Project Proposals

Damien Beecroft

August 2022

Project One

Abstract

How does one appropriately combine the vast wealth of theory from numerical analysis and dynamical systems with physics informed neural networks to most efficiently and accurately solve differential equations? This is a complex problem whose answer depends heavily upon the equation, the available data, and a multitude of other factors. I will explore this trade-off to provide guidelines for how to incorporate data and theory effectively for a range of different scenarios. In this paper I explore a variety of test problems, each with wildly different properties: the Helmholtz equation, Burger's equation, and Navier-Stokes equation.

Introduction

In this paper we use multifidelity physics informed neural networks (MFPINNs) as the sole algorithm for our trials. However, unlike in previous works, the single fidelity solver will be a classical numerical method as opposed to a PINN. This framework is ideal because it can naturally incorporate numerical methods along with PINNs and many other methods (such as training a PINN to learn the residual of a numerical solution) can be interpreted as a sub-case of training an MFPINN. There are three broad metrics for success in the trials that we perform in this paper: accuracy, speed, and generalizability. With these three variables we construct the Pareto front of MFPINNs as one shifts the computational burden between the numerical method and the PINN.

From the studies of Kochkov et al. [1] we know that there exist scenarios in which machine learning indeed extends the Pareto front of classical numerical methods. This paper examines the usage of machine learning to improve finite volume method approximations of fluids with high accuracy solution data. They report that their method is roughly eighty times faster than the classical finite volume method. Furthermore, their algorithm is applied locally and can therefore be scaled to be used for different initial and boundary conditions. I will reproduce these results within my studies and subsequently extend upon them to see how the performance changes as one has less or lower quality solution data.

The first and central project is an analysis of a multifidelity finite basis PINN (MFFBPINN) with a numerical method as the single-fidelity solver. For this project I would choose one differential equation to analyze–most likely the two-dimensional Navier-Stokes equation at a reasonable Reynold's number–and perform a wide array of tests on this sample problem.

$$u_t + uu_x + vu_y - \nu u_{xx} = -\frac{p_x}{\rho}$$
$$v_t + uv_x + vv_y - \nu v_{yy} = -\frac{p_y}{\rho}$$

Navier-Stokes is an ideal test problem because classical numerical methods are known to struggle simulating fluids. Furthermore, the dynamics depend on the local state of the differential equation. This means our our neural network–if trained properly–will have an easier time generalizing to problems outside of the training domain. I intend to use a finite volume method as the numerical solver. There are

two main numerical experiments I want to perform in this paper. The first experiment will be to vary the ratio of computational resources that are allocated to the numerical solver versus the PINNs while keeping the total computation time and available solution data constant. The second experiment will be to fix the structure of the MFFBPINN while changing the amount of solution data that is available. I will use the results from these experiments to address the following questions. What are the trade-offs when one is choosing between funneling computational resources into neural networks as opposed to a finer mesh? When do neural networks become a useful tool for modelling? Can MFFBPINNs correct for the biases in numerical solvers e.g. the artificial viscosity? Can I reproduce the results of Kochkov et al. [1] in the scenario where only training data is used? It would also be interesting to repeat these experiments on more complex domains where we cannot leverage grid regularity e.g. flow around a complicated shape or through an irregular pipe. How does this change the efficacy of machine learning?

The last problem I want to address is a theoretical one. Assuming that the numerical solver is sufficiently accurate, the MFFBPINN be close to the true solution even before initialization. Therefore, analyzing the linearized loss landscape should give a good approximation of where the true solution is as well as how well conditioned the problem is. Furthermore, when one trains a PINN they sample the residual at a finite number of collocation points from an uncountably infinite number of domain points. How large does the batch need to be so that we can expect that the neural network will improve upon the numerical solver?

Project Two

After the PINN experiments are done I want to repeat the studies from project one with multifidelity finite basis DeepONets. DeepONets do not work well alone. I wonder whether the aide of numerical analysis can make DeepONets viable. This is the very important experiment. Having to retrain a network for every set of initial conditions is unreasonable in a wide range of applications.

Project Three

I think it would be interesting to look at the loss landscape of the MFFBPINNs with numerical solvers. We know that at the beginning of training you are within a certain error ϵ of the true solution. This means that evaluating a linear or quadratic approximation of the neural network at initialization would

likely give one a good approximation of the loss landscape at the minima. Furthermore, there is something that has struck me as a bit strange about the PINN training process. In numerical methods the density of grid points is paramount to successful convergence. However, density of collocation points in batches for PINNs is not–at least in my experience–treated with the same importance. When one creates a batch of collocation points I believe they should be sampled at or above the Nyquist frequency of the highest frequency mode in the solution. I am curious to see whether my intuition can be tested by analyzing the loss landscape. It may also be interesting to analyze the convergence of MFFBPINNs with numerical solvers through a combination of numerical solvers and the neural tangent kernel.

References

[1] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.