Guaranteeing Conservation Laws with Projection in Physics-Informed Neural Networks

Anthony Baez MIT
Wang Zhang MIT
Ziwen Ma MIT
Subhro Das IBM Research
Lam M. Nguyen IBM Research
Luca Daniel MIT

Abstract

Physics-informed neural networks (PINNs) incorporate physical laws into their training to efficiently solve partial differential equations (PDEs) with minimal data [6]. However, PINNs fail to guarantee adherence to conservation laws, which are also important to consider in modeling physical systems. To address this, we proposed PINN-Proj, a PINN-based model that uses a novel projection method to enforce conservation laws. We found that PINN-Proj substantially outperformed PINN in conserving momentum and lowered prediction error by three to four orders of magnitude from the best benchmark tested. PINN-Proj also performed marginally better in the separate task of state prediction on three PDE datasets.

1 Introduction

Physics-informed neural networks (PINNs) [6] are a class of neural networks that incorporate laws of physics, e.g. partial differential equations (PDEs), to efficiently and accurately learn to solve PDEs from a small sample of data. In PINNs, the physics equation is assumed to be partially known and is incorporated into the loss function of the neural network. This creates a soft constraint where the PINN is penalized for producing outputs that violate the physical equation. This approach allows flexibility during training so the PINN can balance both fitting the observed data and conforming to the governing PDE.

This approach, however, permits PINN to violate conservation laws, which are fundamental principles of physics. A PINN that always follows conservation laws would be more accurate and trustworthy when applied to model real-world systems. Adding another soft constraint would not address this issue. Instead, we could impose a hard constraint that ensures the PINN's prediction always obeys conservation regardless of the nature of the data.

In this work, we propose a novel projection technique that can guarantee adherence to conservation laws in a PINN by projecting its output into a functional space where the conservation law is not violated. This projection approach establishes a rigid, inviolable boundary that also allows adaptability during training. The projection is applied at both training and testing, so the model learns to produce outputs that observe conservation laws and accurately represent the PDE.

To evaluate the effectiveness of this projection method, we compared a PINN model modified with the projection method, referred to as PINN-Proj, to a vanilla PINN and a PINN model with a soft constraint on the conservation law, referred to as PINN-SC. These models were trained and evaluated on three physics PDEs: the Advection equation, the viscous Burgers' equation, and the Korteweg-De Vries equation. The models were evaluated on the error of its predictions of the state u and the conserved quantity c. We found that PINN-Proj significantly outperformed PINN on the conservation task while performing marginally better than PINN on the state prediction task.

D3S3: Data-driven and Differentiable Simulations, Surrogates, and Solvers @ NeurIPS 2024.

2 Related Work

In previous work, the conservation law has been incorporated into the PINN by adding another term to the loss function, which creates a soft constraint. Training can either be done entirely with this modified loss function [8], or in two stages where the modified loss function is used in the second stage of training [3]. The physics PDE can also be completely replaced by a Hamiltonian [2] or a Lagrangian [1], which describe how the energy of a physical system is conserved.

Another branch of previous work uses hard constraints in PINNs. In topology optimization, a hard constraint can be used to keep the volume of the fluid described by the PDE below a certain value [4]. The boundary conditions of the physics PDE can be incorporated as hard constraints as well, and the addition of this constraint has been found to lower testing error [5]. A projection incorporated as a layer in a neural network has also been used as a hard constraint to guarantee that a non-PINN neural network preserves learned conservation laws in a dynamical system [9].

3 Method

3.1 Physics-informed Neural Network

The physics-informed neural network (PINN) can be defined as

$$f = u_t + \mathcal{N}[u] \tag{1}$$

where $u(x,t)$ is the solution to the equation and a function of spatial coordinate x and time coordinate t. $u(x,t)$ is parameterized by a neural network. $\mathcal{N}[\cdot]$ is a differentiable nonlinear operator that acts on $u(x,t)$. The PINN can learn a solution for the governing PDE using the loss function, $\mathcal{L}$, defined as

$$\mathcal{L} = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(x_u^i, t_u^i) - u^i|^2 + \frac{1}{N_f}\sum_{i=1}^{N_f} |f(x_f^i, t_f^i)|^2 \tag{2}$$

Here, $\{x_u^i, t_u^i, u^i\}$ are the initial and boundary data on $u(x,t)$, while $\{x_f^i, t_f^i\}$ are collocation points. The first term of $\mathcal{L}$ is the mean squared error between the PINN prediction $u(x,t)$ and ground truth u^i. The second term of $\mathcal{L}$ is the mean squared error of $f(x,t)$, as f would be zero if a perfect solution was found. We will refer to $u(x,t)$ as the state of the PDE.

3.2 Conservation Law

Let us begin with momentum as our conserved quantity. The physical systems we used with PINN are subject to the law of conservation of momentum, as they do not have inflow or outflow of momentum. If we choose a PDE where the state, u, is the velocity of a fluid, then the total momentum of the system, which we define as our conserved quantity c, at time t would be

$$c(t) = \int_X u(x,t)\,dx \approx \sum_{x\in X} u(x,t)\Delta x \tag{3}$$

where X is the spatial domain. If momentum is conserved in the system, $c(t)$ would remain constant for all times t.

3.3 Projection Method

The prediction of the state from PINN-Proj, $u_{proj}(x,t)$, is defined as

$$u_{proj}(x,t) = u(x,t) - \int_X \frac{u(x',t)}{|X|}dx' + \frac{c}{|X|} \tag{4}$$

where $u(x,t)$ is the nominal prediction of the state from PINN. In our projection equation, the second term calculates and removes the momentum of the system at time t using a dummy variable x', and

the third term adds the known conserved momentum value to the system. This projects the output of the neural network into a space where conservation of momentum is guaranteed. Now using $u_{proj}(x,t)$ as the prediction for the state, we use Equation 3 to calculate the total momentum of the system predicted by PINN-Proj,

$$\int_X u_{proj}(x,t)\,dx = \int_X u(x,t)dx - \int_X \int_X \frac{u(x',t)}{|X|}dx'dx + \int_X \frac{c}{|X|}dx \tag{5}$$

$$= \int_X u(x,t)dx - \int_X u(x',t)dx' + c = c \tag{6}$$

which is always c and is therefore always conserved.

3.4 Experimental Setup

We trained three different models: PINN, PINN-SC, PINN-Proj. A soft constraint was added to PINN-SC by adding the mean squared error between the current momentum (second term of Equation 4) and the conserved momentum (third term of Equation 4) to the loss function. A coefficient of 10 was multiplied to this term as well. The training setup is the same as the original PINN method [6] unless otherwise mentioned. The models were trained on three datasets which contained x, t, and u values generated from the Advection Equation [7], the viscous Burgers' Equation [6], and the Korteweg-De Vries Equation [6]. Each PDE dataset contained 256 x values, 100 t values, and 25,600 total state values u. All PDE datasets modeled conserved systems, so there was no input or output of momentum into the systems.

The training set contained 100 points that were randomly sampled from each PDE dataset and 10,000 collocation points calculated using Latin Hypercube Sampling. Each trial had a different random sample from its PDE dataset [1]. More details about the setup can be found in the Appendix.

3.5 Evaluation

The accuracy of PINN, PINN-SC, and PINN-Proj was measured on the metric of predicting state values, u, and on conservation of the conserved quantity, c. The errors of these tasks were referred to as Error u and Error c, respectively. The ground truth of c was the mean total momentum over all t values for each dataset. Relative $\mathcal{L}_2$ error was used for Error u, and absolute $\mathcal{L}_2$ error was used for Error c because the ground truth total momentum of all three systems is zero. The average errors across five trials were reported.

4 Results

PDE	PINN		PINN-SC		PINN-Proj	
	Error u	Error c	Error u	Error c	Error u	Error c
Advection Eq.	**2.10E-03**	3.85E-02	5.13E-03	4.17E-03	2.24E-03	**1.31E-06**
Burgers' Eq.	2.80E-03	3.02E-02	2.16E-03	2.12E-03	**1.82E-03**	**1.58E-06**
Korteweg-De Vries Eq.	3.02E-02	5.55E-01	2.34E-02	2.40E-02	**1.65E-02**	**1.65E-06**

Table 1: Average $\mathcal{L}_2$ error for state prediction task (Error u) and conservation task (Error c)

Table 1 shows the results of training PINN, PINN-SC, and PINN-Proj on our PDE datasets. On the task of predicting the state, PINN-Proj has the lowest Error u on the Burgers' and Korteweg-De Vries Equation datasets, scoring 3.4E-04 and 6.9E-03 lower than that of the next best model on that task, PINN-SC, respectively. PINN has the lowest Error u on the Advection dataset, scoring 1.4E-04 lower than that of the next best model on that task, PINN-Proj.

[1] All code is available at `https://github.com/antbaez9/pinn-proj`

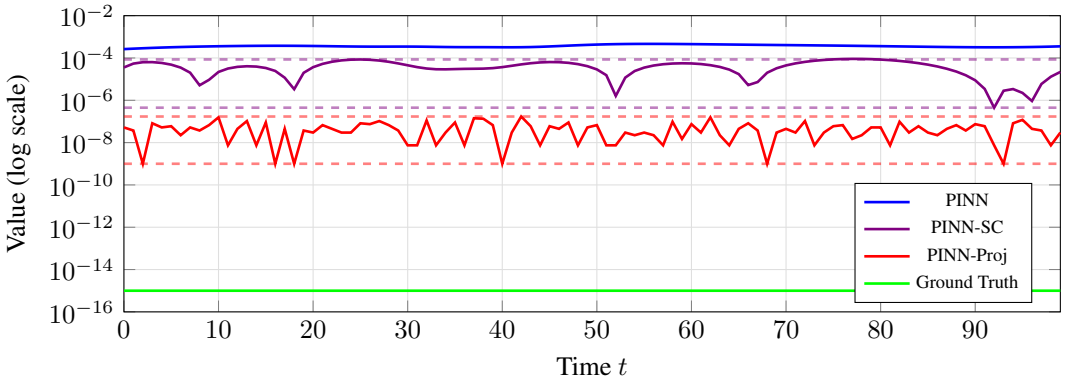

Figure 1: Predicted and ground truth of c over time for first trial of Burgers' Equation

On the task of maintaining the conserved quantity, PINN-Proj significantly outperforms PINN and PINN-SC on all three datasets. On the Advection Equation and Burgers' Equation dataset, PINN-Proj outperforms the next best model, PINN-SC, by three orders of magnitude, and on the Korteweg-De Vries Equation dataset, PINN-Proj outperforms PINN-SC by four orders of magnitude.

We can see in Figure 1 the total momentum of PINN, PINN-SC, and PINN-Proj at each time t, which are their predictions of the conserved quantity c. The c values in Figure 1 also reflect the results in Table 1, as PINN-Proj is more accurate than PINN-SC, which is in turn more accurate than PINN. The dashed lines show the upper and lower bounds for the values in PINN-SC and PINN-Proj.

5 Discussion

On Error u, PINN-Proj performs best on two datasets and PINN performs best on one dataset. Both margins that PINN-Proj performs better than the next best model are higher than the PINN's margin of outperformance. The results, therefore, demonstrate that the projection method is able to reliably and significantly improve the conservation of the conserved quantity, and even marginally improve performance on the state prediction task compared to using a soft constraint or no constraint in PINN.

Across all three datasets, Error c for the PINN-Proj method is near 1E-6. One possible cause for the error not being lower could be the discretization of the x and t domains of all datasets was done to five digits, which is just under the order of magnitude of the Error c. Another pattern in the c values is that PINN-Proj, and to a lesser extent PINN-SC, have more variation than in PINN. This variation is likely due to the many floating-point calculations during the integration in PINN-SC and PINN-Proj and is lower in PINN-SC because of the flexibility of the soft constraint. This could also be worsened by the discretization accuracy of x and t. The projection method significantly increased training time as well, since the integration involved many calculations at every epoch during training. These issues of accuracy, variability, and computational cost will be further addressed in future work.

The main limitation of our method is that it is only applicable to systems with no net input or output to the system over time, which is a special case of the Neumann boundary condition. So, future work will involve extending the projection method to general cases of different boundary conditions. We also were limited by solvers available to generate our own dataset, so we relied on previously generated datasets and benchmarks. If we were to use a dedicated solver, it would allow more control, consistency, and a wider range of PDEs to be tested.

6 Conclusion

This paper proposes a novel projection method to guarantee conservation in a PINN. To determine its efficacy, we compared a PINN that uses the projection method, PINN-Proj, to an unmodified PINN and a PINN with a soft constraint, PINN-SC. After applying these models to datasets generated from the Advection, Burgers', and Korteweg-De Vries PDEs, we found that PINN-Proj, while adding computational cost, greatly improves and guarantees conservation of momentum while marginally improving performance on state prediction in conserved systems when compared to PINN.

References

[1] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2019.

[2] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

[3] Shuning Lin and Yong Chen. A two-stage physics-informed neural network method based on conserved quantities and applications in localized wave solutions. *Journal of Computational Physics*, 457:111053, 2022.

[4] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

[5] Geoffrey Négiar, Michael W. Mahoney, and Aditi Krishnapriyan. Learning differentiable solvers for systems with hard constraints. In *The Eleventh International Conference on Learning Representations*, 2023.

[6] 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.

[7] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[8] Gang-Zhou Wu, Yin Fang, Nikolay A Kudryashov, Yue-Yue Wang, and Chao-Qing Dai. Prediction of optical solitons using an improved physics-informed neural network method with the conservation law constraint. *Chaos, Solitons & Fractals*, 159:112143, 2022.

[9] Wang Zhang, Tsui-Wei Weng, Subhro Das, Alexandre Megretski, Luca Daniel, and Lam M Nguyen. Concernet: a contrastive learning based framework for automated conservation law discovery and trustworthy dynamical system prediction. In *International Conference on Machine Learning*, pages 41694–41714. PMLR, 2023.

A Appendix

The PINN models contained 9 hidden linear layers of 20 neurons. The weights of the neural network were initialized using Xavier normal initialization. All activation functions were hyperbolic tangent. The optimizer was L-BFGS. The L-BFGS optimizer used stops itself when optimality conditions are met, so the number of epochs of training varied between trials and between datasets.

The Advection equation had a velocity of flow of 0.1 and its dataset was generated using PDEBench. The viscous Burgers' equation had a diffusion constant of 0.1. The Korteweg-De Vries equation had a dispersive term of 0.0025.