7. Conclusions

In this project, we implemented from scratch a library proposing a variety of functionalities for Bayesian Physics-Informed Machine Learning applied to Scientific Computing, an approach that enabled to tackle differential problems with an empowered version of Neural Networks that can take physical information into account and quantify uncertainty of the prediction.

The extension of the library is mainly in the horizontal direction, because it can provide a variety of functionalities for many tasks related to the B-PINNs framework, ranging from model training to dataset management. Its flexibility should moreover make it a suitable starting base for the refinement of the functionalities, with the ideal ultimate goal of producing a fully-comprehensive library for Bayesian Physics-Informed Deep Learning.

In the implementation, we relied on Python's available tools for OOP, which enabled to design a skeleton for the library that should be enough rigid to guarantee a coherent pipeline and enough flexible to reduce the amount of functionalities constraints.

On the other hand, for what concerns the library validation, with the proposed showcase of results we stressed on the main implementation pillars (algorithms, physical information and code portability), to then concentrate the effort for obtaining quality results in a variety of applications for the specific HMC method.

Further developments of the library are highly encouraged; the library is open to plugins by design, because it has been thought for the extension either of new training algorithms and of new differential problems that can be dealt with, by implementing the corresponding classes inheriting from `Algorithm` or from `Equation`.

For more advanced problems, third part datasets require to be in the same format as the ones generated in the project to represent a suitable source of data for the model. Therefore, it could be very useful to add a simple interface for integrating and exploring real word data before pre-processing.

Consequently, another feature that can be added it the possibility to deal with the visualization of higher dimensional cases, by implementing suitable visualization tools for 2D uncertainty quantification and for the overall visualization of the 3D case.

For what concerns extensions of functionalities to the library, a proposed challenge can be the inclusion of tools to deal with parameter estimation problems, for which we already set-up the premises and the space to build upon with `PhysNN`. The crucial part in this development is the addition of the gradients on physical parameters $\boldsymbol{\lambda}$ to be learnt into the class `LossNN` and probably a wrapper for $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$.

On the user-experience side, we could propose the implementation of a structured pipeline, based on grid search or more advanced techniques with the aim of partially automatize the tuning task as well, still being careful to the computational times required when asking for many queries of the parameters.

A. Installation Guide

The code is contained in the GitLab repository `PACS_bpinns`.
To run the scripts contained in the executable repository, the user needs:

- `python` version 3.10.* (download from here)
- `virtualenv` version 20.14.* (download from here)

Installation for Windows:

1. Go into the directory of your project with `cd project_folder_path`
2. Create an empty virtual environment with `py -m venv .\my_env_name`
3. Enter into the virtual environment with `my_env_name\scripts\activate`
4. Check that the environment is empty with `pip freeze`; normally, it should print nothing
5. Install the required packages from the `.txt` file `requirements.txt` with `pip install -r requirements.txt`
6. Run again `pip freeze` and check that the environment is no longer empty
7. Add the environment folder to your `.gitignore` (in order to avoid pushing the packages on git!)
8. To exit from the virtual environment, use `deactivate`

Installation for Linux and Mac:

1. Go into the directory of your project with `cd project_folder_path`
2. Create an empty virtual environment with `virtualenv .\my_env_name`
3. Enter into the virtual environment with `source my_env_name\bin\activate`
4. Check that the environment is empty with `pip freeze`; normally, it should print nothing
5. Install the required packages from the `.txt` file `requirements.txt` with `pip install -r requirements.txt`
6. Run again `pip freeze` and check that the environment is no longer empty
7. Add the environment folder to your `.gitignore` (in order to avoid pushing the packages on git!)
8. To exit from the virtual environment, use `deactivate`

References

[1] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: a review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, apr 2017.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015.

[3] Daniele Ceccarelli. Bayesian Physics-Informed Neural Networks for inverse uncertainty quantification problems in cardiac electrophysiology. Master's thesis, Politecnico di Milano, 2021.

[4] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on Bayesian Neural Networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR 2015*, 2014.

[6] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[7] Kevin Linka, Amelie Schäfer, Xuhui Meng, Zongren Zou, George Em Karniadakis, and Ellen Kuhl. Bayesian Physics Informed Neural Networks for real-world nonlinear dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 402:115346, 2022. A Special Issue in Honor of the Lifetime Achievements of J. Tinsley Oden.

[8] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *NIPS 2016*, 2016.

[9] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (Part I): data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[10] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (Part II): data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[11] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-PINNs: Bayesian Physics-Informed Neural Networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 425:109913, jan 2021.