Summer Internship for Damien Beecroft

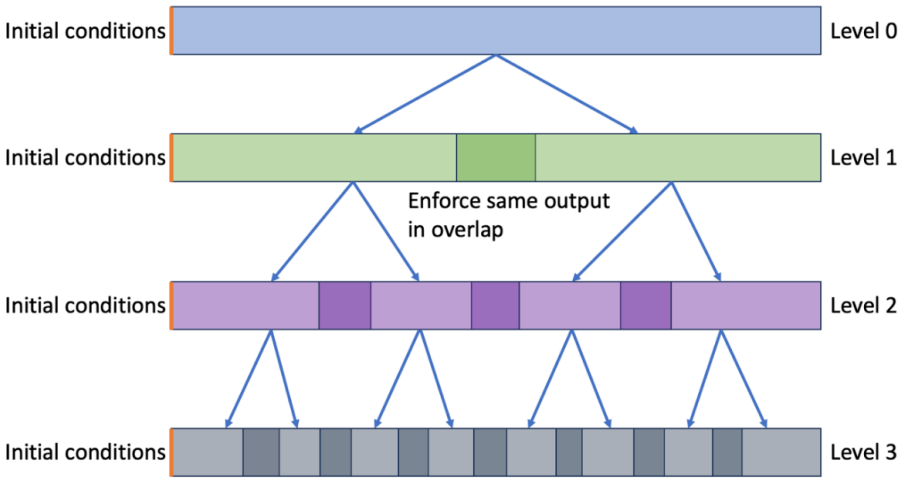
Advisors: Amanda Howard, Panos Stinis

August 4, 2023

Physics-informed neural networks, or PINNs, have shown great promise in learning the solution to partial differential equations. However, there are still cases where PINNs fail to train. This summer, we will develop new techniques for cases where PINNs fail to train.

This document serves only as a suggested outline, and creativity is encouraged. The remaining weeks will be filled in as we see how the results develop. Many extensions are possible, including training deep operator networks (DeepONets) for a range of input parameters, or switching to model other dynamical systems.

The suggested reading for each week consists of background information that you may find helpful as you work. I suggest you write a short, 1-2 sentence, summary of each paper and add it a working document after reading each paper. This will be the introduction to your final report, and later the introduction for a publication based on your work. Articles are available through PNNL when you're logged into the VPN.



We will consider a multifidelity domain decomposition approach, where a PINN is trained in the full domain first. Then, the full domain training is used as a low fidelity prediction for a prediction in a subdivided domain. Two PINNs are trained in level 1, and the domains overlap so initial conditions are not needed to train the subdomain that occurs for later time. This process is repeated until the desired accuracy is reached across the domain.

We will be working with Alexander Heinlein, who is an expert in domain decomposition. We will start by testing this setup on the case of an undamped pendulum, to see if we can overcome the issue of converging to fixed points for long time. If successful, we will move to more complex examples. Creativity is encouraged in designing training algorithms.

To Do

Reading:

- Parallel Physics-Informed Neural Networks via Domain Decomposition: `https://arxiv.org/pdf/2104.10013.pdf`
- Fourier Analysis Sheds Light on Deep Neural Networks: `https://arxiv.org/abs/1901.06523`

Coding:

- Do Now
  - You can precompute the weights that correspond to each batch ahead of time and generate them in the batch.
  - Preselect the points in the batch. Don't randomly choose some points in each batch.
  - Make the batching more efficient. Generate the weights for the newest level inside of the batch.
  - Given a time scale that trains, decrease the network size until failure.
  - Work on implementing the wave equation
  - Make the multifidelity code efficient
- Pendulum
  - Run the domain decomposition code in `Pendulum_DD`
  - Point selection and evaluation
    - Make sure that the code does not evaluate every network at all points. Only evaluate networks on points within the support of their weight function. There are two potential methods for how to do this.
      - Change the data generator function to only generate output points in a given domain
      - Once you are given a point, check whether or not the point is in the support of the weight function.
    - Work on parallelization of neural networks defined on different sub-domains.
    - Testing
      - Running with and without the residual driven point selection
      - Look at the sensitivity of the weights of the loss function
    - Do testing on how high we can get the maximum time of the pendulum. Currently it is at 10. Can we get it to 20? 50?
- Wave Equation
- Allen-Cahn
- Lorenz

Week 1

Videos:

- Stanford CS229M Lecture 13 (Neural Tangent Kernel): `https://www.youtube.com/watch?v=btphvvnad0A&list=RDCMUCBa5G_ESCn8Yd4vw5U-gIcg&index=1`

Reading:

- PINNs: `https://www.sciencedirect.com/science/article/pii/S0021999118307125`
- Multifidelity PINN: `https://arxiv.org/pdf/1903.00104.pdf`
- Multifidelity DeepONets: `https://arxiv.org/pdf/2204.09157.pdf`
- Multifidelity Continual Learning: `https://arxiv.org/pdf/2304.03894.pdf`
- Fixed points and PINNs: `https://arxiv.org/pdf/2203.13648.pdf`
- Multilevel Domain Decomposition for PINNs: `https://arxiv.org/pdf/2306.05486.pdf`
- Point selection for PINNs: `https://www.sciencedirect.com/science/article/pii/S0045782522006260`
- Respecting Causality is All you Need: `https://arxiv.org/pdf/2203.07404.pdf`

Coding:

I was sent the pendulum code by Amanda. I took some time this week to go through it and get a feel for what is going on. I have not run the code since I do not have my PNNL laptop. I plan to have the code up and running early next week.

Week 2

Reading:

- A Method for Representing Periodic Functions and Enforcing Exactly Periodic Boundary Conditions with Deep Neural Networks: `https://arxiv.org/pdf/2007.07442.pdf`
- How and Why PINNs Fail to Train: `https://arxiv.org/pdf/2007.14527.pdf`
  - Neural Tangent Kernel Convergence and Generalization: `https://arxiv.org/abs/1806.07572`
  - Deep Neural Networks as Gaussian Processes: `https://arxiv.org/abs/1711.00165`

Coding:

- Get access to Marianas
- Set up GitHub repository
- Set up GitHub repository on Marianas and pull code from `pnnl_research`
- Run Amanda's code locally
- Plot solutions from Amanda's code
- Tried to get `onet_scripts` running on Marianas. It did not work.

Week 3

Notes:

- Tried to get `onet_scripts` running on Marianas. It did not work. I spent a lot of time on this.
- I went through the original code Amanda sent me slowly to understand exactly how it works.
- I went through the altered code in `Pendulum_DD` to understand exactly how it works. This was rather time consuming.
- I realized that the problem of only applying neural networks to points within their domains can be solved with a rather simple sorting algorithm.

Coding

- Implementing domain sorting code.

Week 4

Coding

- I created a function that sorts the points amongst the tree of domains.
- I created a super class that tracks which nodes are on which levels of the tree.
- I created a function that evaluates the nodes on each level and sorts them accordingly.
- I began taking the code apart and putting it back together in order to get things working.

Week 5

Coding

- This week I went to the PNNL office in Seattle to meet Amanda and Sarah.
- Sarah and I met with Shady. Shady was able to figure out why Sarah and I could not get the code running on Marianas. There were extra packages in our `python` and `.conda` files. After deleting these packages the code ran.
- I spent time trying to figure out how to set up the visit to Pasco. Of course, Concur got the better of me in the end and I did not end up going to Pasco.
- I worked on integrating the theory I had worked out in week 4 in with the `Pendulum_DD` code.
- I began to have difficulties with Jax as I was beginning to integrate the code in with Amanda's.
- I read through the Jax documentation in order understand how to implement the multifidelity network architecture.

Week 6

Coding

- I learned about pytrees in Jax. I began attempting to translate the code that I have to be compatible with Jax.
- I began to realize that I could not get pytrees working with in Jax, at least the way I was implementing it.

- I talked to Amanda and she suggested that I try making batches for each subset where different neural networks are defined.
- I tried implementing the multiple batching, however, the code was not running. Jax was again giving me problems.
- I tried running the original `Pendulum_DD` code that Amanda sent me and it gave me an error. I tried extracting the original zip file that Amanda gave me and that also gave me the same error. I cannot get the `Pendulum_DD` code to run at all now. I am doubting if I ever actually ran it.

Week 7

Coding

- I realized that `Pendulum_DD` code that I was sent never ran in the first place. It had a bug where `self.NDomains` was converted to a traced value implicitly. The code then attempted to make a numpy array with this code, but this can't be done with a traced variable.
- I did the pendulum code up to time 20. It failed to train.
- I increased the amount of resources used on the single fidelity network for the T = [0,20] training session. The single fidelity network was a bit better, but still struggled.
- I modified the code so that neural networks are only evaluated on points that lie within their support. However, the implementation was sloppy and the code is far slower than the code where all neural nets are evaluated everywhere.
- I began thinking about how to make the code more efficient. I believe that the main reason is the for loops in my code and the poor batching method. I am working on changing both of these.

Week 8

Week 9

Week 10