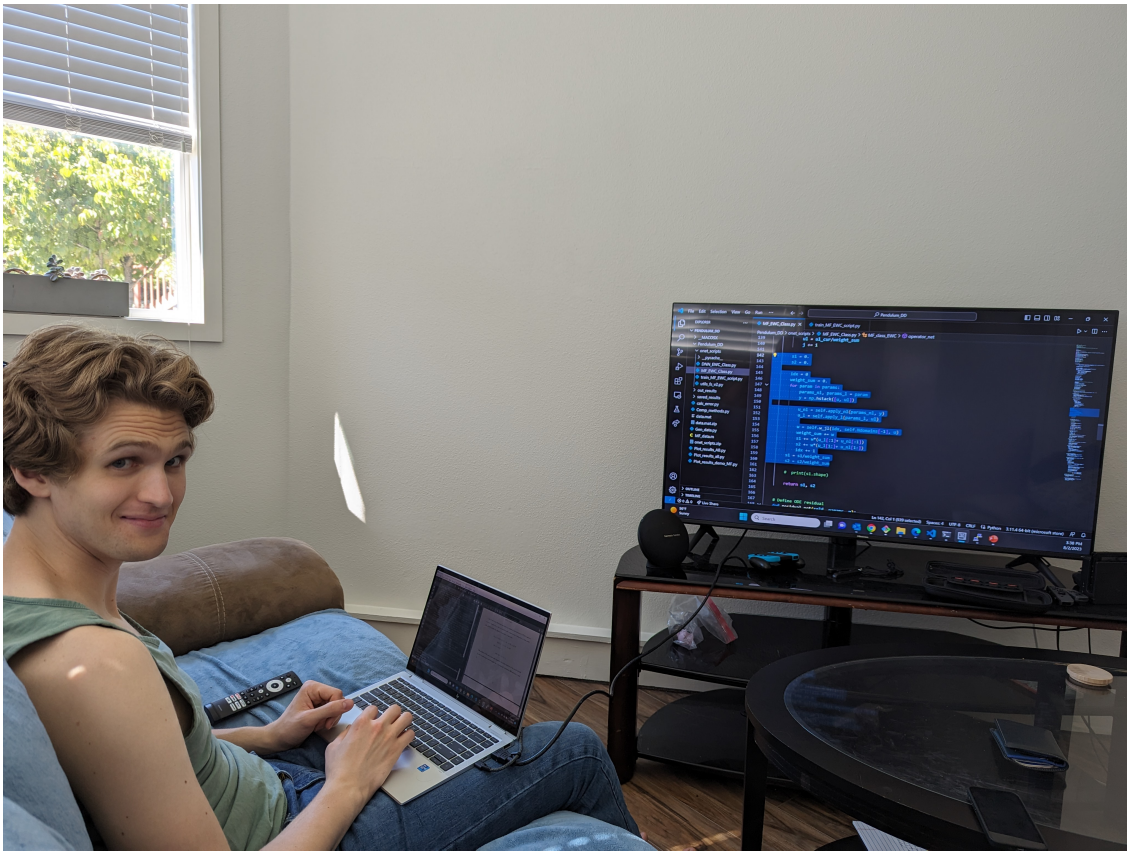

Figure 1: My internship at Pacific Northwest National Lab was virtual. My office was anywhere I lugged my computer. On some of my more adventurous days I ventured all the way down to my living room to use the television as a secondary monitor. On an even more auspicious occasion my girlfriend and soon to be doctor Caitlin Neher was kind enough to take my photo for this final report.

1 Introduction

In 2016 Raissi et al. (cite) invented the physics informed neural network (PINN). This framework enables machine learning algorithms to learn solutions to differential equations with just the physical constraints. The dream is to achieve a flexible differential equation solvers that are able to find high accuracy solutions with little expert knowledge. However, this dream is still a ways off. We briefly review the concept of a PINN here. Suppose that we have the following differential equation.

$$\begin{aligned} u(t,x)_t + N[u(t,x)] &= 0 \quad \text{for} \quad x \in \Omega,\ t \in [0,T] \\ B[u(t,x)] &= 0 \quad \text{for} \quad x \in \partial\Omega \\ u(0,x) &= u_0(x) \end{aligned}$$

N and B are known differential operators that contain no time derivatives. PINNs construct a neural network $\tilde{u}(t,x)$ that, is penalized every training step in proportion to how poorly the network satisfies the physical constraints. In particular,

$$\begin{aligned} \tilde{u}(t,x)_t + N[\tilde{u}(t,x)] &= l_1 \quad \text{for} \quad x \in \Omega,\ t \in [0,T] \\ B[\tilde{u}(t,x)] &= l_2 \quad \text{for} \quad x \in \partial\Omega \\ \tilde{u}(0,x) - u_0(x) &= l_3 \end{aligned}$$

$$\text{total loss} = l_1 + l_2 + l_3$$

The gradient of the loss is taken with respect to the hidden variables of the neural network in order to determine how the weights are to be adjusted. Automatic differentiation is used to differentiate the neural network with respect to the temporal and spatial variables. Now that we have discussed the basic set up of PINNs, we can touch on the current difficulties of training PINNs.

Summary of Papers

- PINNs: A PINN penalizes the solution if it does not satisfy the physical constraints of the differential equation. PINNs train on a single initial condition.

- Multifidelity PINNs: These are PINNs that are capable of learning from a combination of low and high fidelity data. To this end, MPINNs learn correlations between the different data fidelitites and use this correlation to overcome sparse high fidelity data.

- Multifidelity DeepONets: Multifidelity DeepONets use multifidelity data just like MPINNs. However, DeepONets learn the differential equation for all initial conditions.

- Multifidelity Continual Learning: This method attempts to increase the domain of accuracy of a neural network. Suppose we have a neural net, $\mathcal{NN}_{i-1}$, that can solve a PDE on Ω_{i-1}. $\mathcal{NN}_{i-1}$ is used as a low fidelity approximation train a neural net $\mathcal{NN}_i$ to solve the PDE on a larger set: Ω_i.

- Fixed Points in PINNs: PINNs have a tendency to converge to fixed points that satisfy the physical constraints of the PDE, but not the initial or boundary conditions. This problem is less prevalent when one trains the PINN on a shorter time interval.

- MDD for PINNs: This method iteratively partitions the domain of the PDE into a sequence of overlapping sub-domains and trains PINNs on each sub-domains. The PINN trained on the previous level is used as a low-fidelity approximation for the PINNs on the lower levels.

- Point Selection for PINNs: This paper introduces a probability distribution for sampling new points to compute the PINN residual and improve training. This paper introduces a probability distribution using the residual that is used to find the best locations to sample the residual and train the PINN.

- How and Why PINNs Fail to Train: This paper linearizes the neural network about the initial parametrization, θ^0, and performs analysis on the path of the continuous gradient descent of the neural network using this linearization. A variety of very strong assumptions are needed for this analysis to be useful.

- Respecting Causality: This paper introduces an adjustment to the PINN loss function that encourages the PINN to solve the differential equation at earlier time steps before the later ones. This helps improve performance.

Hard Bias in PINNs to Satisfy Structure of Dynamical Systems

Within the PINNs Physics informed neural networks struggle to capture When analyzing a dynamical system, there are a variety of features that one uses to characterize the solution behavior. These features include

- Fixed points:
  -