Physics-Informed Neural Networks for modeling astrophysical shocks

Physics-Informed Neural Networks for Modeling Astrophysical Shocks

S. P. Moschou[1], E. Hicks[1], R. Y. Parekh[1], D. Mathew[1], S. Majumdar[1], and N. Vlahakis[2]

[1]Applied Research, Quantiphi, Marlborough, MA 01752, USA
[2]National and Kapodistrian University of Athens, Greece
Corresponding author: S. P. Moschou, sofiap.moschou@gmail.com

Abstract. Physics-Informed Neural Networks (PINNs) are machine learning models that integrate data-based learning with 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. Notably, PINNs suffer from competing losses during gradient descent that can lead to poor performance especially in physical setups involving multiple scales, which is the case for shocks in the gravitationally stratified solar atmosphere. 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, (b) to quantitatively show that data assimilation is seamless in PINNs and small amounts of data can significantly increase the model's accuracy, and (c) to solve the forward time-dependent problem for different temporal horizons. We addressed the poor performance of PINNs through an effective normalization approach by reformulating the fluid dynamics PDE system to absorb the gravity-caused variability. This led to a huge improvement in the overall model performance with the density accuracy improving between 2 and 16 times. Finally, we present a detailed critique on the strengths and drawbacks of PINNs in tackling realistic physical problems in astrophysics and conclude that PINNs can be a powerful complimentary modeling approach to classical fluid dynamics solvers.

1. Introduction

Shocks are ubiquitous throughout multiple astrophysical systems. In this study we focus on the solar termination shock. The termination shock marks the boundary between the supersonic and the subsonic solar wind outflow, as it's the region at which the solar wind speed decreases sharply as a result of its interaction with the interstellar medium. NASA has launched an entire fleet of satellites‡ that measure the solar wind and its impact on the heliosphere. The heliophysics satellites make in-situ measurements at specific locations around the Sun. There have been direct in-situ measurements of the

‡ For more information please see the official NASA website on Heliophysics missions `https://www.nasa.gov/mission_pages/sunearth/missions/index.html`.

plasma at the Heliopause, see e.g. [1]. The termination shock is a standing shock wave at the distance of about $\sim$ 100 AU as measured by Voyager 1 and Voyager 2 space probes [e.g. 2, 3, 4]. A complete 3D coverage of the entire heliosphere is impossible and especially for the outer parts of the solar system the observational coverage is very limited. Thus, numerical modeling and simulations are often necessary to study the solar wind evolution throughout the entire heliosphere [e.g. 5].

Machine learning and neural network models have been revolutionizing pretty much any field that deals with data. In their paper [6] first showed that given sufficient expressivity and a deterministic relationship between input and output variables, feedforward neural networks can be used as universal function approximators. Machine Learning and deep learning (neural network-based) methods have already been employed in a number of data-driven solar and space weather applications for tasks that range from principal component analysis, to classification, regression and forecasting [7, 8, 9, 10, 11, 12, 13, 14, 15]. We refer to [16] and references therein for an in depth review regarding machine learning advancements in space physics.

Physics-Informed Neural Networks (PINNs) are a special category of deep learning models that combine the power of partial differential equations (PDEs) and neural networks to solve a wide range of problems ranging from cases with no data available to data-driven approaches [17, 18, 19, 20, 21, 22]. PINNs leverage the power of automatic differentiation - a property of deep learning models [see e.g. 23, and references therein] - to combine classic physics laws with neural networks. In general, neural networks learn by example and can reach high accuracy when a lot of data are available. If we have sufficient amounts of observations, PINNs can be used to discover the underlying physics by e.g. determining the parameters of PDEs from observational data in a series of problems known as inverse problems [24, 25, 26, 27, 19, 28, 29, 30, 31]. When observational data are difficult to obtain or sparse, PINNs can leverage the power of physics laws expressed as PDEs together with initial and boundary conditions to solve what is known as the forward problem - a process that resembles closely the classic forward modeling from computational fluid dynamics (CFD) solvers for example.

Over the last decade, a number of papers have been published on the applicability of PINNs forward problems on fluid dynamics use-cases [17, 18, 20, 21, 32, 22]. While PINNs have been successful in the forward modeling of simpler physical scenarios, they still suffer from (a) low accuracy even with many training points, (b) local minima in cases of non-convex solution spaces and (b) lack of general applicability and high costs as they require extensive hyper-parameter tuning especially for complex physical setups such as compressible fluids and discontinuities [see e.g. discussions in 33, 34]. A number of approaches trying to tackle these issues have started to get explored by e.g. iteratively training the neural networks in sub-domains [33, 35, 36, 37, 38], introducing better sampling methods [20, 39], adding extra gradient boosted terms [34] or weighting terms in areas of high gradients [22], without a generally acceptable solution existing that tackles all three issues, i.e. low accuracy and local minima, discontinuities and high hyperparameter search.

In a number of recent works PINNs have been used for modeling simpler and classic shock wave problems [17, 20, 21, 22]. While there have been machine learning studies on interstellar shock prediction or Coronal Mass Ejection transit times [40, 41], there have not been any PINNs studies investigating astrophysical shocks in the presence of a gravitational field. In this paper we will use numerical simulation results for benchmarking our Physics-Informed Neural Network model of the evolution of the solar wind from the solar corona to capturing the termination shock at the edge of the heliosphere. This is the first paper where PINNs are applied to an astrophysical shock wave problem. Specifically, PINNs are used to model the standing termination shock created at the edge of the heliosphere.

In Section 2, we describe the fluid dynamic based shock wave theory (Section 2.1), the core CFD setup that we used to simulate the benchmarking PLUTO dataset (Section 2.2) and the PINNs approach (Section 2.3). In Section 3, we present the main results of this work organized in three subsections, namely the inverse problem in Section 3.1, data assimilation with PINNs Section 3.2 and the forward problem in Section 3.3. Then we move on the the Discussion Section 4, where we dive deeper into the implications of the present work in the current heliophysics context and discuss the advantages and drawbacks of PINNs. Finally, we finish with our Conclusions in Section 5.

2. Methods

2.1. Analytic Solution

The fluid dynamics conservation laws are a system of hyperbolic partial differential equations and can be written in the following general form

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} + \mathbf{S} = 0 \tag{1}$$

where $\mathbf{U}$ is the density of the physical quantity, $\mathbf{F}$ is its flux and $\mathbf{S}$ are denoting any source terms. Following this formulation, the conservative form of the set of equations describing the hydrodynamics of a compressible fluid in the gravitational field of the Sun can be written as follows

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial r} - \frac{2}{r}\rho u = 0, \tag{2}$$

$$\frac{\partial (\rho u)}{\partial t} + \frac{\partial (\rho u^2 + P)}{\partial r} + \frac{2}{r}\rho u^2 + \rho \frac{GM_\odot}{r^2} = 0, \tag{3}$$

$$\frac{\partial}{\partial t}\left(\frac{\rho u^2}{2} + \frac{P}{\gamma - 1}\right) + \frac{\partial}{\partial r}\left(\frac{\rho u^3}{2} + \frac{\gamma}{\gamma - 1} P u\right) + \frac{2}{r}\left(\frac{\rho u^3}{2} + \frac{\gamma}{\gamma - 1} P u\right) + \rho u \frac{GM_\odot}{r^2} = 0, \tag{4}$$

where ρ is the fluid density, u is the velocity, P is the pressure, r is the radial distance from the Sun, $M_\odot$ is the solar mass and G is the gravitational constant.

The above system of equations describes the acceleration of the solar wind near the Sun where the flow becomes supersonic, as well as the sharp deceleration near the edge

of the heliosphere caused by the interaction with the interstellar medium. As discussed in the introductory Section 1, a termination shock is formed, and marks the boundary between the supersonic and the subsonic solar wind outflow. In this paper, we make the assumption of a spherically symmetric solar wind outflow to reduce the problem dimensionality from 3D to 1D and simplify the problem.

There are no explicit heat source terms in the above set of equations (2)-(4). However, in order to properly simulate the solar wind acceleration we need to include some form of energy input into the system. In order to avoid the numerical complexity of explicitly including heating and cooling terms in the analytic and numerical models we approximate the role of the heating through an effective polytropic index γ. The possible range of values for the polytropic index of a natural fluid is γ is $1 \leq \gamma \leq 5/3$. In particular, for ideal gases where there is no exchange of energy between the gas and the environment the polytropic index is $\gamma = 5/3$. We can however use an effective polytropic index ($\gamma < 5/3$) that mimics the effects of adding heating in the system. In our case we use an effective polytropic index of $\gamma = 1.1$.

The solar wind corresponds to a steady-state solution of the above system of differential equations (2) - (4). In fact, we can analytically integrate these equations to find three integrals. The first is the conservation of mass $\rho u r^2 = \rho_0 u_0 R_\odot^2$, which gives a relation between the mass flux at an arbitrary distance r as a function of the mass flux at the solar surface $R_\odot$ (quantities at the solar surface are denoted with the subscript 0). The second is the entropy conservation $P/\rho^\gamma = P_0/\rho_0^\gamma$ that holds as long as the flow is smooth (without discontinuities). The third is the Bernoulli integral

$$E = \frac{u^2}{2} + \frac{\gamma}{\gamma - 1}\frac{P}{\rho} - \frac{GM_\odot}{r}. \tag{5}$$

With some algebraic manipulations we can obtain the following differential equation for the radial profile of the velocity,

$$\frac{du}{dr} = \frac{u}{r}\frac{2c_s^2 - GM_\odot/r}{u^2 - c_s^2}, \tag{6}$$

where $c_s = \sqrt{\gamma P/\rho}$ is the sound speed. This equation shows that as the wind accelerates it crosses a sonic critical point (where $u_c = c_s$) at distance $r_c = GM_\odot/2c_s^2$. We can find parametric solutions of the above equation as a function of the value of the Bernoulli integral. Using the expressions of the integrals the parametric solutions can be computed and visualized as isocontours of the following equation

$$\frac{M^2}{2} + \frac{1}{\gamma - 1}\left(\frac{M_0}{x^2 M}\right)^{\gamma - 1} - \frac{\lambda}{x} = \frac{M_0^2}{2} + \frac{1}{\gamma - 1} - \lambda, \tag{7}$$

where $M = u/c_{s0}$ is the Mach number (the velocity in units of the sound speed at the surface of the Sun), M_0 its value at the solar surface, $x = r/R_\odot$ is the distance from the center of the Sun in units of the solar radius $R_\odot$, and $\lambda = \frac{GM_\odot}{R_\odot c_{s0}^2}$ is a dimensionless constant. This is essentially a mathematical reformulation of the dynamic solar wind solution [42, 43, 44, 45, see e.g.] and we can use it to obtain all the mathematically

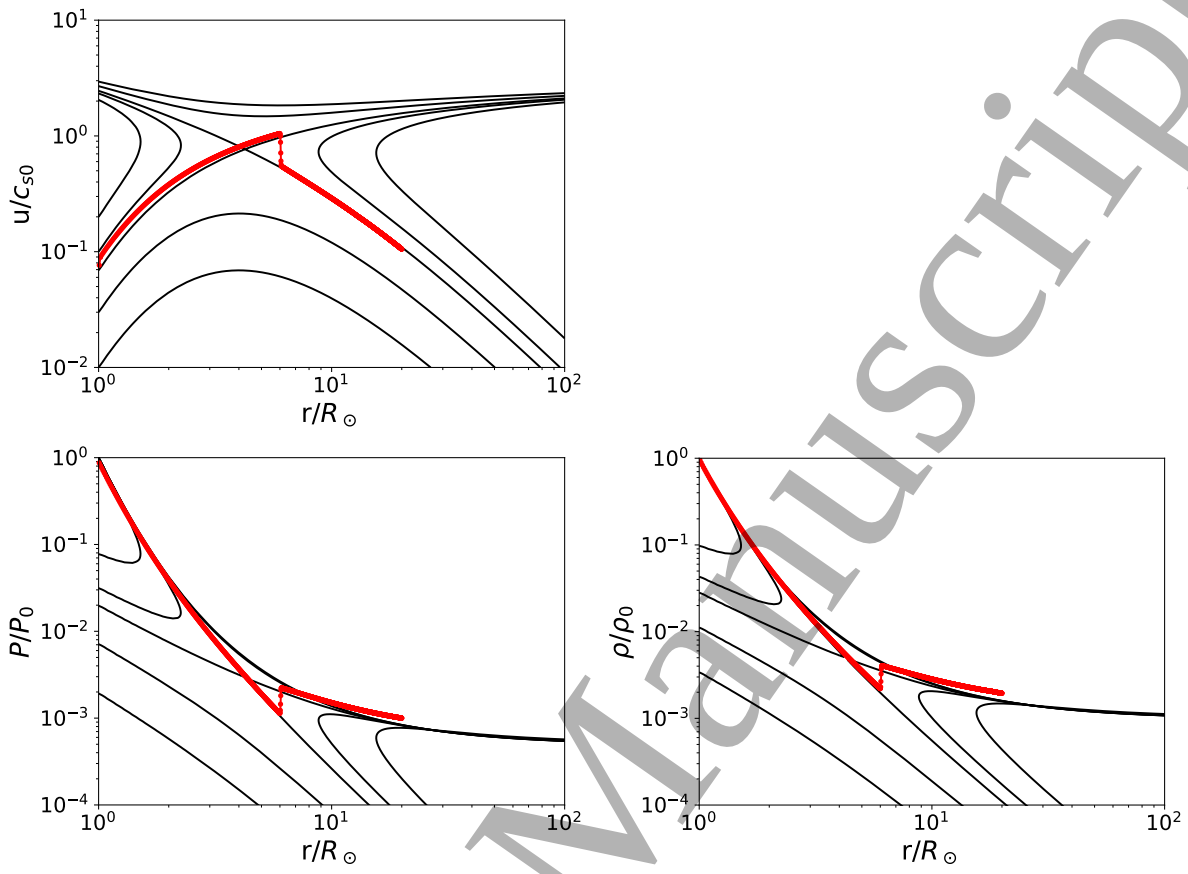

Figure 1. Isocontours with the same Bernoulli integral value of the analytic families of steady-state solutions to the system of PDEs (black curves) and PLUTO simulation results of the entire solar wind solution, which forms a shock and jumps to a new solution branch (red line with points). These analytic families of solutions correspond to a parameter of $\lambda = 5$.

acceptable families of steady state solutions of the Equation (6). The isocontours and analytic solutions of the PDE system (2)-(4) can be seen in Figure 1.

The only physically acceptable solution that connects the subsonic wind near the Sun with infinity is the one that passes through the critical point, since this is the only one that becomes supersonic [42, 43, 44, 45] and can be matched through a standing shock with a low pressure medium at high distances. The conditions at the critical point give the velocity at this point $u_c/c_{s0} = (4M_0/\lambda^2)^{(\gamma-1)/(5-3\gamma)}$, its position $r_c = GM_\odot/2u_c^2$, and the corresponding value of the Bernoulli constant $E_c = \frac{5-3\gamma}{2(\gamma-1)}u_c^2$. Combining these three equations with the equation for the isocontours of equation (7), we find that for the critical solution the speed at the solar radius is the solution of the following equation

$$\frac{M_0^2}{2} + \frac{1}{\gamma-1} - \lambda = \frac{5-3\gamma}{2(\gamma-1)}(4M_0/\lambda^2)^{2(\gamma-1)/(5-3\gamma)}. \quad (8)$$

Since M_0 is significantly smaller than unity we can ignore the first term and get the

approximate solution

$$M_0 \approx \frac{\lambda^2}{4}\left[\frac{2-2\lambda(\gamma-1)}{5-3\gamma}\right]^{(5-3\gamma)/(2\gamma-2)}. \tag{9}$$

Now, if we consider that the solar wind cannot flow to infinity unobstructed, but instead will at some point encounter the interstellar medium that has some pressure of its own $P_{\mathrm{r_{max}}}$, we get the following. For a given pressure of the environment at distance from the Sun $r = r_{max}$, i.e. the interstellar medium, $P_{\mathrm{r_{max}}}$ a shock will be formed at the position where the ram pressure of the wind $P_{ram} = \rho u^2$ roughly equals the pressure of the interstellar medium $P_{ram} = P_{\mathrm{r_{max}}}$. This is expected to happen at a large distance where the wind speed is close to its terminal value $u_\infty = \sqrt{2E_c}$ and using the conservation of mass we get

$$r_{\mathrm{shock}} \sim R_\odot \sqrt{\frac{\rho_0 c_{s0}^2}{P_{\mathrm{r_{max}}}}} \left(\frac{5-3\gamma}{\gamma-1}\right)^{1/4} \left(\frac{2}{\lambda}\right)^{(\gamma-1)/(5-3\gamma)} M_0^{(2-\gamma)/(5-3\gamma)}, \tag{10}$$

which is a rough estimation of the location of the standing shock. Using the approximate solution for M_0 we get

$$r_{\mathrm{shock}} \sim R_\odot \sqrt{\frac{\rho_0 c_{s0}^2}{P_{\mathrm{r_{max}}}}} \left(\frac{5-3\gamma}{\gamma-1}\right)^{1/4} \frac{\lambda}{2}\left(\frac{2-2\lambda(\gamma-1)}{5-3\gamma}\right)^{(2-\gamma)/(2\gamma-2)}, \tag{11}$$

which is roughly the expected location of the standing shock as a function of the conditions at the solar surface and the pressure at infinity.

The shock jump conditions state that the fluxes of mass, momentum and energy before and after the shock remain the same

$$[\rho u] = 0, \qquad \left[\rho u^2 + P\right] = 0, \qquad \left[\frac{\rho u^3}{2} + \frac{\gamma}{\gamma-1} P u\right] = 0, \tag{12}$$

where the square parenthesis represent the mathematical difference between the fluxes in the parenthesis before and after the shock. Coming back to Figure 1 of the solution-space for the solar wind and considering the jump conditions, then we expect that the steady state solution of the termination shock is reached once the pressure balance between the solar wind and the interstellar medium has been reached and the new state still respects the conservation laws. Given that the Bernoulli constant remains the same, thus the shock solution at some point $r = r_{shock}$ after the critical point moves to the decreasing branch of the solution space shown in Figure 1 with energy E_c, if the shock is standing. This solution is represented with the red points in Figure 1 and it was obtained numerically using a simulation algorithm as we will describe in the section below.

2.2. Synthetic Dataset

In the previous section, we determined analytically the characteristics of the expected solution for a standing shock. The complete solution follows the solar wind accelerating branch, that passes from the critical point, and at position $r \sim r_{shock}$ jumps to the

decreasing-with-distance family of solutions as it feels the effects of the interstellar medium's pressure $P_{r_{max}}$. We cannot determine analytically the precise location of the shock jump, and for that we need to run a numerical simulation. In order to obtain benchmarking data for training a Neural Network we use the astrophysical code PLUTO [46]. PLUTO is used for simulating the solar wind as a single fluid under the ideal hydrodynamics approximation.

PLUTO, similar to other widely used CFD codes, solves the hydrodynamics equations in their non-dimensional form. For that non-dimensional "code" units are used to avoid numerical issues due to extremely low or large numerical values of the quantities of interest. All physical quantities are normalized with respect to the characteristic scales of the system in question. For this non-dimensionalization process we need to specify three units and express all other quantities with respect to the chosen three. For convenience we chose the following characteristic units (a) the length unit is chosen to be the radius of the Sun $r_0 = R_\odot = 6.957 \times 10^{10}$ cm, (b) the density unit is taken as the density at the base of the solar corona (approximately at the solar radius) $\rho_0 = 0.2 \times 10^{-6}$ g cm^{-3}, and (c) the velocity unit as the sound speed at the base of the corona $u_0 = c_{s0}$. Then if we measure time in units $t_0 = r_0/u_0$, and pressure in $P_0 = \rho_0 u_0^2$, the above equations (2)-(4) become dimensionless and the $GM_\odot$ term should be replaced with the dimensionless constant $\lambda = \frac{GM_\odot}{r_0 u_0^2}$. It's worth noting that the code units used for time in PLUTO are given as derivative units, i.e. as a function of the length and velocity units $t_0 = r_0/u_0$. For our case r_0 is chosen as the solar radius $R_\odot$, and u_0 is the sound speed at the solar surface. This dimensionalization of fluid dynamic equations allows us to apply our solution to different stars provided they have the same non-dimensional parameters, which in our case is the parameter λ. In our case, the parameter λ used corresponds to a value of $\lambda = 5$.

For this simulation, we use 1D spherical geometry assuming a spherically symmetric solar wind. The geometric domain extends between [1, 20] $R_\odot$ and there is a mesh of 1,500 points therein. The time domain covered extends between [0, 200] t_0 until the solution convergences to the steady state. The PDEs are solved with an adaptive time-step computed through the Courant condition and an adaptive mesh refinement of 4 levels with equal weights between velocity, density and pressure $u : \rho : p$, i.e. $1 : 1 : 1$. The initial conditions at r_0 are $u_i = 2\ u_0$, $\rho_i = 10^{-3}\ \rho_0$ and $p_i = 10^{-3}\ p_0$ and the boundary conditions at $r_{min} = R_\odot$ and $r_{max} = 20\ R_\odot$ are set to

$$\left(\frac{\partial u}{\partial r}\right)_{r_{min}} = 0, \tag{13}$$

$$\rho_{r_{min}} = \rho_0, \tag{14}$$

$$p_{r_{min}} = \frac{1}{\gamma} p_0 \tag{15}$$

and

$$\left(\frac{\partial u}{\partial r}\right)_{r_{max}} = 0, \tag{16}$$

$$\left(\frac{\partial \rho}{\partial r}\right)_{r_{max}} = 0, \tag{17}$$

$$p_{r_{max}} = 10^{-3} p_0. \tag{18}$$

The PLUTO simulation starts at $t = 0$ and after several crossing-times pass and nearly 75,000 solver iterations the propagated disturbances from the boundary conditions settle and we finally arrive at a steady state solution with $t_{final} = t_{ss} = 200\, t_0$. At the end of the simulation the shock stops travelling and relaxes at its standing location at a distance of $r_{shock} \sim 6\ R_\odot$ as shown by the red line of Figure 1.

2.3. Physics-Informed Neural Network Models

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

$$\mathcal{L}_{PINN} = \mathcal{L}_{pde} + \mathcal{L}_{icbc} + \mathcal{L}_{data}. \tag{19}$$

where in our case $\mathcal{L}_{pde}$ refers to the residual losses of PDE system (2)-(4) 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.

In this work, we employ PINNs [17, 18] for the study of astrophysical shocks and in particular for the heliospheric termination shock. For our neural network models we use DeepXDE, a Python-based library for PINNs, which was first introduced and open sourced by [39]. We chose DeepXDE because it is user-friendly, while allowing for multiple standard machine learning backends, such as TensorFlow or PyTorch. For our experiments, we used TensorFlow 2.10 as the backend and hard boundary conditions [39] for all three physical quantities, namely velocity u, density ρ, and pressure P, to match the PLUTO setup as discussed in Section 2.2. In terms of the neural network architecture, we have used fully connected neural networks (FNN). Finally, we chose a hyperbolic tangent as our activation function, a Glorot uniform initializer and an Adam optimizer [47] with a cosine learning rate scheduler for all the experiments presented in this work. The best network architecture, size and parameters were determined after experimentation.

In this work, we present three different ways PINNs can readily contribute to the study of the solar termination shock. Those three ways are (a) the inverse problem, where PINNs are used to discover the underlying physics from data (Section 3.1), (b) the data assimilation, where PINNs can seamlessly incorporate ad-hoc observations to increase their predictive accuracy (Section 3.2), and (c) the forward problem, where PINNs get to solve PDEs similar to classic numerical solvers in the forward modeling (Section 3.3).

3. Results

3.1. Inverse Problem

In this section, we use PLUTO data to determine the effective polytropic index value that is a parameter in the PDE equations (2)-(4). Determining an effective polytropic index is equivalent to determining the nature of the heating process present in the solar wind directly from data. In the following experiments, we aim to solve the inverse problem, wherein we use a PINN to model the underlying physics and infer the polytropic index γ in a supervised fashion, i.e. from observations or PLUTO data in this case.

Table 1. Inverse experiments with E_γ % denoting the percentage error of the predicted $\hat{\gamma}$ with respect to the true γ.

Exp. No.	Time domain	Points	Epochs	$\mathcal{L}_{pde}$	$\mathcal{L}_{data}$	$\hat{\gamma}$	E_γ %
1	$t > 0$	40k	60k	7.44e-06	5.62e-05	1.2	9.1
2	$t > 0$	60k	60k	6.91e-06	4.61e-05	1.2	9.1
3	$t > (t_{ss}/2)$	10k	60k	8.01e-07	2.62e-06	1.11	0.9
4	$t > (t_{ss}/2)$	20k	60k	7.96e-07	6.22e-06	1.11	0.9
5	$t > (t_{ss}/2)$	40k	60k	5.61e-07	5.07e-06	1.1	0.0
6	$t > (t_{ss}/2)$	60k	60k	9.70e-07	4.34e-06	1.12	1.8
7	$t > (t_{ss}/2)$	75k	60k	8.48e-07	3.38e-06	1.11	0.9
8	$t = t_{ss}$	1.5K	60k	1.16e-07	1.18e-07	1.09	0.9

For our first experiment on the inverse problem (i.e. inferring $\hat{\gamma}$ from data), we train a fully connected neural network with an input layer with 2 neurons (t, r), 10 hidden layers consisting of 64 neurons each and a output layer with 3 neurons (u, P, ρ). We use the mean squared error (MSE) loss function and train the model using the ADAM optimizer over 60,000 iterations. To train this model, we sample 40,000 data points from the PLUTO synthetic dataset and 10,000 data points that are randomly sampled from the entire spatiotemporal domain from $t > 0$ to the end of the simulation and until the steady state is reached $t = t_{ss}$. The results for this experiment can be seen in Figure 2, where the top panel represents the PDE losses, the middle panel represents the data losses, and the third and bottom panel represents the convergence of $\hat{\gamma}$ to the ground truth value $\gamma = 1.1$. During training, both data and PDE losses show a decreasing trend with the predicted $\hat{\gamma}$ reaching as low as $\hat{\gamma} = 1.20$. This results in a percentage error of the order of $\sim 9\%$ deviation from the ground truth $\gamma = 1.1$. This experiment corresponds to experiment No. 1 in Table 1 and the accuracy of the model's learnt intermediate mapping between input and output variables can be visually inspected in Figure 3.

To further improve the model and since neural networks are typically more accurate if trained with more training data, we run another experiment corresponding to No. 2 in Table 1 with the same hyperparameters but with 20,000 more data points sampled

from the entire $t > 0$ PLUTO dataset. The experiment did not result in significant improvements to the predicted $\hat{\gamma}$ value and instead we once again arrived at an error of about 9%. This average error is computed at the final time-step, i.e. at the steady state, but it's a measure of the ability of the PINN to predict the time-dependent evolution of the fluid throughout the entire temporal domain $t > 0$. The accuracy of the model's predicted polytropic index $\hat{\gamma}$ depends on the accuracy of the intermediate mapping of the input (t, r) to the output variables (u, ρ, P). We will explore this further in the rest of this section.

As part of our next set of experiments, we use the same model architecture and training hyperparameters, but now we sample 40,000 points from the second half of the transient phase until the steady state is reached from the PLUTO data, i.e. from $t > (t_{ss}/2)$ to $t = t_{ss}$. The intuition behind choosing a later timestep is to train the model with data when the system has approached closer to the steady state, thus leading to less variance in the training data distribution. This corresponds to experiment No. 3 in Table 1 and the visual representation of the results can be seen in Figure 2. Clipping half of the time-domain to exclude the first half of the transient phase led to better model performances with our losses and the predicted polytropic index $\hat{\gamma}$ converging significantly faster and falling within less than $< 1\%$ error at $\hat{\gamma} = 1.11$ for the majority of these experiments in the data sampling ranges explored, see experiments No. 3-7 in Table 1.

As a final experiment, we use the same architecture and training hyperparameters, but this time we sample 1.5k datapoints from PLUTO where $t = t_{ss}$ as can be seen in experiment No. 8 in Table 1. For this experiment we consider a single timeframe when the system has reached the steady state which means there is no time-variability or in other words this corresponds to the least variance experiment. Even with significantly less data (2.5 % of previous experiments) our model converges to less that $< 1\%$ error at $\hat{\gamma} = 1.09$ within the same number of iterations. The faster convergence of this model to the predicted polytropic index value $\hat{\gamma} = 1.09$ can be seen in Figure 2.

The above results lead us to conclude that the PINN has an easier time learning an accurate mapping between (t, r) and (u, ρ, P) in physical situations where the variance in the predicted quantities is lower. A smoother semi-steady state or slowly changing with time flow leads to a more accurate prediction. This is further substantiated and demonstrated in Figure 3. The experiment with data sampled from the full time-domain $t > 0$ (red curve) has a larger error in the predicted output variables ($\hat{u}$, $\hat{\rho}$, $\hat{P}$) than the experiment with data sampled from the second half of the PLUTO simulation (blue curve) for the same data, i.e. 40,000 sample points. Further evidence for the same trend is the fact that the last experiment (No. 8 in Table 1) with only 1,500 sample points all sampled from the last timeframe of the PLUTO data corresponding to the steady state has the lowest error of predicting the output variables ($\hat{u}$, $\hat{\rho}$, $\hat{P}$).

3.2. Data Assimilation

In this section, we focus on data assimilation use-cases where we use a small number of training points to help the PINN converge faster in forward modeling. In the rest of this section, we present a few experiments demonstrating the importance of the assimilated data sampling on the PINNs performance in predicting the PLUTO ground truth steady state solution. The setups and the results of these experiments are shown in Table 2.

Table 2. Data assimilation with PINNs

Exp	Points	Sampling	$\mathcal{L}_{pde}$	$\mathcal{L}_{data}$	E_u %	E_p %	E_ρ %
A	2	shock left and right	2.64e-05	1.55e-08	8.83	6.66	10.56
B	10	linear	1.77e-05	8.92e-06	2.71	8.93	10.64
C	10	error weighted	2.03e-05	7.77e-08	1.52	5.94	4.10

For all the experiments of this section, since we are only interested in the steady state solution we take the time-independent version of the (2)-(4) PDE system, i.e. we ignore the time derivatives. On top of that, we set the r_{min} and r_{max} boundary conditions to hard constraints instead of including them in the total loss function [39]. Hard constraints are a last linear layer of the neural network that enforces the boundary conditions. The reason for setting the boundary conditions into hard constraints is twofold (a) we want to imitate the way classic solvers solve the PDEs as close as possible and in PLUTO for example the boundary conditions are given and they drive the entire solution dynamics and thus variations are not permitted and (b) we want to relieve the PINN from the added difficulty of optimizing another six losses, i.e. losses for all three (u, P, ρ) at r_{min} and r_{max}.

For all the experiments on data assimilation, we train a fully connected neural network with an input layer with 1 neuron (r), 5 hidden layers consisting of 100 neurons each and a output layer with 3 neurons. We use the mean squared error (MSE) loss function and train the model using the ADAM optimizer over 60,000 iterations. To train this model, we sample 40,000 data points that are randomly sampled from the spatial domain.

Experiment A In the first experiment we used exactly two data points from the PLUTO steady state data, one just before the shock and one just after the shock. This is experiment A in Table 2. The intuition being that giving specifically these two data points should help the network to accurately capture the standing shock region. The results for this experiment can be seen in Figure 4. As shown in Table 2, the overall percentage errors for the predicted fluid quantities for experiment A are $E_u = 8.83\%$, $E_p = 6.66\%$, $E_\rho = 10.56\%$, where E_u, E_p, and E_ρ are the velocity, pressure and density percentage errors respectively.

In the top left panel of Figure 4 we see the velocity which seems to be predicted correctly near both ends of the computational domain, but the predicted $\hat{u}$ peak is

higher than the ground truth and the predicted curve shows some small divergence in the central 50 % of the computational domain around the shock area. The shock jump is captured accurately in the the narrow discontinuity area. The pressure is shown in the middle left panel of Figure 4 and is the one with the lowest error. The pressure profile is predicted accurately in the vast majority of the computational domain including the shock area with only diverging in the near areas before and after the shock. Finally, in the middle right panel of Figure 4 we see that the density $\hat{\rho}$ is accurately predicted from the beginning of the computational domain up until a short distance before the shock wave. The shock jump is captured accurately thus verifying our hypothesis that data sampling near the shock area helps capture the shock dynamics. Further from the shock area, we see that density does a couple of oscillations around the ground truth value from the shock to the end of computational domain at $r = 20R_{\odot}$.

In conclusion, with only 2 training data points and hard boundary conditions the PINN model captured the steady state solution with about 10 % error in the fluid quantities after 60,000 epochs. In this case we carefully chose the data points near the shock wave in order to help the neural network to capture the discontinuity.

Experiment B While it's a good idea to choose the training data points carefully and place them at the areas of interest, i.e. the shock wave in our case, in the general case we might not know where the shock is. In fact, in a real world scenario where we want to model the termination shock, we would use as much satellite data as possible from all available missions in the solar system without knowing in advance the position of the shock and without having the control over the spacecraft coverage of the solar system space.

In this experiment, we pick ten data points from the PLUTO steady state data linearly spaced throughout the entire computational domain. Five equally spaced points before and after the shock. This is experiment B in Table 2 and the results for this approach can be seen in the left column panels of Figure 5. Providing more points helped the network converge closer to the ground truth solution obtained by PLUTO as can be seen in Figure 5. We also see a reduction in the percentage error of the velocity and an increase in the percentage error of the pressure as we can see in Table 2. The overall percentage errors for the predicted fluid quantities for experiment B are $E_u = 2.71\%$, $E_p = 8.93\%$, $E_\rho = 10.64\%$.

The velocity at the top left panel of Figure 5 seems to be predicted correctly at the bigger part of the computational domain including the shock region, but the predicted $\hat{u}$ peak is marginally higher than the ground truth, but lower than the experiment A predicted velocity. The only area diverging from the ground truth are the regions between the shock points and the neighbouring points on both sides of the shock. In the middle left panel of Figure 5 we can see the pressure, which has a very similar behavior like that one of experiment A. Finally, in the bottom left panel of Figure 5 we see that the density $\hat{\rho}$ is accurately predicted from the beginning of the computational domain up until a short distance before the shock wave. The shock jump is captured

accurately thus verifying once more that appropriate data sampling near the shock area helps capture the shock dynamics. Further from the shock area, we see that density shows several oscillations around the ground truth value passing through all 5 data points that are between the shock and the end of the domain.

The overall retrieval of the ground truth is better in this case where we have sampled data points throughout the computational domain compared to the previous experiment. However, we still see that the losses in pressure and density are around 10 %.

Experiment C This is experiment C in Table 2. In experiment B the density showed oscillations at larger distances from the Sun, as it dropped in magnitude with inverse square law and the MSE loss had trouble predicting accurately quantities like the density and the pressure with values much smaller than unity and as low as $\rho_{r_{max}} \sim 10^{-3}$. Given that, in experiment C we decided to use 10 points again, but to rearrange them so that we now place the data points in the areas where experiment B gave higher errors. In this case, we give more weight in the areas before and after the shock by adding more points and we also upsample the area with the lower density values to rectify for the oscillations at larger distances. This way only 3 points are sampled before the shock and 7 points are sampled after the shock. The results for this approach can be seen in the right column panels of Figure 5. As shown in Table 2, the overall percentage errors for the predicted fluid quantities for experiment C are $E_\rho = 4.10\%$, $E_u = 1.53\%$, $E_p = 5.94\%$, which are the lowest errors and best results for all the experiments presented so far.

The velocity as seen in right top panel of Figure 5 seems to be predicted correctly everywhere apart from the area around the $r = 5R_\odot$ distance. The velocity has the lowest relative error compared to all three predicted profiles. The predicted pressure is shown in the middle right panel of Figure 5 and it appears to follow the PLUTO solution closely throughout the entire domain of interest including the shock region. Finally, the density $\hat{\rho}$ as seen in the bottom right panel of Figure 5 is the most accurate prediction and very close to the ground truth solution everywhere apart from a small area near the $r = 10R_\odot$ distance. The shock jump is also captured accurately similar to the previous experiments.

3.3. Forward Problem

The forward modeling refers to using a set of PDEs and initial and boundary conditions in order to obtain an accurate solution for the dynamics of the fluid. Similar to classic CFD solvers, PINNs can also be used for forward modeling [e.g. 17, 39].

In this case, we are modeling a shock wave, which appears as a discontinuity or jump in the radial profiles for all the physical quantities (u, P, ρ). Neural networks have numerical trouble predicting discontinuous functions and regions of high gradients. For that reason, we introduced a PINNs weight term similar to what was done in [22] in order to capture the shock more accurately and to allow for faster convergence. The

[22] weight λ_{WE} has the following form, which we mention here for completeness

$$\lambda_{WE} = \frac{1}{\epsilon(|\nabla \cdot \hat{u}| - \nabla \cdot \hat{u}) + 1}, \tag{20}$$

where $\hat{u}$ is the predicted velocity and ϵ is the parameter that controls the strength of the down-weighting. The weighting term reduces the importance of the PDE residuals near the shock compression area ($\nabla \cdot \hat{u} < 0$), thus allowing the neural network to gradually learn to predict the smooth regions before and after the shock with high accuracy and shrinking their distance until a thin discontinuity region is formed.

In this series of experiments, we train a fully connected neural network with an input layer with 2 neurons (t, r), 5 hidden layers consisting of 100 neurons each and a output layer with 3 neurons (u, P, ρ). We use the mean squared error (MSE) loss function and train the model using the ADAM optimizer over 60,000 iterations. To train this model, we sample 40,000 data points that are randomly sampled from the spatial domain. Finally, we use hard constraints for the boundary conditions. The setups and the results of these experiments are shown in Table 3.

Table 3. Forward problem with PINNs. The first four experiments correspond to the forward time-dependent problem with output target variables (u, P, ρ), whereas the bottom three cases correspond to the reformulated (u, y_1, y_2), with $y_1 = r^{2\gamma}P$ and $y_2 = r^2\rho$.

t_{init} (t_0)	Outputs	$\mathcal{L}_{pde}$	$\mathcal{L}_{ic}$	E_u %	E_p %	E_ρ %
100	(u, P, ρ)	6.52e-06	3.24e-06	61.05	136.44	168.58
140	(u, P, ρ)	6.99e-05	5.37e-06	12.81	38.91	94.99
150	(u, P, ρ)	9.45e-05	6.19e-06	15.84	35.80	67.23
190	(u, P, ρ)	1.20e-04	7.78e-06	1.37	14.90	12.43
100	(u, y_1, y_2)	7.94e-06	8.53e-06	12.08	12.78	10.50
150	(u, y_1, y_2)	8.66e-06	7.77e-06	8.88	11.25	13.85
190	(u, y_1, y_2)	1.28e-05	1.19e-05	2.88	5.60	6.11

During this study we perform a number of forward experiments in the time-dependent case. In the left column panels of Figure 6 we see the results of these experiments for all three dependent variables (u, P, ρ). In particular, solving the forward problem with PINNs is a computationally challenging task, especially for cases like this one, with complex physics, such as discontinuities in the presence of a gravitationally stratified atmosphere. The problems are especially evident when the time horizon we need to integrate the PDEs towards is long, which is the case here. This is quantitatively showcased in the left column panels of Figure 6, where we see a collection of experiments each of which uses as initial condition one of the intermediate steps computed by PLUTO and solves the forward problem until $t = t_{ss} = 200$. As we can see in those plots and in Table 3, we start with a very poor prediction of the dependent variables (u, P, ρ) for the first experiment with $t_{init} = 100$ and gradually as we use an initial condition closer and closer to the steady state, we approach closer to the ground truth solution.

While the velocity is showing a qualitatively and quantitatively improved convergence to the ground truth data (see Figure 6 and Table 3) as we get closer to the steady state t_{ss}, the density and the pressure show oscillatory behavior with large relative amplitudes locally. This is due to the fact that density and pressure drop drastically with distance roughly following an inverse square law $\rho \propto r^{-2}$ due to the gravitational field of the Sun. In order to boost the model performance we define two new quantities y_1 and y_2 and reformulate the PDEs (2) -(4) to solve for these new quantities, with $y_1 = r^{2\gamma}P$ and $y_2 = r^2\rho$. The results of these experiments can be seen in the right column panels of Figure 6 and the bottom 3 rows of Table 3. For most experiments with this reformulation we got a huge performance improvement in the overall relative errors which ranges from a factor 2 for the $t_{init} = 190$-couple of experiments to a factor 16 for the $t_{init} = 100$ case.

4. Discussion

In recent years, PINNs have been successfully used for tackling a range of fluid dynamics forward and inverse problems amongst which, there has also been a handful of studies modeling shock waves [20, 21, 22]. This is the first paper that uses PINNs to model the heliospheric termination shock from the base of the solar corona to the edge of the heliosphere including the gravitational force and a heating proxy.

4.1. Generality Preserving Assumptions

Accurately capturing the time evolution of the solar wind from the solar corona to the edges of the heliosphere requires a complex multi-physics multi-scale model, combined with multi-messenger observations and access to immense computational resources. There is no such universally accepted model to date and in order to carry out this research we need to make a number of assumptions. Namely, we focus on a hydrodynamic model wherein (a) the solar wind outflow has been assumed to be spherically symmetric which reduces the dimensionality of the problem from 3D to 1D, (b) the boundary and initial conditions are not changing with time, which leads to a steady state solution with a standing termination shock, (c) the solar wind acceleration and heating can be approximated by an effective polytropic index. These assumptions still capture the relevant fluid dynamics processes from the outflow of the solar wind to the creation of the shock as described by equations (2)-(4).

The termination shock is located at a distance of about ~ 100 AU [e.g. 2, 3, 4]. In order to reduce the numerical cost of both the CFD and PINN models and without loss of generality, we have reduced the numerical domain to $r \in [1, 20]\ R_\odot$ for both PLUTO and PINNs models. As explained in detail in Section 2, for a given pressure of the environment $P_{\mathrm{r_{max}}}$, a shock will be formed at the position where the ram pressure of the wind ρu^2 roughly equals $P_{\mathrm{r_{max}}}$. In other words, we have ensured that there is no loss of generality and all the relevant shock wave physics are captured properly. Including

the full scales, i.e. shock at a distance of ~ 100 AU, would only require that we increase the computational domain by a factor 10^3 and set the interstellar pressure $P_{\mathrm{r_{max}}}$ to a reduced value approximately scaling as $\propto r^{-2}$, i.e. reduced by a factor of 10^6. Such a setup would require thousands if not millions of times more computational resources than what we have used. Indicatively, we mention that the forward experiments took roughly ~ 1 hour each on a single V100 NVIDIA GPU on the reduced domain. Thus we would need months to years of V100 NVIDIA GPU time to solve the problem on a domain $r \in [1\ R_{\odot}, 100\ AU]$ with a similar accuracy.

For all the experiments presented in this work we have used synthetic PLUTO data produced with a polytropic index (γ) of 1.1. The inverse PINNs model was able to retrieve the correct polytropic index used for the synthetic data with high accuracy. In general, the polytropic index (γ) is used in the hydrodynamic PDE system to approximate the solar wind heating and acceleration mechanisms. As shown in a recent paper, this formulation cannot be deemed as a realistic approximation of the solar coronal heating and wind acceleration for $\gamma \geq 1.25$ [48], which further substantiates the results presented in the classical paper by [42], who showed that transonic winds exist provided that $\gamma < 3/2$. Motivated from the above physical reasons we chose a polytropic index gamma of 1.1, which is right in the middle of the range $1 < \gamma < 1.25$. In [49], the authors show in Fig. 1 therein that larger polytropic indexes (γ) lead to solar winds that become supersonic further away from the Sun. For example for a $\gamma = 1.2$ (and $\lambda = 3$) the standing shock would form at a distance of about 22 $R_{\odot}$ and we would need to extend our computational domain to properly capture that solution. It's worth noting here that we also tested the inverse model for the case of $\gamma = 1.2$ and we were able to retrieve the correct value $\hat{\gamma} = 1.19$, which corresponds to a 0.8% error. While the computational cost for larger values of polytropic indexes γ increases as we need to extend the computational domain, the inverse PINNs model is able to retrieve the right γ-value with the same accuracy.

4.2. Strengths of PINNs

In this work for the first time we applied PINNs for modeling astrophysical shocks in a gravitationally stratified environment. PINNs operate in fundamentally different ways than classic physics solvers as they learn by example through back-propagation and gradient descent minimization of multiple losses, while classic solvers have evolved through decades of research into robust solvers with well studied behaviors regarding convergence. The inverse problem is based on utilizing observational data to discover unknown physics and as such it's an area that PINNs - that have been constructed to learn from such data - thrive at, as we showed earlier in this work (see e.g. Section 3.1). In particular, we showed that PINNs are able to retrieve the correct $\gamma = 1.1$ with a relative error ~ 1 % even in cases when we use a fraction of the available data (e.g. 30,000 out of 150,000 as done in Section 3.1).

Moreover, PINNs are uniquely equipped to deal with data assimilation problems

as they have been designed to solve PDE systems alongside observational data, which they can incorporate and learn from in a seamless and native fashion. This scenario has special importance and relevance in heliophysics, as there is a plethora of satellites available taking in-situ measurements throughout the heliosphere. In contrast to PINNs, data assimilation cannot be handled by classic CFD codes, apart from the case of determining initial and boundary conditions. In Section 3.2 we showed that even with minimal amounts of data, PINNs can solve the PDE system for the physical quantities of interest with increasing accuracy as we gradually provide them with more observational data.

Another advantage of PINNs is the fact that they are very user-friendly [39] and as such they allow for setting up a new problem within a few lines of code which in turn allows for very quick experimentation cycles. We indicatively mention here the forward problem results that we presented in Section 3.3. We were able to tackle the poor accuracy in density and pressure due to their multi-scale variability by simply defining new (y_1, y_2) variables and reformulating the PDE system to solve for (u, P, ρ) in a straightforward manner. The PDE solvers in CFD codes while robust, they do not necessarily provide equally easy ways for customization.

4.3. Limitations of PINNs

While PINNs are arguably powerful in adding value when observational data are abundant, they still face several challenges and significant improvements need to be made in order for PINNs to get widely adopted for forward modeling realistic heliophysics problems. First-generation neural network architectures such as multi-layer perceptrons cannot match the robustness and accuracy of CFD codes, such as PLUTO, in integrating through long time-periods [39]. In particular, as explained in [39] currently PINNs are slower than finite element codes, they predominantly operate on a single GPU, their neural architectures are relatively simple, a lot of trial and error fine-tuning is required, and they perform poorly in multi-physics and multi-scale problems.

Furthermore, as argued in [50] fully connected PINNs architectures can lead to unstable training and inaccurate results, in particular in complex physical scenarios with multi-scale variability similar to the case examined in this paper [19, 51, 52]. [53] argued that the reason for these challenges is the fact that PINNs attempt to minimize multiple loss terms at once through gradient descent. The neural network optimizer might have to deal with losses that differ by several orders of magnitude which makes the minimization task and reaching a unique solution challenging. To address this issue [32] and [53] suggested adaptive weighting, which improves results, but does not address the fundamental problems in the PINNs optimization procedure [52].

Some recent studies have tried to address these challenges in a number of ways. Specifically, [33] and [38] tried to tackle the poor accuracy of PINNs by dividing the computational domain and solving the PDE system in segments. In Section 3.3, we

show-cased that integrating over segments does indeed improve the model convergence and accuracy. For a more detailed discussion on the issue of PINNs when long-time integration is necessary and possible solutions we refer to [54] and references therein. Finally, operator neural networks as well as more advanced neural architectures both show promising results [55, 56, 57, 58, 54, 59].

5. Conclusions

This is the first paper using PINNs for modeling shocks in while accounting for the gravitational force. In particular, we used PLUTO simulated data as benchmark to train PINNs in different training setups. We showed that Neural Networks and their powerful back-propagation and gradient descent learning technique can be combined with the physics PDEs to tackle problems from no or little data to data-intensive cases for astrophysical shocks. We successfully used PINNs to solve the inverse problem and retrieve the correct effective polytropic index that was used to produce the PLUTO data with an astounding accuracy of less than 1%. This demonstrates that PINNs can be used to discover underlying physics as captured in observations. In this case we recovered the correct effective γ-index 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 have showcased that PINNs can be a valuable complimentary modeling approach that can utilize all the available observations in a native fashion and inform the final prediction to improved accuracy. Finally, we showed that PINNs are still in the early phases of their maturity and cannot match the robustness and accuracy of classic physics PDE solvers in forward modeling, but they can still produce acceptable results if used for short time horizon integration. In particular, PINNs suffer from competing losses during gradient descent that can lead to poor performance especially when there is multi-scale variability in the fluid variables, as is the case for the density and pressure in this study due to the gravitational stratification of the solar atmosphere. In order to rectify for this we reformulated the conservative PDE system to absorb the inverse square law r^{-2} decrease of the fluid pressure and density, which vastly improved the convergence and retrieval of the ground truth solution by 2 to 16 times in some cases. PINNs and in general machine and deep learning have been started to get adopted for a number of use-cases in the heliophysics regime.

There is still a lot of work that remains to be done until the maturity of CFD models is achieved by PINNs, but recent results are promising about PINNs acting as an alternative modeling approach in astrophysical fluid dynamics. The 1D solar wind model we used for this study compared to higher dimensionality models has the added benefit that it accepts an analytic solution. This is an important step in the model development and benchmarking process before one moves towards a more realistic and complex multidimensional solar wind model, for which only numerical solutions

exist, which in turn come with their own computational errors. In a future study, we plan to generalize this model to 2D including the following most prominent physical considerations, i.e. (a) of the solar rotation and the (b) solar magnetic fields similar to what was done in [49]. Eventually, the goal would be to show that PINNs are able to stand alongside the realistic state-of-the-art 3D solar wind models that include more sophisticated heating and solar wind acceleration mechanisms through e.g. Alfvén wave heating, [see e.g. 60, and references therein].

Data Availability Statement

The PINNs model data can be reproduced through the publicly available notebooks in the GitLab repository `https://gitlab.qdatalabs.com/open_source/applied_research_papers/pinns_astrophysical_shocks`. The PLUTO benchmarking data are available in the same repository.

References

[1] Konstantinos Dialynas, Stamatios M. Krimigis, Robert B. Decker, and Donald G. Mitchell. Plasma pressures in the heliosheath from cassini ena and voyager 2 measurements: Validation by the voyager 2 heliopause crossing. *Geophysical Research Letters*, 46(14):7911–7919, 2019. doi: https://doi.org/10.1029/2019GL083924. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019GL083924`.

[2] R. B. Decker, S. M. Krimigis, E. C. Roelof, M. E. Hill, T. P. Armstrong, G. Gloeckler, D. C. Hamilton, and L. J. Lanzerotti. Voyager 1 in the Foreshock, Termination Shock, and Heliosheath. *Science*, 309(5743):2020–2024, September 2005. doi: 10.1126/science.1117569.

[3] E. C. Stone, A. C. Cummings, F. B. McDonald, B. C. Heikkila, N. Lal, and W. R. Webber. Voyager 1 Explores the Termination Shock Region and the Heliosheath Beyond. *Science*, 309(5743):2017–2020, September 2005. doi: 10.1126/science.1117684.

[4] R. B. Decker, S. M. Krimigis, E. C. Roelof, M. E. Hill, T. P. Armstrong, G. Gloeckler, D. C. Hamilton, and L. J. Lanzerotti. Mediation of the solar wind termination shock by non-thermal ions. , 454(7200):67–70, July 2008. doi: 10.1038/nature07030.

[5] Merav Opher, Abraham Loeb, James Drake, and Gabor Toth. A small and round heliosphere suggested by magnetohydrodynamic modelling of pick-up ions. *Nature Astronomy*, 4:675–683, March 2020. doi: 10.1038/s41550-020-1036-0.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL `https://www.sciencedirect.com/science/article/pii/0893608089900208`.

[7] M. G. Bobra and S. Couvidat. Solar Flare Prediction Using SDO/HMI Vector Magnetic Field Data with a Machine-learning Algorithm. , 798(2):135, January 2015. doi: 10.1088/0004-637X/798/2/135.

[8] Enrico Camporeale, Algo Carè, and Joseph E. Borovsky. Classification of Solar Wind With Machine Learning. *Journal of Geophysical Research (Space Physics)*, 122(11):10,910–10,920, November 2017. doi: 10.1002/2017JA024383.

[9] Kostas Florios, Ioannis Kontogiannis, Sung-Hong Park, Jordan A. Guerra, Federico Benvenuto, D. Shaun Bloomfield, and Manolis K. Georgoulis. Forecasting Solar Flares Using Magnetogram-based Predictors and Machine Learning. , 293(2):28, February 2018. doi: 10.1007/s11207-018-1250-4.

[10] Ahmed Lethy, Mohamed A. El-Eraki, Aalaa Samy, and Hanafy A. Deebes. Prediction of the Dst Index and Analysis of Its Dependence on Solar Wind Parameters Using Neural Network. *Space Weather*, 16(9):1277–1290, September 2018. doi: 10.1029/2018SW001863.

[11] A. Papaioannou, A. Anastasiadis, A. Kouloumvakos, M. Paassilta, R. Vainio, E. Valtonen, A. Belov, E. Eroshenko, M. Abunina, and A. Abunin. Nowcasting Solar Energetic Particle Events Using Principal Component Analysis. , 293(7):100, July 2018. doi: 10.1007/s11207-018-1320-7.

[12] Cristina Campi, Federico Benvenuto, Anna Maria Massone, D. Shaun Bloomfield, Manolis K. Georgoulis, and Michele Piana. Feature Ranking of Active Region Source Properties in Solar Flare Forecasting and the Uncompromised Stochasticity of Flare Occurrence. , 883(2):150, October 2019. doi: 10.3847/1538-4357/ab3c26.

[13] Manolis K. Georgoulis, D. Shaun Bloomfield, Michele Piana, Anna Maria Massone, Marco Soldati, Peter T. Gallagher, Etienne Pariat, Nicole Vilmer, Eric Buchlin, Frederic Baudin, Andre Csillaghy, Hanna Sathiapal, David R. Jackson, Pablo Alingery, Federico Benvenuto, Cristina Campi, Konstantinos Florios, Constantinos Gontikakis, Chloe Guennou, Jordan A. Guerra, Ioannis Kontogiannis, Vittorio Latorre, Sophie A. Murray, Sung-Hong Park, Samuelvon von Stachelski, Aleksandar Torbica, Dario Vischi, and Mark Worsfold. The flare likelihood and region eruption forecasting (FLARECAST) project: flare forecasting in the big data & machine learning era. *Journal of Space Weather and Space Climate*, 11:39, May 2021. doi: 10.1051/swsc/2021023.

[14] Spiridon Kasapis, Lulu Zhao, Yang Chen, Xiantong Wang, Monica Bobra, and Tamas Gombosi. Interpretable Machine Learning to Forecast SEP Events for Solar Cycle 23. *Space Weather*, 20(2):e2021SW002842, February 2022. doi: 10.1029/2021SW002842.

[15] Zeyu Sun, Monica G. Bobra, Xiantong Wang, Yu Wang, Hu Sun, Tamas Gombosi, Yang Chen, and Alfred Hero. Predicting Solar Flares Using CNN and LSTM on Two Solar Cycles of Active Region Data. , 931(2):163, June 2022. doi: 10.3847/1538-4357/ac64a6.

[16] E. Camporeale. The Challenge of Machine Learning in Space Weather: Nowcasting and Forecasting. *Space Weather*, 17(8):1166–1207, August 2019. doi: 10.1029/2018SW002061.

[17] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv e-prints*, art. arXiv:1711.10561, November 2017. doi: 10.48550/arXiv.1711.10561.

[18] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv e-prints*, art. arXiv:1711.10566, November 2017. doi: 10.48550/arXiv.1711.10566.

[19] M. Raissi, P. Perdikaris, and G. 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, February 2019. doi: 10.1016/j.jcp.2018.10.045.

[20] Zhiping Mao, Ameya D. Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, March 2020. doi: 10.1016/j.cma.2019.112789.

[21] *Investigation of Applying Physics Informed Neural Networks (PINN) and Variants on 2D Aerodynamics Problems*, volume Volume 3: Computational Fluid Dynamics; Micro and Nano Fluid Dynamics of *Fluids Engineering Division Summer Meeting*, 07 2020. doi: 10.1115/FEDSM2020-20184. URL `https://doi.org/10.1115/FEDSM2020-20184`. V003T05A055.

[22] Li Liu, Shengping Liu, Hui Xie, Fansheng Xiong, Tengchao Yu, Mengjuan Xiao, Lufeng Liu, and Heng Yong. Discontinuity Computing using Physics-Informed Neural Network. *arXiv e-prints*, art. arXiv:2206.03864, June 2022. doi: 10.36227/techrxiv.19391279.

[23] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018. URL `http://jmlr.org/papers/v18/17-468.html`.

[24] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, April 2017. doi: 10.1126/sciadv.1602614.

[25] Sheng Zhang and Guang Lin. Robust data-driven discovery of governing physical laws with error bars. *Proceedings of the Royal Society of London Series A*, 474 (2217):20180305, September 2018. doi: 10.1098/rspa.2018.0305.

[26] Jens Berg and Kaj Nyström. Data-driven discovery of PDEs in complex datasets. *Journal of Computational Physics*, 384:239–252, May 2019. doi: 10.1016/j.jcp.2019.01.036.

[27] E. Camporeale, X. Chu, O. V. Agapitov, and J. Bortnik. On the Generation of Probabilistic Forecasts From Deterministic Models. *Space Weather*, 17(3):455–475, March 2019. doi: 10.1029/2018SW002026.

[28] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, April 2020. doi: 10.1126/sciadv.aay2631.

[29] Hao Xu. DL-PDE: Deep-Learning Based Data-Driven Discovery of Partial Differential Equations from Discrete and Noisy Data. *Communications in Computational Physics*, 29(3):698–728, June 2021. doi: 10.4208/cicp.OA-2020-0142.

[30] E. Camporeale, George J. Wilkie, Alexander Y. Drozdov, and Jacob Bortnik. Data-Driven Discovery of Fokker-Planck Equation for the Earth's Radiation Belts Electrons Using Physics-Informed Neural Networks. *Journal of Geophysical Research (Space Physics)*, 127(7):e30377, July 2022. doi: 10.1029/2022JA030377.

[31] Nicolas Boullé, Christopher J. Earls, and Alex Townsend. Data-driven discovery of Green's functions with human-understandable deep learning. *Scientific Reports*, 12:4824, March 2022. doi: 10.1038/s41598-022-08745-5.

[32] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, February 2021. doi: 10.1016/j.jcp.2020.109951.

[33] Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, jun 2020. doi: 10.1016/j.cma.2020.113028. URL `https://doi.org/10.10162Fj.cma.2020.113028`.

[34] Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, April 2022. doi: 10.1016/j.cma.2022.114823.

[35] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational Physics-Informed Neural Networks For Solving Partial Differential Equations. *arXiv e-prints*, art. arXiv:1912.00873, November 2019. doi: 10.48550/arXiv.1912.00873.

[36] Ke Li, Kejun Tang, Tianfan Wu, and Qifeng Liao. D3M: A deep domain decomposition method for partial differential equations. *arXiv e-prints*, art. arXiv:1909.12236, September 2019. doi: 10.48550/arXiv.1909.12236.

[37] Ehsan Kharazmi, Zhongqiang Zhang, and George E. M. Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, February 2021. doi: 10.1016/j.cma.2020.113547.

[38] 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, feb 2022. doi: 10.1016/j.cma.2021.114474. URL https://doi.org/10.10162Fj.cma.2021.114474.

[39] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[40] Jon Vandegriff, Kiri Wagstaff, George Ho, and Janice Plauger. Forecasting space weather: Predicting interplanetary shocks using neural networks. *Advances in Space Research*, 36(12):2323–2327, January 2005. doi: 10.1016/j.asr.2004.09.022.

[41] D. Sudar, B. Vršnak, and M. Dumbović. Predicting coronal mass ejections transit times to Earth with neural network. , 456(2):1542–1548, February 2016. doi: 10.1093/mnras/stv2782.

[42] E. N. Parker. The Hydrodynamic Theory of Solar Corpuscular Radiation and Stellar Winds. , 132:821, November 1960. doi: 10.1086/146985.

[43] E. N. Parker. Theory of Solar Wind. In *International Cosmic Ray Conference*, volume 1 of *International Cosmic Ray Conference*, page 175, January 1963.

[44] E. N. Parker. Dynamical Properties of Stellar Coronas and Stellar Winds. I. Integration of the Momentum Equation. , 139:72, January 1964. doi: 10.1086/147740.

[45] E. N. Parker. Dynamical Theory of the Solar Wind. , 4(5-6):666–708, September 1965. doi: 10.1007/BF00216273.

[46] A. Mignone. High-order conservative reconstruction schemes for finite volume methods in cylindrical and spherical coordinates. *Journal of Computational Physics*, 270:784–814, Aug 2014. doi: 10.1016/j.jcp.2014.04.001.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[48] Chen Shi, Marco Velli, Stuart D. Bale, Victor Réville, Milan Maksimović, and Jean-Baptiste Dakeyo. Acceleration of polytropic solar wind: Parker Solar Probe observation and one-dimensional model. *Physics of Plasmas*, 29(12):122901, 12 2022. ISSN 1070-664X. doi: 10.1063/5.0124703. URL https://doi.org/10.1063/5.0124703.

[49] R. Keppens and J. P. Goedbloed. Numerical simulations of stellar winds: polytropic models. , 343:251–260, March 1999. doi: 10.48550/arXiv.astro-ph/9901380.

[50] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, January 2022. doi: 10.1016/j.jcp.2021.110768.

[51] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate

modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, October 2019. doi: 10.1016/j.jcp.2019.05.024.

[52] Olga Fuks and Hamdi A Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1), 2020.

[53] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, January 2021. doi: 10.1137/20M1318043.

[54] Katarzyna Michałowska, Somdatta Goswami, George Em Karniadakis, and Signe Riemer-Sørensen. Neural Operator Learning for Long-Time Integration in Dynamical Systems with Recurrent Neural Networks. *arXiv e-prints*, art. arXiv:2303.02243, March 2023. doi: 10.48550/arXiv.2303.02243.

[55] Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv e-prints*, art. arXiv:1910.03193, October 2019. doi: 10.48550/arXiv.1910.03193.

[56] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-Informed Neural Operator for Learning Partial Differential Equations. *arXiv e-prints*, art. arXiv:2111.03794, November 2021. doi: 10.48550/arXiv.2111.03794.

[57] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier Neural Operator with Learned Deformations for PDEs on General Geometries. *arXiv e-prints*, art. arXiv:2207.05209, July 2022. doi: 10.48550/arXiv.2207.05209.

[58] Arda Mavi, Ali Can Bekar, Ehsan Haghighat, and Erdogan Madenci. An unsupervised latent/output physics-informed convolutional-LSTM network for solving partial differential equations using peridynamic differential operator. *Computer Methods in Applied Mechanics and Engineering*, 407:115944, March 2023. doi: 10.1016/j.cma.2023.115944.

[59] Wenhui Peng, Zelong Yuan, Zhijie Li, and Jianchun Wang. Linear attention coupled Fourier neural operator for simulation of three-dimensional turbulence. *Physics of Fluids*, 35(1):015106, January 2023. doi: 10.1063/5.0130334.

[60] Nishtha Sachdeva, Ward B Manchester, Igor Sokolov, Zhenguang Huang, Alexander Pevtsov, Luca Bertello, Alexei A. Pevtsov, Gabor Toth, and Bart van der Holst. Solar wind modeling with the Alfven Wave Solar atmosphere Model driven by HMI-based Near-Real-Time maps by the National Solar Observatory. *arXiv e-prints*, art. arXiv:2212.05138, December 2022. doi: 10.48550/arXiv.2212.05138.

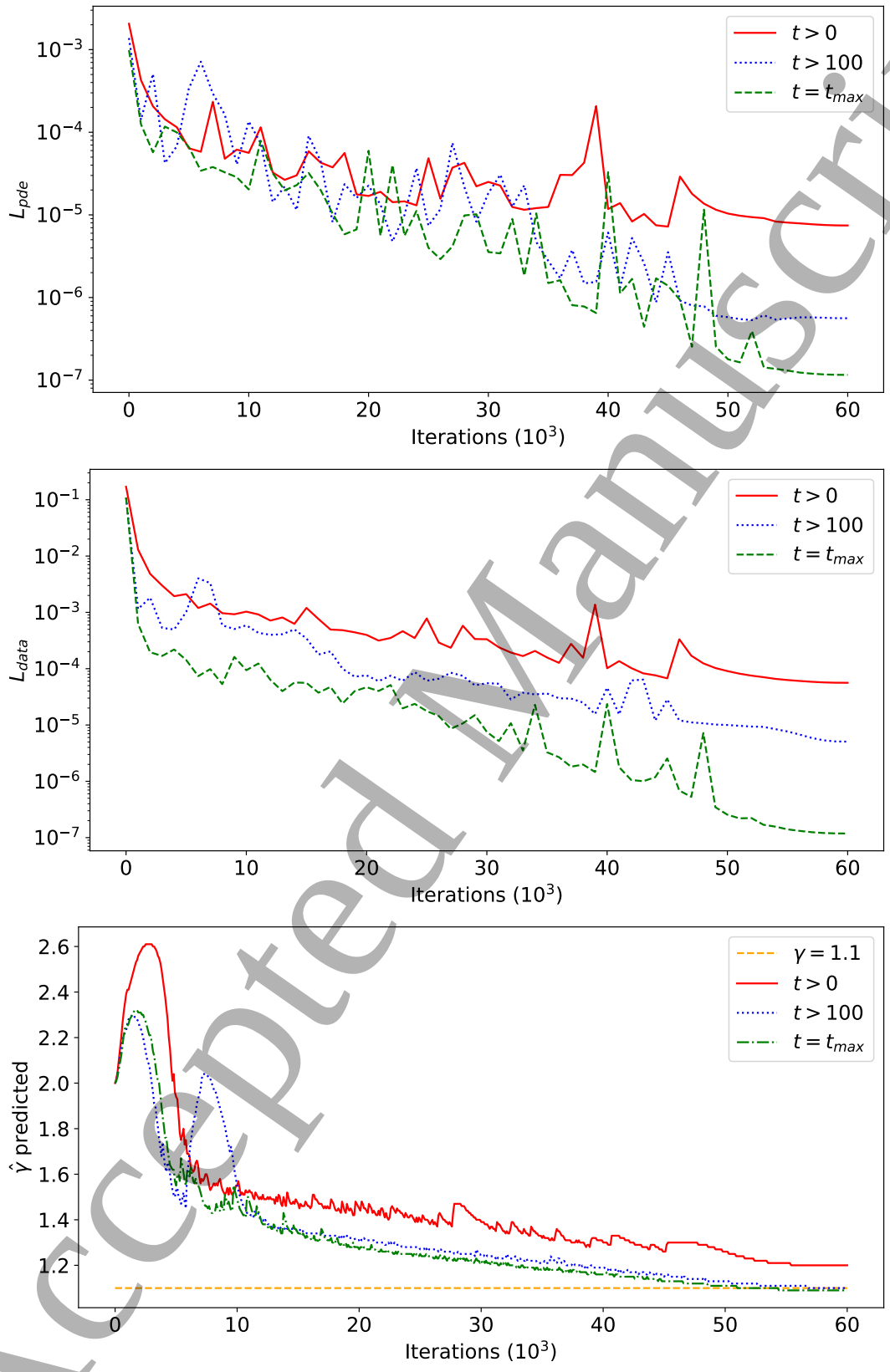

Figure 2. Inverse problem results in the top panel we show the PDE losses, in middle panel we show the data losses and in bottom panel we show the convergence of the $\hat{\gamma}$ value to $\gamma = 1.1$.

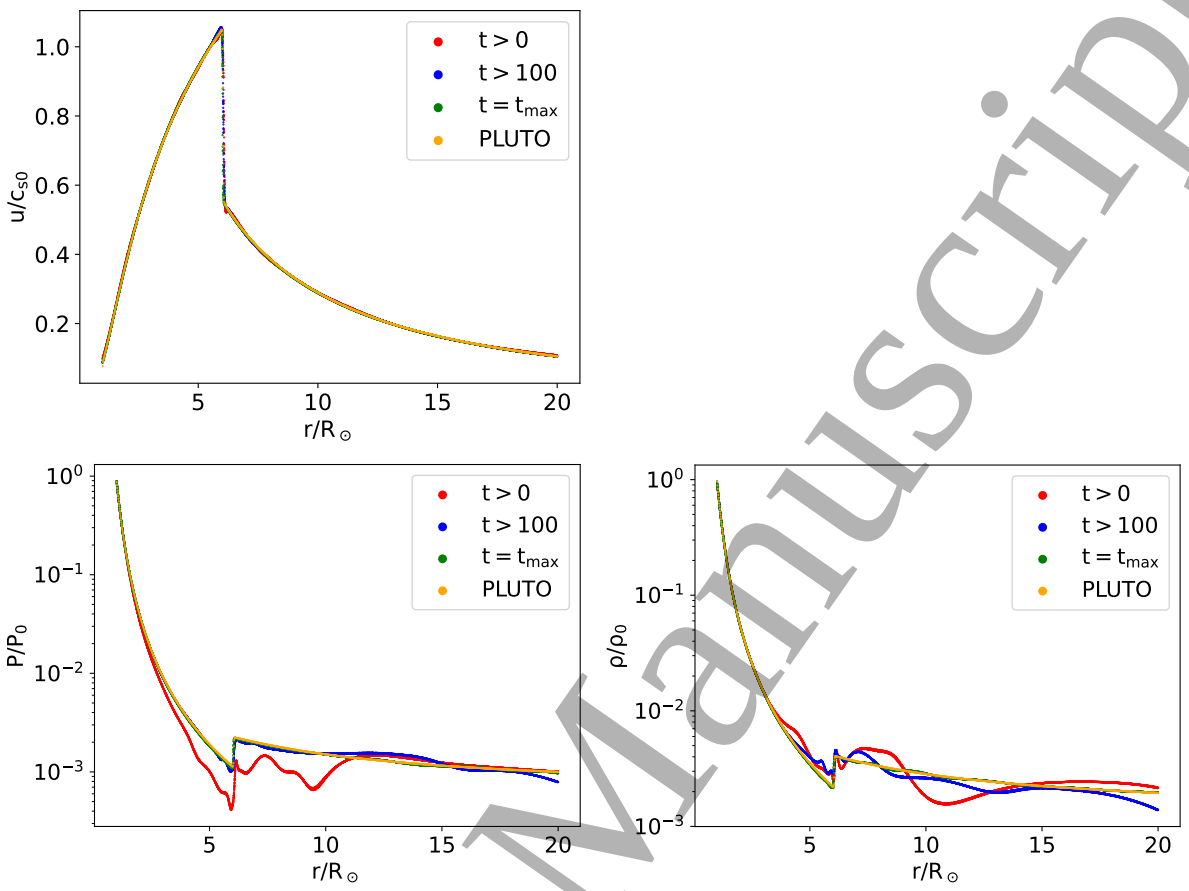

Figure 3. Results for inverse problem: the top left panel of this figure shows the predicted velocity $\hat{u}$ for the experiments No.1, No.5 and No.8 accordingly as captured in the Table 1 together with the ground truth velocity from the PLUTO simulation, the bottom left panel shows the predicted pressure $\hat{P}$ for the same experiments, and bottom right panel shows the density profile $\hat{\rho}$ for same experiments. The velocity is predicted with high accuracy by all the experiments, while pressure and density are better predicted for experiments with data points sampled closer to the end of the PLUTO simulation due to lower variance and faster convergence in those cases.

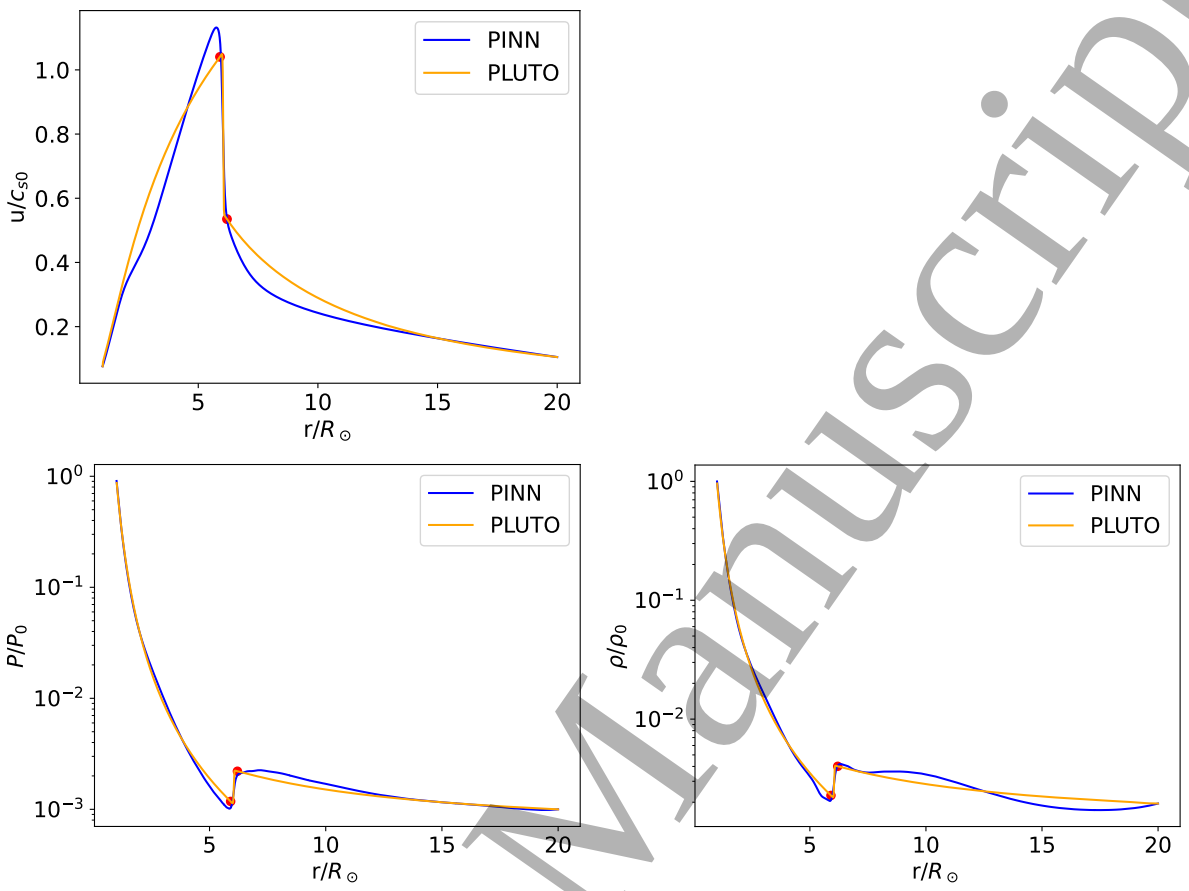

Figure 4. The top left panel shows the velocity profile, the middle left the predicted pressure profile and the middle right panel shows the density profile for the 2 point PINN experiment (blue curve) together with the PLUTO ground truth (orange curve) with the red points indicating the sampled data assimilated.

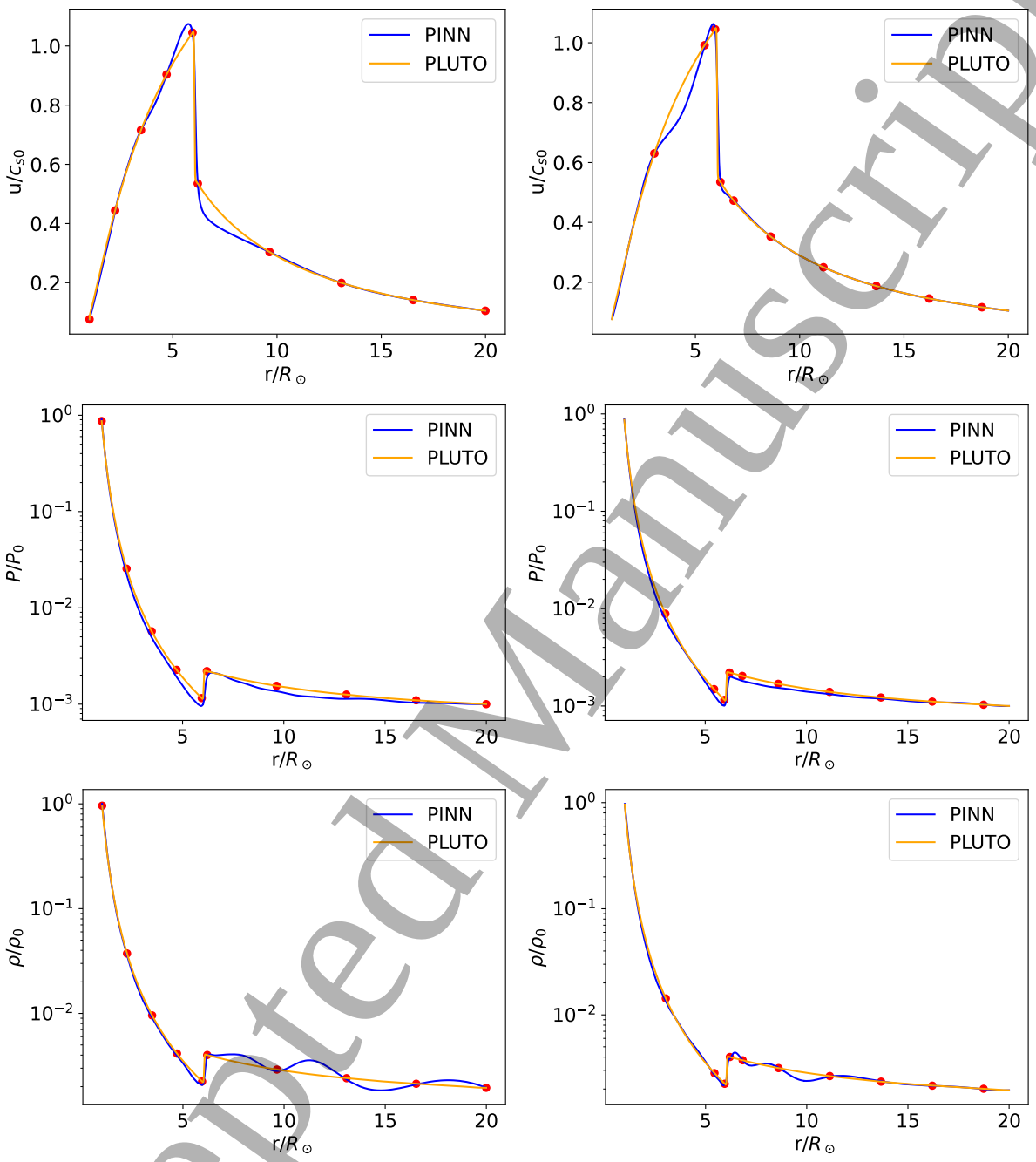

Figure 5. Left column panels correspond to Experiment B and right column panels correspond to Experiment C from Table 2. The top panels show the predicted velocity profiles, the middle panels show the pressure profiles and the bottom panels show the density profiles for the 10 point PINN experiments (blue curve) together with the PLUTO ground truth (orange curve) with the red points indicating the sampled data assimilated.

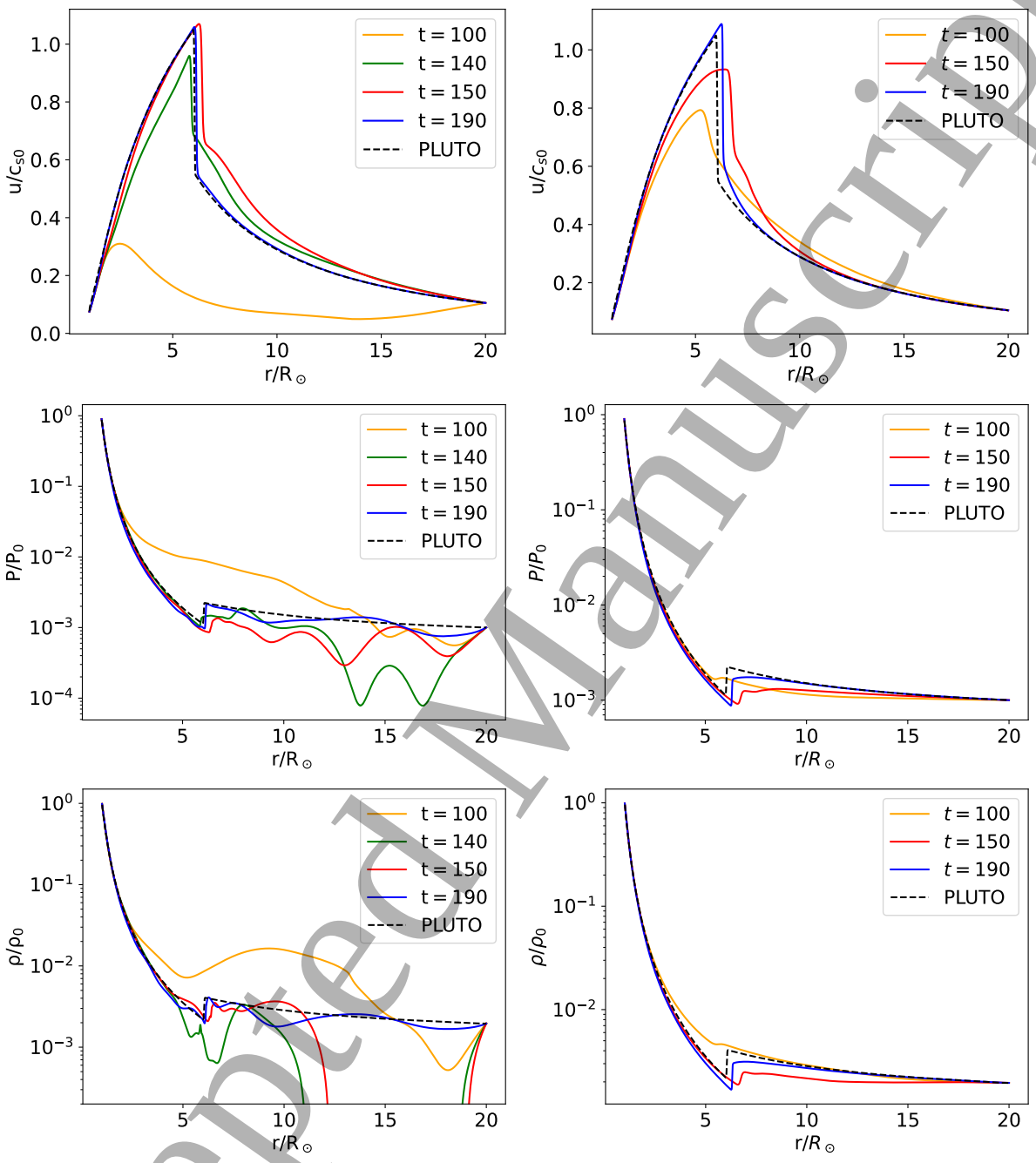

Figure 6. Results for the forward problem: top left panel shows the predicted velocity $\hat{u}$ for the experiments and PLUTO simulation, bottom left panel shows the predicted pressure $\hat{P}$ for same experiments, and the bottom right panel shows the predicted density profile $\hat{\rho}$ for the same experiments. The legends indicate the initial time of the numerical domain with respect to the PLUTO $t_{init} = 0$ and $t_{final} = t_{ss} = 200$ code units.