# Meeting Agendas

Damien

July 27, 2023

# June 14, 2023

Participants: Amanda and Damien

## Agenda

Today, we are discussing the following three papers

- 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`

## Questions

### Multifidelity PINNs

Q We do not have to train the final final neural network that takes in $y_H$ and outputs $f_e$, correct? This is simply used for penalization?

A This is correct.

Q Does penalizing the gradient force the solution to converge to an approximation that is locally at least first order?

A They penalize the gradient because it trains better.

Q Can you discuss in more detail how each sub-network is trained?

A Training is done separately generally for each network separately. However, linear and nonlinear are trained simultaneously. The low-fidelity method can be trained separately. Training can be done all simultaneously?

Q How is data used for training in Figure 3? Did they train on data not in Figure 3a?

A Figure 3a contains all the training data. There is probably more testing data.

### Multifidelity DeepONets

Q What are the essential differences between PINNs and DeepONets?

- For DeepONets you DO need to train the last network since you do not know the functional form of the operator, correct?

A PINNs train for one initial condition. Deep operator networks attempt to learn how all initial conditions map to solutions. The operator form is known, so the last part is known. We do not need to train the operator.

- What are the key differences in how PINNs and ONets are trained?

A Need smaller time steps and more data for DeepONets as opposed to PINNs.

Q How are $M_L$ and $P_L$ related in Figure 1?

A $M_L$ contains the points/data needed to get a unique solution of the differential equation. $P_L$ are simply data that we interpolate.

**Continual Learning**

Q You talked about this a bit yesterday, but what are the assumptions we make as far as knowledge about the low-fidelity model we are given?

- Even if we do not know the data the low-fidelity model was trained on, do we know the domain that data was in?

A We assume that the low-fidelity network is correlated with the solution on the new data set. We do not necessarily even know what domain the previous network was trained on.

- If the previous answer is yes, then why are we treating $\mathcal{NN}_{i-1}$ as a low-fidelity predictor on $\Omega_{i-1}$?

Q Why do we need to train a single fidelity and a multifidelity PINN on $\Omega_1$?

A The results were better when you initialize the multifidelity approach in this manner.

**General Questions**

Q Have you performed experiments with noise? Are these methods robust to noisy data?

A There is error added to the low-fidelity data in Burger's equation in the ONet paper. The high-fidelity data helps compensate for this noise.

# June 22, 2023

Participants: Amanda and Damien

## Agenda

I am having a hard time running the code on Marianas. The pendulum code works fine on my local machine, but there is something wrong still.

### Comments

- The code works when I run it on my machine, so this has to be a problem with the environment.
- I have tried running two different slurm scripts: `mybatch.slurm` and `batch.slurm`. Both `.slurm` files give the same output.

## Questions

Q Once I have run the lines listed below, do I need to run them again every time? Do the lines following `conda create --name jax-cuda` create a permanent environment that I can just load with `conda activate jax-cuda` from now on? Do I need to run the following lines again?

- `module purge`
- `module load cuda/11.4`
- `module load python/miniconda3.9`
- `source /share/apps/python/miniconda3.9/etc/profile.d/conda.sh`
- `conda create --name jax-cuda`
- `conda activate jax-cuda`
- `conda install scipy=1.10.0`
- `pip install --upgrade pip`
- `pip install --upgrade "jax[cuda11_pip]" -f https://storage.googleapis.com/jax-releases/jax_cuda`
- `conda install tqdm`
- `pip install torch`
- `conda install cudnn`

A No, you do not need to re-install the packages in the environment every time.

Q I put the above lines into a .sh file and the output when I run the .sh file is different than when I run the code otherwise. Is there a reason for this? I feel like it should be the same?

A Just put lines directly into the terminal since you do not need to run the code every time.

Q What is `SF_script.py`? Should I have access to this file?

Amanda's Comments:

- Possible reasons for error:
  - The version of cuDNN may not be compatible with what I need to do.
  - Try to re-install jax in the CUDA environment
  - This could be a memory error, so run on dl and not dl-shared
  - If it is a memory error, then it could be that the system is pre-allocating memory that does not exist

  - Take the line: `os.environ["XLA_PYTHON_CLIENT_PREALLOCATE"]="false"`. Put it in the script between when you import all the packages and when you start the code
  - If the code is not running on Marianas, developing locally is a good thing to do.

Multi-fidelity PINN Stuff:

- Amanda sent me an Overleaf for the multi-fidelity domain decomposition.
- Currently the code is not smart about how it selects points.
- There will be a running list of problems to work on in the Overleaf.

## June 26, 20223

Participants: Damien and Alexander

### Agenda

In this meeting we are discussing goals and expectations for the upcoming weeks.

### Goals

- Implement multi-fidelity framework for domain decomposition. (This is the primary task to complete before anything else)
  - Do one level of the DD method separately and then plug this into the multi-fidelity framework.
  - Use 1D pendulum as the first test problem.
  - See how the method performs without the insights of the other papers.
  - Create data structures in a manner that they are scalable to higher dimensional problems.
- Implement point selection methods that depend on the magnitude of the residual as opposed to training with fixed collocation points.
- Implement adaptive refinement based on the norm of the residual.
- Implement causality in the algorithm.

### Notes

- What is going to be the official document for the write up?

# June 28, 2023

Participants: Damien and Amanda

## Agenda

In this meeting we are discussing issues with the `Pendulum_DD` code and aspects within it that need to be changed.

### Things To Work On

- Pendulum
  - Running the domain decomposition code
  - Changing how the point selection is done to be smarter
    * Find where the neural networks are being evaluated at residual points.
    * Make sure that you are not evaluating every network at all the points. Only evaluate networks on points within the support of their weight function.
    * THERE ARE TWO OPTIONS FOR CHOOSING WHICH NETWORKS EVALUATE WHICH POINTS: Change the data generator function to generate output points in a given domain OR once you are already given a point, check whether or not the point is in the support of the weight function.
    * The data generator function can be smart about parallelization of the points.
    * 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
  –

# June *, 2023

## Agenda

In this meeting I want to clarify any remaining questions that I have about the code in `onet_scripts`. Note: I use verbatim font when I am referencing something in the code. Also note: I added some comments, so the line numbers may not be precise (sorry about that).

## Questions

### Original Code

Q It seems that there are seven total neural networks trained: `0` through `5` and `A`. Exactly how are these neural networks related to each other? On what data is each network trained?

A `A` is the single fidelity network. Networks 0 through 5 are a series of multifidelity networks. Each network uses the most recently trained network as the low fidelity approximation. Each network is trained on a different, randomly sampled subset on the time points on the interval $[0, 10]$.

Q `train_MF_EWC_script.py, 105-106`: What are `epochs` and `epochsA2` used for? What is the difference?

A `epochs` is the number of epochs used in the multifidelity networks. `epochsA2` is the number of epochs used in the single fidelity networks.

- Questions that I had that are not important enough to spend time discussing currentlym (I believe).

  Q `train_MF_EWC_script.py, 155-156`: What is the purpose of `F` and `lam`?

  Q What does `EWC` stand for again?

  Q `DNN_EWC_Class.py, 215`: How does the `item(res_dataset)` line work? I know it generates new batches of data, but I don't exactly understand the logic.

  Q `train_MF_EWC_script.py, 202`: What is the class `DataGenerator_res2` for? Why do we need a second residual class?

  Q What is the low fidelity model for each neural network? Is it simply the previous trained network? Is it some combination of all the previously trained networks?

  Q `utils_fs_v2.py, 46`: What does the explicit definition of the `__getitem__` method doing? How does it get the batch?

  Q `train_MF_EWC_script.py, 216`: In this line we are passing in `params_A`, which is the set of parameters from the initial single-fidelity network. We also pass in all the parameters from all of the previous networks, `params_prev`. Why are we passing in all the previous networks? It feels like we should only need to pass in the previous network.

  Q `MF_EWC_Class.py, 58-59`: These lines are setting the parameters of the nonlinear and linear networks to the value that was computed in the previous network?

### Pendulum_DD

Q `train_MF_EWC_script.py, 239`: What is each of the five indices into `params_prev` for?

A
- The fifth index is for choosing between the weight matrix and the shift vector.
- The fourth index chooses which layer of the network you are accessing. Has type "tuple".
- The third index chooses whether you look at the non-linear or the linear network. Has type "list".
- The third layer determines which neural network on the corresponding level you are going to look at. Has type "tuple".

- The first index chooses which level of the domain decomposition hierarchy you are looking at. Has type "list".
- `params_prev` holds all of the parameters from all of the neural networks on all of the layers of the domain decomposition. Has type "list".

Q `MF_EWC_Class.py, 62, 63`: Why are the nonlinear and linear networks being set equal to what they are on these lines?

Q `MF_EWC_Class.py, 116`: What is `u` in the context of the `operator_net` function? What is `ul`?

A It seems that `u` is the set of collocation points where the neural net is analyzed. `ul` is low-fidelity solution at these collocation points. This notation is rather confusing. Be careful.

- You can precompute the weights that correspond to each batch ahead of time and generate them in the batch.
- Preselect the points