


# Physics Informed Neural Networks (PINNS)

🕒 Created	@October 18, 2023 11:40 PM
🕒 Last Edited Time	@October 19, 2023 12:14 AM
📄 Type	article
👤 Created By	 Shoumik Majumdar
🔗 Link	<a href="https://iopscience.iop.org/article/10.1088/2632-2153/acf116">https://iopscience.iop.org/article/10.1088/2632-2153/acf116</a>

As part of a research engineer at Quantiphi, I worked on an interesting application of Machine Learning called PINNs. Physics-Informed Neural Networks (PINNs) are machine learning models that integrate data-based learning with partial differential equations (PDEs). At a high level, finding the solution of partial differential equations (PDEs) using artificial neural networks is an approach normally referred to as PINNs. It has been gaining traction in research and industry across various use cases from fluid dynamics to acoustic and structural engineering - anything and everything that requires solving physical equations governed by PDEs.

The use of classical Computational fluid dynamics (CFD) solvers is widespread across the industry as the go-to solvers for these problems. While these methods are popular, they come with certain drawbacks, including high computational expenses for complex industrial issues (largely due to necessary mesh sizes) and challenges in effectively utilising external data sources, such as sensor data, to facilitate the solution of the partial differential equations (PDEs).

In this work, for the first time we extend PINNs to model the numerically challenging case of astrophysical shock waves in the presence of a stellar gravitational field.

The most common way to include physics laws in a PINN is to add to the regular neural network data loss  $L_{data}$ , the PDE residuals  $L_{pde}$ , together with the initial condition and boundary condition losses  $L_{icbc}$ , as extra loss terms in the total loss function  $L_{PINN}$ .

Then, the total PINNs loss becomes

$$L_{PINN} = L_{pde} + L_{icbc} + L_{data}$$

where in our case  $L_{pde}$  refers to the residual losses of PDE system or in other words the error within which the model satisfies the PDEs. This formulation allows for the model to leverage the powerful back-propagation and loss minimization methods of modern neural networks to learn a model that not only can match the observations, but also adheres to the fundamental physics principles.  $L_{data}$  is the rather familiar loss from data that we see in supervised learning.

We applied PINNs in three different setups ranging from modeling astrophysical shocks in cases with no or little data to data-intensive cases. Namely, we used PINNs (a) to determine the effective polytropic index controlling the heating mechanism of the space plasma within 1% error - solving the inverse problem (b) to quantitatively show that data assimilation is seamless in PINNs and small amounts of data can significantly increase the model's accuracy - scalability of the solution with more data, and (c) to solve the forward time-dependent problem for different temporal horizons - solving the forward problem.

The paper provides detailed explanation to all the experiments and results that we used to make our conclusions.

Through our experiments we demonstrate that PINNs can be used to discover underlying physics as captured in observations. In this case we recovered the correct effective  $\gamma$ -index (governed by the PDEs) which is a proxy of the solar wind heating and acceleration mechanism. Classic CFD models have been through many decades of maturity and robust benchmarking, but still to this day they are not able to seamlessly incorporate observational data beyond their initial and boundary conditions. We showcased that PINNs can be a valuable complementary modeling approach that can utilize all the available observations in a native fashion and inform the final prediction to improved accuracy.

The paper was published in the **Machine Learning: Science and Technology** journal and the paper can be found here: [Physics-informed neural networks for modeling astrophysical shocks](#)