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. In this work we integrate multifidelity methods into multilevel FBPINNs to improve convergence. We tested this method on a collection of test problems and analyzed the errors and solution structure to determine the quality of the results. This project can benefit anyone that is trying to model a differential equation that classical numerical or analytical methods struggle to solve.

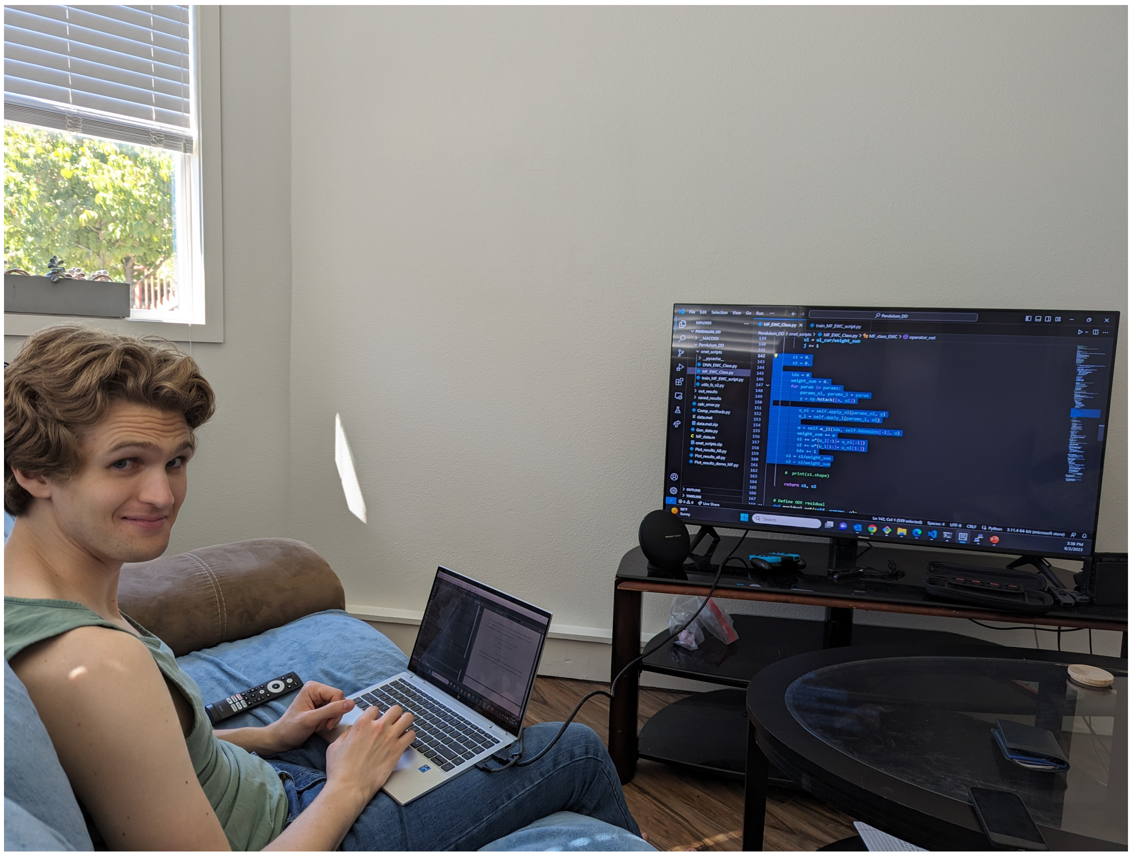

Figure 1: My internship at Pacific Northwest National Lab was virtual. My office was anywhere I lugged my computer. On some of my more adventurous days I ventured all the way down to my living room to use the television as a secondary monitor. On an even more auspicious occasion my girlfriend and soon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

1 Introduction

Physics informed neural networks (PINNs) learn the solution to a differential equation for a given set of initial and boundary conditions [3]. We briefly review PINNs here. Suppose that we have the following differential equation.

$$\begin{aligned} u(t,x)_t + N[u(t,x)] &= 0 \quad \text{for} \quad x \in \Omega,\ t \in [0,T] \\ B[u(t,x)] &= 0 \quad \text{for} \quad x \in \partial\Omega \\ u(0,x) &= u_0(x) \end{aligned} \tag{1}$$

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.

$$\begin{aligned} \tilde{u}(t,x)_t + N[\tilde{u}(t,x)] &= l_1 \quad \text{for} \quad x \in \Omega,\ t \in [0,T] \\ B[\tilde{u}(t,x)] &= l_2 \quad \text{for} \quad x \in \partial\Omega \\ \tilde{u}(0,x) - u_0(x) &= l_3 \\ \text{total loss} &= l_1 + l_2 + l_3 \end{aligned} \tag{2}$$

The gradient of the total loss is taken with respect to the hidden variables of the neural network 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.

This is a simple and elegant framework. However, PINNs struggle to learn oscillatory and multi-scale solutions. They often converge to erroneous fixed point solutions in these scenarios [4]. This makes sense since fixed point solutions have a small residual and small hidden weights. There are a variety of methods and techniques that have been introduced to improve the convergence of PINNs.

2 Description of Project

My research this summer focused on testing a new extension of PINNs: multifidelity finite basis physics informed neural networks (MFFBPINNs). This method is the combination of two other extensions of PINNs: multifidelity PINNs and multilevel finite basis PINNs.

Multifidelity PINNs learn from two separate data sources: a high and low fidelity data source. The multifidelity network learns the correlation between the two data sources to achieve an accurate prediction of the true solution [2]. This process is illustrated in Figure 2.

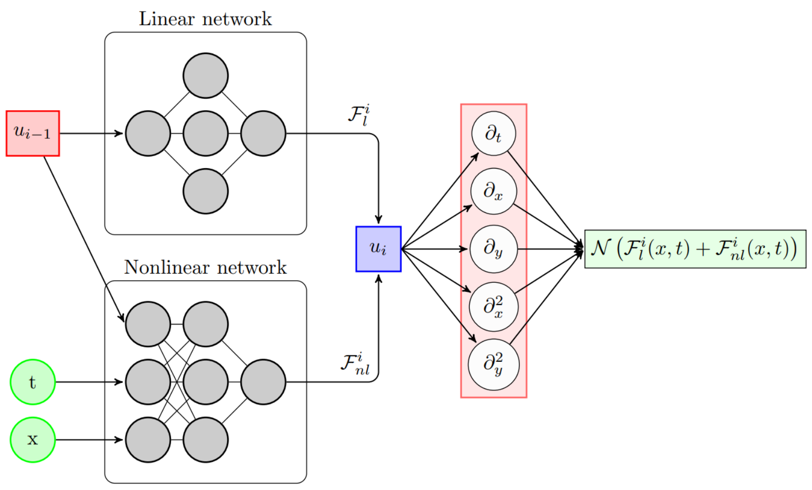

Figure 2: This is a graphic of a multifidelity PINN from Meng at al. [2]. The multifidelity PINN takes in the low fidelity approximations (u_{i-1}) and the collocation points (x, t) and outputs a new set of approximations (u_i). This network can then be differentiated to see how well it satisfies the differential equation.

Multilevel finite basis PINNs take the problem domain and decompose it into a hierarchy of overlapping subdomains [1]. A neural network is trained on each of these subdomains and the levels are averaged to get the overall solution. This process is illustrated in Figure 3.

Now we can introduce MFFBPINNs. Suppose we are attempting to learn the differential equation posed in Equation 1. MFFBPINNs construct a hierarchy of neural networks. On the zeroth level of the hierarchy is a single fidelity neural network that requires no low fidelity data to make a prediction. This single fidelity method could be a classical PINN

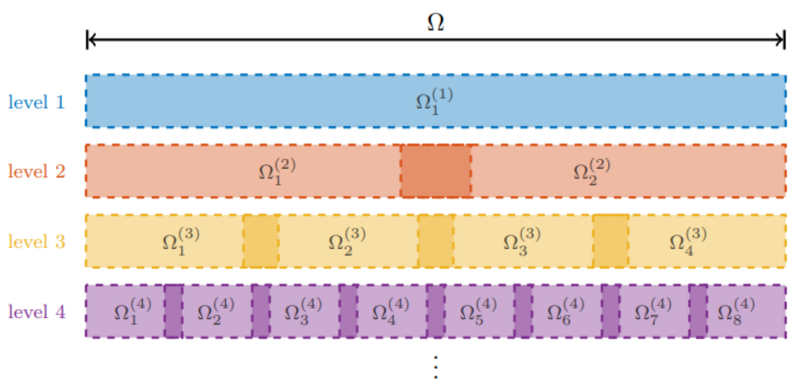

Figure 3: This is a graphic illustrating multilevel finite basis PINNs from Dolean et al. [1]. Level 1 is trained first. Then, level 2 is trained to correct for the residual of level 1. Levels 1 and 2 are averaged to get the new estimate of the global solution. Level 3 is trained to correct for the errors of the new global solution. Levels 1, 2, and 3 are averaged. This process continues until a desired level of accuracy is achieved.

or even the solution of a numerical solver. The only caveat is that we need to be able to take spatial and temporal derivatives of this single fidelity solution. After the single fidelity network is trained on Ω, the domain is broken up into a set of overlapping subdomains for the first multifidelity level: $\{\Omega_n^1\}_{n=0}^{N_1}$. Each multifidelity PINN on this level takes in the evaluation of the single fidelity PINN on the zeroth level as its low fidelity approximation to the solution. Now, the multifidelity PINNs on the first level are trained while the single fidelity network's parameters are frozen. For the second level of multifidelity PINNs the process is repeated. The domain is broken up into a new set of subdomains: $\{\Omega_n^2\}_{n=0}^{N_2}$. However, now the multifidelity networks on the first level provide the low fidelity data for the second level. This process repeats until a desired accuracy is achieved. This process is illustrated in Figure 4.

3 Contributions Made to the Project

I had two tasks to accomplish during this project. The first was to create an efficient implementation of the MFFBPINN algorithm. The second was to test the MFFBPINNs on a collection of test problems. This position was ten weeks. I spent roughly the first three

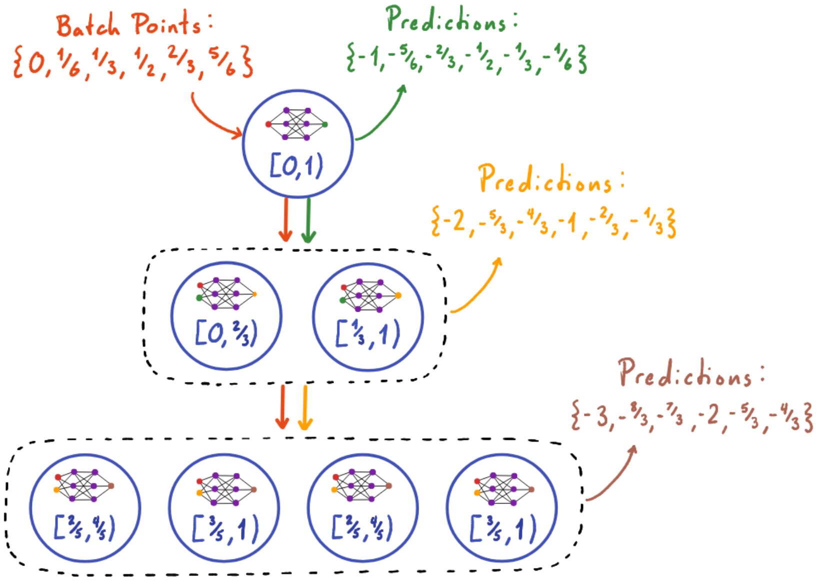

Figure 4: A simple diagram of the MFFBPINN algorithm for a 1D problem where the number of domains is doubled at each step.

weeks reading up on the PINNs, setting up my computer, learning to use the PNNL GPU cluster, and trying to understand source code that Amanda sent me. I began my attempt to improve on Amanda's source code on week four. She had a rather ad hoc implementation of the MFFBPINN algorithm set up for a damped pendulum equations. I call this code "ad hoc" because every neural network on every subdomain was evaluated on the entire domain. However, only the points inside the domain of the neural network have a non-zero weight. Therefore, many points were evaluated by the network only to be zeroed out by the weight function later on. This is very inefficient. I began developing an algorithm that fixed this problem and was also very flexible so that one could generalize the implementation to higher dimensions with more complex subdomains. I spent weeks four and part of week five doing this. Towards the end of the fifth week I began trying to integrate Jax into my method. This did not work. Jax is very efficient at the cost of flexibility. I could not figure out how to integrate Jax into my algorithm. I tried for roughly a week and a half to no avail. At this point Amanda and Panos said that I should prioritize getting results over finishing the algorithm. Thus, I used the ”ad hoc” pendulum code to produce results. The code struggled to converge for this problem, so I had to play around with parameters to get good results. I also implemented the causality weighting from Wang et al. [5]. I also had to implement the domain decomposition for the wave and Allen-Cahn equations. I used this code to produce the following results. ADD SOME RESULTS

4 New Skills and Knowledge

I gained a great deal of knowledge during the course of this internship. I began the internship by reading papers on PINNs and different methods for overcoming their convergence issues. After I had gained a sufficient basis of knowledge on PINNs I began setting up my account on the PNNL GPU cluster: Marianas. Setting up and using this account forced me to brush up my skills in a variety of tools including Anaconda, Bash, Slurm, and GitHub. After I had set up my account on the cluster I began working on implementing MFFBPINNs to solve a variety of test problems: damped pendulum, wave equation, and Allen-Cahn equation. Working on the MFFBPINN implementation extended my knowledge of Python and introduced me to Jax. Furthermore, I did not have much hands on experience training neural networks prior to this internship. This summer has given me the opportunity to put theory into practice and vastly expand my knowledge in the field of machine learning.

5 Experience and Impact on My Career

Machine learning is by far the hottest research topic currently. There are many research positions in this area. Until now I was not certain what these positions would be like on a day to day basis. My internship at PNNL has given me valuable insight on this matter and will enable me to make a more informed decisions concerning my career path.

6 Relevance to the Mission of NSF

The statutory mission of NSF is, "To promote the progress of science; to advance national health, prosperity, and welfare; and to secure the national defense; and for other purposes." The methods being developed in this work may one day help engineers and scientists solve challenging differential equations without expertise in analytical or numerical methods. This could in turn accelerate the development and production of technologies for the benefit of humanity.

7 Acknowledgements

I acknowledge and appreciate the support from the National Science Foundation Mathematical Sciences Graduate Internship program. This project was completed with support from the U.S. Department of Energy, Advanced Scientific Computing Research program, under the Scalable, Efficient and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems (SEA-CROGS) project (Project No. 80278), The computational work was performed using PNNL Institutional Computing at Pacific Northwest National Laboratory. Pacific Northwest National Laboratory (PNNL) is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

References

[1] Victorita Dolean, Alexander Heinlein, Siddhartha Mishra, and Ben Moseley. Multilevel domain decomposition-based architectures for physics-informed neural networks, 2023.

[2] Xuhui Meng and George Em Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. *Journal of Computational Physics*, 401:109020, jan 2020.

[3] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations. *CoRR*, abs/1711.10561, 2017.

[4] Franz M. Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C. Geiger. On the role of fixed points of dynamical systems in training physics-informed neural networks, 2023.

[5] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks, 2022.