FO-PINNs: A First-Order formulation for Physics Informed Neural Networks

Rini J. Gladstone
Department of Civil and Environmental Engineering
University of Illinois Urbana-Champaign
Champaign, IL 61801
rjg7@illinois.edu

Mohammad A. Nabian
NVIDIA
Santa Clara, CA 95051
mnabian@nvidia.com

Hadi Meidani
Department of Civil and Environmental Engineering
University of Illinois Urbana-Champaign
Champaign, IL 61801
meidani@illinois.edu

Abstract

We present FO-PINNs, physics-informed neural networks that are trained using the first-order formulation of the Partial Differential Equation (PDE) losses. We show that FO-PINNs offer significantly higher accuracy in solving parameterized systems compared to traditional PINNs, and reduce time-per-iteration by removing the extra backpropagations needed to compute the second or higher-order derivatives. Additionally, unlike standard PINNs, FO-PINNs can be used with exact imposition of boundary conditions using approximate distance functions, and can be trained using Automatic Mixed Precision (AMP) to further speed up the training. Through two Helmholtz and Navier-Stokes examples, we demonstrate the advantages of FO-PINNs over traditional PINNs in terms of accuracy and training speedup. FO-PINN has been developed using Modulus framework by NVIDIA and the source code for this is available in `https://developer.nvidia.com/modulus`.

1 Introduction

Physics Informed Neural Networks (PINNs) [1][5][9][10] are a class of deep learning frameworks that can solve parameterized problems governed by PDEs without any training data. The boundary and initial conditions of the system along with the governing PDEs are incorporated to the loss function. PINNs have been used successfully in solving forward and inverse problems [2][4][5][6][12]. They, however, experience a decline in accuracy as the order of the PDEs or the number of parameters increase [3][7]. This is partially due to the sharp variations in the second and higher-order derivatives that are computed by automatic differentiation. These variations destabilize the network training. Another limitation of PINNs is the soft imposition of boundary conditions compared to what is offered by other numerical methods such as finite element. Using the theory of R-functions and approximate distance functions, [11] introduced a generalized formulation for exact boundary condition imposition for PINNs. However, this method suffers from an exploding Laplacian issue for losses that involve

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

second- or higher-order derivatives. Moreover, PINNs are not suited to be trained using Automatic Mixed Precision (AMP) [8], which is widely used in training of modern deep learning models. This is because second and higher-order derivatives require a different gradient scaling that is beyond the scope of AMP, and training of traditional PINNs using AMP blows up after a few iterations due to improper scaling.

Our contributions : Addressing these shortcomings of PINNs, we propose a novel scheme which redefines PINNs as an effective solution approach for parameterized and higher-order problems via a first order formulation. We show that this first-order formulation of PINNs (1) solves the parameterized problems more accurately by smoothing out the sharp variations in the second and higher-order derivatives and by enabling exact boundary condition imposition, and (2) speeds up training by reducing the number of required backpropagations and enabling the use of AMP for training.

2 Method

2.1 First-order Physics Informed Neural Networks (FO-PINNs)

In standard PINNs, the output of the neural network consists of the dependent variables in the PDE that is being solved. Derivatives, of any order, are computed directly using these outputs by automatic differentiation. In FO-PINNs, for solving a PDE (of order d), in addition to the dependent variables, neural network output also consists of the derivatives (of order up to $d-1$) of the dependent variables. The second and higher-order PDEs are reformulated as a series of first-order PDEs with additional compatibility equations to ensure the compatibility between the predicted and exact derivatives. These compatibility equations are incorporated as additional terms in the loss function. With this formulation, automatic differentiation is used only to compute first-order derivatives of the network outputs w.r.t. inputs, and this significantly reduces the number of required backpropagations compared to standard PINNs. As an example, consider the following Helmholtz equation:

$$k^2 u + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f, \tag{1}$$

where k and f are the wave number and source term, respectively. In FO-PINNs, the first-order derivatives $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$, $\frac{\partial u}{\partial z}$ are defined as new (output) variables u_x, u_y and u_z, respectively. The Equation 1 is reformulated to produce the following governing equation on the new output variables:

$$k^2 u + \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} = f, \tag{2}$$

with the following compatibility equations:

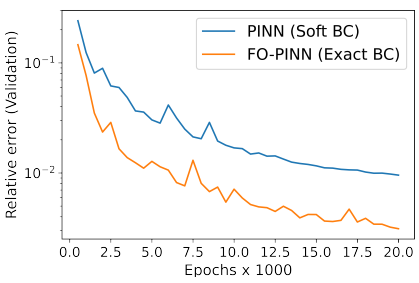
$$u_x = \frac{\partial u}{\partial x}, u_y = \frac{\partial u}{\partial y}, u_z = \frac{\partial u}{\partial z}. \tag{3}$$

Thus, the output of a FO-PINN model includes u_x, u_y and u_z as well as u. In addition to the PDE and boundary condition losses, the loss function also consists of compatibility loss terms according to Equation 3. The first order spatial derivatives, i.e. $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$, $\frac{\partial u}{\partial z}$ are calculated using automatic differentiation.

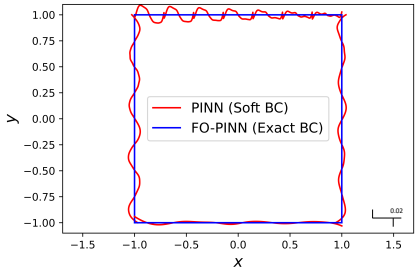
2.2 Exact imposition of boundary conditions

Exact imposition of boundary conditions (BC) in PINNs is challenging and non-trivial. Most of the purposed approaches are limited to square domains and suffer from convergence issues due to over-constrained solutions. A viable approach for complex geometries has been introduced in [11]. It consists of calculating the Approximate Distance Function (ADF) to the boundaries using the theory of R-functions, and formulating a boundary-condition and geometry-aware solution ansatz.

Let $D \subset \mathbb{R}^d$ denote the computational domain with boundary ∂D. Let $\phi(\mathbf{x})$ be the approximate distance function such that $\phi(\mathbf{x}) = 0$ for any point $\mathbf{x}$ on ∂D. For Dirichlet boundary condition, if $u = g$ is prescribed on ∂D, then the solution ansatz is given by $u_{sol} = g + \phi u_{net}$, where u_{sol} is the approximate solution, and u_{net} is the neural network output. The calculation of ADFs for various



(a) Error plot for predicted u



(b) Predicted u at the boundary

Figure 1: Results for Helmholtz equation on a square domain. The relative error is computed using the analytical solution for this example.

geometries and solution ansatz for Neumann, Robin and mixed boundary conditions are given in [11]. This approach, however, suffers from an exploding Laplacian issue at the corner of the geometries when the governing PDEs consist of second or higher-order derivatives. By bypassing second and higher-order derivative computations, FO-PINNs enable the use of this exact boundary condition imposition approach for improved solution accuracy. Furthermore, we improve the stability and time-to-convergence of the FO-PINNs, which are trained with exact BC imposition, by normalizing and non-dimensionalizing the PDE losses.

3 Results

In this section, we present two examples which demonstrate the advantages of FO-PINNs over traditional PINNs. In section 3.1, we show the accuracy gain of FO-PINNs, with exact BC imposition using ADFs, when applied to the Helmholtz problem. In section 3.2, we compare the performance of FO-PINNs over standard PINN solvers for parameterized systems governed by the Navier-Stokes equations. Both standard PINNs and FO-PINNs trained in the following examples are fully connected networks with 6 layers, 512 neurons per layer and Swish activation function. The networks were trained using the Adam optimizer. We used a single V100 GPU for training the models.

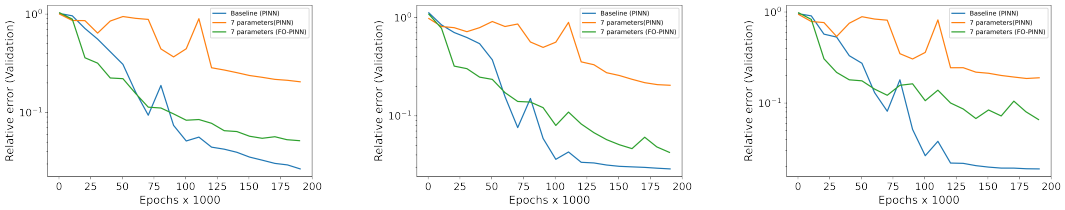
3.1 Exact BC imposition with FO-PINNs

As the first example, we solve the Helmholtz equation on a square domain with the condition $u = 0$ on the entire boundary. The exact BC is imposed using the solution from FO-PINNs by defining the solution ansatz as described in Section 2.2. From Figure 1a we observe that the validation error of FO-PINN with exact BC imposition is about one order of magnitude lower than that of PINN with soft BC imposition. It is to be noted that traditional PINNs cannot implement exact BC using ADFs due to the challenges described in section 2.2, and are thus trained with a soft BC imposition. Figure 1b shows the predicted u at the boundary for both the models. While FO-PINN ensures that u is exactly 0 for the entire boundary, PINN with soft BC predicts the boundary values with an error of magnitude 10^{-2}.

The training of FO-PINNs is 2.2x faster than that of PINNs (0.010 seconds per iteration for FO-PINN compared to 0.022 seconds per iteration for PINN with the same network architecture). This can be attributed to the reduced number of backpropagation steps required for the training of FO-PINNs. Additionally, we observed that training of FO-PINNs using AMP can achieve 1.33x speedup (0.0075 seconds per iteration) compared to FO-PINNs without the use of AMP. These two performance gains combined makes the training of FO-PINNs significantly faster compared to the training of PINNs (i.e., 2.9x speedup in this example).

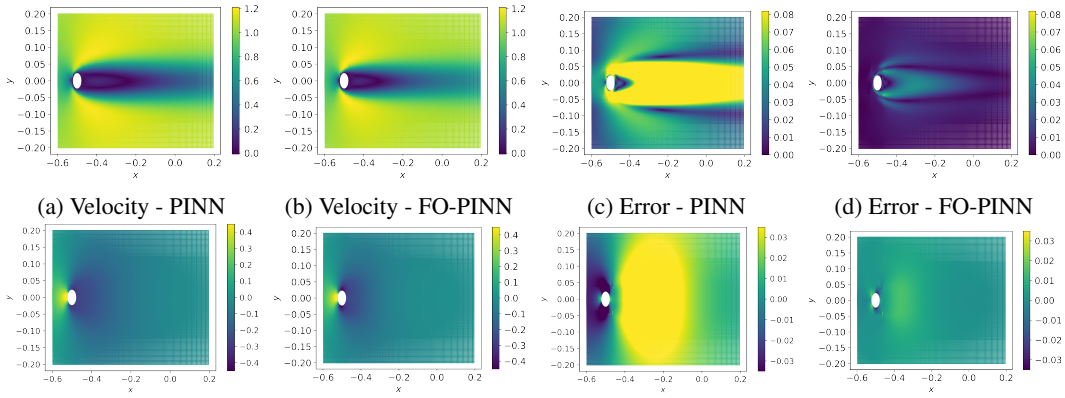
3.2 FO-PINNs for parameterized systems

To study the performance of FO-PINNs on parameterized systems, we predict the flow through a parameterized channel with a cylinder slice of radius r and height h located centrally at (a, b). The



(a) p (b) u (c) v

Figure 2: Validation error plots for the prediction of flow of in a parameterized system governed by the Navier-Stokes equations.



(a) Velocity - PINN (b) Velocity - FO-PINN (c) Error - PINN (d) Error - FO-PINN

(e) Pressure - PINN (f) Pressure - FO-PINN (g) Error - PINN (h) Error - FO-PINN

Figure 3: Predicted flow on the parameterized system (with 7 parameters) governed by Navier-Stokes equations using PINN and FO-PINN. Figures (a) through (d) show the predicted velocity magnitude from the two networks and their prediction error. Similarly, figures (e) through (h) show the predicted pressure and the prediction error.

channel has an inlet velocity, v_{in}, which is also considered as a parameter in the parameterized system. The outlet of the channel has zero pressure ($p = 0$) and no slip boundaries have $u = 0$ and $v = 0$. The flow through the channel is governed by the Navier-Stokes equations. The kinematic viscosity, ν and fluid density, ρ from the Navier-Stokes equations are also parameterized for the training of the network. Our goal is to train neural networks solvers for a parameterized system with four geometric parameters, a, b, r and h, two PDE parameters ν and ρ, and one BC parameter, v_{in}.

Let u, v, p be the three variables of interest. In addition to these variables, we add u_x, u_y, v_x, and v_y as outputs of the network. They represent the first order spatial derivatives of u and v. Additional loss terms are introduced to ensure compatibility over first order and second order derivatives. For the predicted outputs of the network $\hat{u}$, $\hat{u}_x$ and $\hat{u}_y$, the following compatibility equations are implemented through the loss functions. $\hat{u}_x = \frac{\partial \hat{u}}{\partial x}$, and $\hat{u}_y = \frac{\partial \hat{u}}{\partial y}$. Similar compatibility relationships are enforced for the predicted outputs $\hat{v}$, $\hat{v}_x$ and $\hat{v}_y$. Using high-fidelity OpenFOAM simulation results as the "ground truth", Figure 2 compares the validation error between the parameterized FO-PINN model and non-parameterized (baseline) and parameterized PINN models. We observe that the error in FO-PINN are lower by an order of magnitude compared to that of PINN for the parameterized system. The FO-PINN for the parameterized system has accuracies closer to that of the baseline, whereas PINN for the parameterized system performs significantly worse compared to that of its non-parameterized counterpart. This can be also observed clearly in the predicted results and errors for velocity and pressure as shown in Figure 3.

4 Conclusion

In this work, we presented FO-PINNs, a first order formulation of Physics Informed Neural Networks that can be used to solve second and higher-order PDEs with only first-order derivative calculations using automatic differentiation. With numerical examples involving second order PDEs, we showed how FO-PINNs can enable exact BC imposition and training with AMP. In the first example, we showed that FO-PINNs can be trained 2.9x faster compared to PINNs, and results in validation error that is one order of magnitude smaller compared to the PINN model. In the second example, we also showed that FO-PINNs provide significantly more accurate results for parameterized systems compared to PINNs. Although the results we presented are for second order PDE problems, the underlying approach is generalizable to problems that involve higher-order PDEs as well. This will involve only adding new output variables and compatibility equations, which results in a marginal increase in network parameters only in the last layer. To summarize, FO-PINNs offer two main advantages: (1) Increased accuracy for parameterized and non-parameterized problems by smoothing out the sharp variation in second and higher-order derivatives and by imposing BCs exactly, and (2) training speedup by removing extra backpropagation steps for computing the higher-order derivatives, and by using AMP for training. As future work, the performance of FO-PINNs for 3D domains and problems with higher-order PDEs will be investigated.

Broader Impact

One of the biggest promises of the PINNs in industrial applications is the capability of solving for parameterized systems in a single training, whereas traditional solvers are limited to non-parameterized simulations. These parameterized models can be used in developing industrial digital twins with real-time predictions. However, traditional PINNs suffer from accuracy decline as the dimensionality of the parameter space increases. FO-PINNs are a viable approach to address this accuracy decline, and take us one step closer to developing reliable AI solutions for industrial systems and facilitating scientific computing. Moreover, the training speedup offered by FO-PINNs opens the door to accelerated industrial design procedures and enables developing more expressive physics-informed models and more optimal hyperparameter tuning.

Acknowledgments and Disclosure of Funding

We would like to thank Dr. N. Sukumar and Dr. Ankit Srivastava for their guidance and ideas in implementing the exact boundary conditions using ADFs for FO-PINNs. Hadi Meidani and Rini Gladstone acknowledge the support by the National Science Foundation under Grant No. CMMI-1752302.

References

[1] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, pages 1–12, 2022.

[2] Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28(8):11618–11633, 2020.

[3] Stefania Fresca, Luca Dede, and Andrea Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes. *Journal of Scientific Computing*, 87(2):1–36, 2021.

[4] Han Gao, Matthew J. Zahr, and Jian-Xun Wang. Physics-informed graph neural galerkin networks: A unified framework for solving pde-governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 390:114502, 2022.

[5] Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Zhiwei Fang, Max Rietmann, Wonmin Byeon, and Sanjay Choudhry.

Nvidia simnet™: An ai-accelerated multi-physics simulation framework. In *International Conference on Computational Science*, pages 447–461. Springer, 2021.

[6] Qin Lou, Xuhui Meng, and George Em Karniadakis. Physics-informed neural networks for solving forward and inverse flow problems via the boltzmann-bgk formulation. *Journal of Computational Physics*, 447:110676, 2021.

[7] Revanth Mattey and Susanta Ghosh. A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, 2022.

[8] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[9] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8):962–977, 2021.

[10] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[11] N. Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

[12] 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, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The URL to access the code and examples are given in abstract
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]