FEL3751 - Tutorial 1

Matthieu Barreau

April 2025

During this tutorial, you will work on the code available at

`https://github.com/mBarreau/FEL3751/`.

Some solutions are available on the branch `solution`.

1 Initialization

This tutorial will use Python. Please check that you have a working version of Python and an appropriate IDE before starting. If this is not the case, you can follow the tutorial there.

1. Copy the files from GitHub. If you are familiar with `git`, you can just clone the repository, otherwise, you can click on the "<> Code" button and download the zip file containing all the files for this tutorial.
2. You open the folder in VS Code and use a recent version of Python as an interpreter in a new virtual environment. You can install the required packages from the file `requirements.txt` or use the command `pip install -r requirements.txt`.
3. I recommend that you install the Jupyter extension, which might be helpful for running and debugging.
4. You can open the file `example.py` and run the first cell to check that all requirements are indeed installed correctly. You might have issues with line 11 if LaTex is not installed on your computer. You can comment on this line if you need.
5. You are ready to start!

The file `example.py` is the executable and contains the example that will be solved using both PINN and NeuralODEs. The file `utils.py` contains useful functions to create a neural network and plotting utilities; you can have a look at this file, but I suggest that you do not edit it for now. The other The two files that you will need to edit first are `pinn.py` and `neural_ode.py`.

2 Physics-Informed Neural Networks

In this first part, you will work on the PINN simulation, modifying the file `pinn.py`.

2.1 Required

1. The first thing to do is to get a basic machine learning algorithm to work. For that, you need to update the `get_mse_data` and `primal_update` functions.
2. Evaluate the performance depending on the system's sampling and its extrapolation properties. Note the number of epochs required for convergence. You can try with different seeds to evaluate the robustness of the algorithm.
3. You will now move to PINN. For that, you need to fill in the function `get_mse_residual` and set the weight `self.weight` to `1`.
4. Evaluate the performance depending on the system's sampling and its extrapolation properties. Note the number of epochs required for convergence. You can try with different seeds to evaluate the robustness of the algorithm.
5. Finally, you can implement the primal-dual algorithm by filling in the `dual_update` function.
6. Evaluate the performance depending on the system's sampling and its extrapolation properties, even outside of the prediction zone. Note the number of epochs required for convergence. You can try with different seeds to evaluate the robustness of the algorithm.

2.2 Optional

1. You can investigate the effect of the number of sampling points `N_phys` to evaluate its impact on the solution. Same with `N_dual`.
2. You can try other strategies for improved training, as the one described in [1, 2].
3. You can try to extend it to PDEs (consider, for example, the heat equation).
4. You can try to add some parameters in the physics equation itself to learn the dynamics.

3 Neural ODEs

3.1 Required

- First, fill in the function `_call` using the Tensorflow Probability documention on the Dormand-Prince ODE solver.
- Evaluate the performance depending on the system's sampling and its extrapolation properties. Note the number of epochs required for convergence. You can try with different seeds to evaluate the robustness of the algorithm.
- Evaluate the extrapolation properties concerning a change in the initial condition (during inference only).
- Implement a "discretize-then-optimize" method using a Runge-Kutta of order 2 method (see the slide from the course).
- Evaluate the performance similarly.

3.2 Optional

- Investigate how to minimize the amplitude of the ε network.
- Investigate how to do uncertainty quantification based on properties of finite difference solvers.
- Investigate the case where the system is not perfectly known (f has a parameter, for instance).

References

[1] Matthieu Barreau and Haoming Shen. Accuracy and robustness of weight-balancing methods for training pinns. *arXiv preprint arXiv:2501.18582*, 2025.

[2] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, 2024.