Solving Hydrodynamic Shock-Tube Problems Using Weighted Physics-Informed Neural Networks with Domain Extension

Alexandros Papados
University of Maryland, College Park
Applied Mathematics, Applied Statistics, Scientific Computing Program
(Dated: March 21, 2021)

This paper focuses in solving complex hydrodynamic shock-tube problems using *Weighted Physics-Informed Neural Networks with Domain Extension* (W-PINNs-DE). To date, it has been shown that PINNs is an efficient numerical tool which provides solutions to partial differential equations (PDEs), despite the fact that, theoretically, it exhibits limited capability in solving problems with only continuous solutions. Specifically, we will be solving the compressible Euler equations that model gas and fluid dynamics. In an effort to demonstrate the extent of W-PINNs-DE, we will solve six hydrodynamic shock-tube test problems. Each of these problems yields solutions that develop discontinuities, such as shocks and rarefaction fans. The suite of problems chosen in this paper is widely used as a validation tool to ensure the ability of newly developed numerical schemes to capture, within tolerable levels, the solutions of the PDEs at hand. Herewith, we validate W-PINNs-DE in the same manner. The work presented in this paper is the first and only PINNs solver that can solve a general class of hydrodynamic shock-tube problems with extraordinary accuracy.

I. INTRODUCTION

Since the late 1990s, machine learning for scientific computing, also known as scientific machine learning, became a popular research area for scientists worldwide. Most notably, (Lagaris et al, 1998) set the stage by solving differential equations using neural networks. Due to the increase in computational power over the years, a flux of papers has been published discussing various machine learning techniques to solve various computational problems. Hence, researchers today incorporate machine learning in several scientific areas, such as extracting physical parameters from experimental data for numerical simulations, detecting cracks in the specific regions of a material, and, to the interest of this paper, solving a highly nonlinear system of partial differential equations (PDEs).

Of the family of numerical solvers for PDEs that incorporate machine learning, Physics-Informed Neural Networks (PINNs) have proven to be the most popular of these methods. PINNs provide an easy-to-use framework to successfully solve forward and inverse problems, with an impressive degree of accuracy. Its simplistic implementation has made it a competitive numerical tool compared to standard methods, such as the finite volume and nonlinear least-squares method. Professors George Karniadakis of Brown University, Maziar Raissi of Colorado University, Boulder, and Paris Perdikaris of the University of Pennsylvania introduced PINNs in the paper, *Physics-Informed Neural Networks: A deep learning framework for solving forward and inverse problems involving partial differential equations* (Karniadakis et al, 2019), published in the *Journal of Computational Physics*. They constructed PINNs to solve "supervised learning tasks while respecting any given law of physics described by a general nonlinear partial differential equation" (Karniadakis et al, 2019). Since the first publication of the method, PINNs became a popular research area in machine learning, and their application to various scientific fields is increasing.

Although PINNs has become a popular method for solving PDEs, just as with any classical numerical method has its limitations, so do PINNs. These limitations are especially prominent when solving conservation laws whose solutions are discontinuous. However, we expect this difficulty since the universal approximation theorem for neural networks (Pinkus, 1999) specifies that neural networks are universal approximators for solutions to PDEs that are *continuous*. There is no theoretical backing that generalizes a neural network's ability to approximate any discontinuous solution. Thus, this raises a significant complexity when designing a neural network that will solve a conservation law. In fluid dynamics, the compressible Euler equations describe gas and fluid flow by relating the density, velocity, and pressure to one another. These equations have immense application when modeling shock-tubes problems and explosions. The finite volume method (FVM) is the most popular method to solve these problems, but it needs adjustments to its original numerical formulation in order to capture the necessary physics. For example, introducing *flux-limiting* is necessary to resolve artificial dispersion and dissipation created by the FVM schemes. Similarly, PINNs without modifications develop similar phenomena when solving for solutions that include shocks and rarefaction fans. Therefore, for PINNs to be a competitive numerical tool to that of the FVM with flux-limiting, we must modify the method to solve complicated problems in fluid dynamics, particularly hydrodynamic shock-tube problems. To briefly show the effectiveness of W-PINNs-DE, in figure 1, we offer a side-by-side comparison of the PINNs and W-PINNs-DE solution for the density term of the classic hydrodynamic shock-tube test problem, *Sod Shock-Tube Problem* (Sod, 1978).

At the time of this paper and to the knowledge of the author, (Michoski et al, 2019) and (Patel et al, 2020) are the only other publications introducing modifications to PINNs to solve the Sod shock-tube problem. Both papers discuss modifications to rectify for spurious oscillations near shocks generated by the original PINNs formulation during training. (Michoski et al, 2019) presents artificial viscosity by adding viscous terms to the Euler equations. Introducing artificial viscosity is in itself a non-physical adjustment to the mathematical formulation.

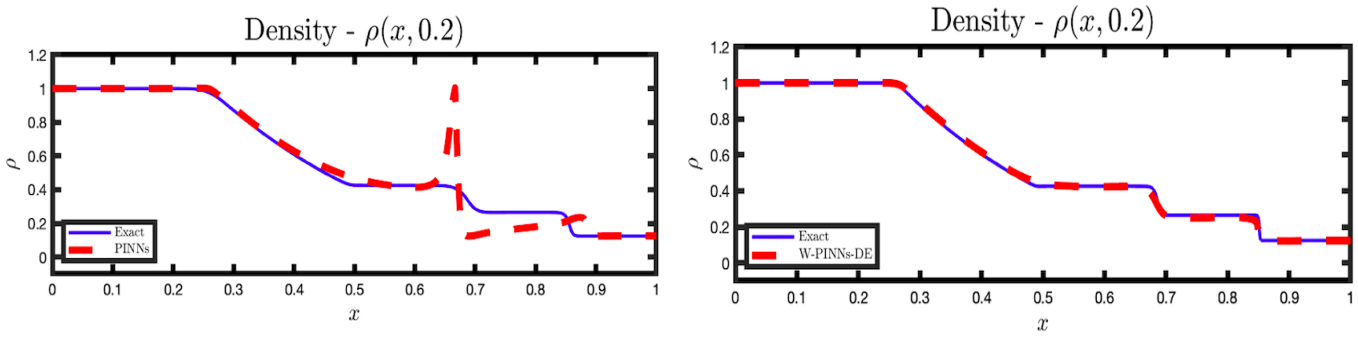

Figure 1 – The left figure is the solution of the density of the Sod Shock-tube problem using the standard formulation of PINNs. The right figure is the solution of the density of the Sod Shock-tube problem using W-PINNs-DE

Hence, adding artificial viscosity stabilizes the solution but at the same time violates, to a certain degree, physical laws in its attempt to capture shocks and rarefaction fans. Moreover, adding artificial viscosity is computationally expensive for neural networks since this requires computing second partial derivatives for each physical quantity, adding computational complexity during backpropagation and automatic differentiation. (Patel et al, 2020) merge an FVM total variation diminishing method with PINNs. The merger further complicates the PINNs algorithm and, thus, takes away from the simplicity of the PINNs method. The adjustments made in both papers do not solve a general class of shock-tube problems but rather solve a specific shock-tube problem in particular. This paper will introduce a simple modification to PINNs that allows for solving a broad category of hydrodynamic shock-tube problems with a high degree of accuracy. W-PINNs-DE proves to solve the Sod shock-tube problem more accurately than the methods proposed in both the two papers mentioned above.

For each test problem, we will solve the Riemann problem. In other words, the initial state of a physical quantity is greater on the left or right with respect to the point of discontinuity, x^*, and remains constant on either side. We incorporate specific weights to the total loss function such that the loss of the initial condition decreases faster than the loss of the PDE. We then extend the original domain to introduce *ghost points* to the neural network for training purposes. Domain extension smooths out the spurious oscillations by providing the neural network with more points to the left and right of x^*, and take on the actual values of the boundary condition, i.e, the boundary conditions for each problem take on value of the boundary of the initial condition. The domain extends depending on the initial state $\boldsymbol{U_0}$. The domain extension technique is formally defined and discussed in section II.

The outline of this paper is as follows. In section II, we present W-PINNs-DE by constructing the method and its algorithm. Section III discusses the validation of W-PINNs-DE. Lastly, in sections IV-X, we solve each of the six hydrodynamic shock-tube test problems and present figures of the solution as well as the relative L_2 error for each physical quantity, ρ, u, and p, respectively.

II. W-PINNS-DE

Let $\Omega \subset \mathbb{R}$ and consider the 1-D compressible Euler equations in characteristic form:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}\frac{\partial \boldsymbol{U}}{\partial x} = 0, \quad (x,t) \in \Omega \times (0,T] \tag{1}$$

where,

$$\boldsymbol{U} = (\rho, u, p)^T, \ \boldsymbol{A} = \begin{pmatrix} u & \rho & 0 \\ 0 & u & \frac{1}{\rho} \\ 0 & \rho a^2 & u \end{pmatrix}$$

where $a = \sqrt{\gamma p/\rho}$ is the speed of sound, ρ is the density, u is the velocity, p is the pressure, and γ is the heat capacity ratio. For a standard hydrodynamic shock-tube problem, typically, the initial condition is of the form:

$$\boldsymbol{U}(x,0) = \boldsymbol{U_0} = \begin{cases} \boldsymbol{u_L}, & x < x^* \\ \boldsymbol{u_R}, & x > x^* \end{cases}, \quad \boldsymbol{u_L} = [\rho_L, u_L, p_L], \quad \boldsymbol{u_R} = [\rho_R, u_R, p_R] \tag{2}$$

with Dirichlet boundary conditions that take the values of the initial condition at the boundaries.

First, we will discuss what it means to extend the domain of the problem. After formally defining how to extend the domain, we will discuss what weights are needed for the loss function. We extend the domain in a manner such that if

$\boldsymbol{u_L} > \boldsymbol{u_R}$, the domain is extended such that the initial state leans to the left of the newly extended domain. If $\boldsymbol{u_R} > \boldsymbol{u_L}$, the domain is extended such that the initial state leans to the right. Extending the domain extends the physical state by providing the initial condition and physics to the neural network advantage for training. Since the initial state, $\boldsymbol{U_0}$, varies but remains constant to the right and left of the point of discontinuity, x^*, we may extend the domain so the neural network trains on more points that remain constant, and hence are easily learnt by the network. By the time the neural network reaches x^* from the right and left, it will realize that there is a sharp change in gradient (a discontinuous change). Extending the domain smooths out any artificial oscillations that are typically generated when solving within the original domain. Domain extension in this manner acts as artificial viscosity, but does not introduce any non-physical terms. Therefore, no physical laws are altered or violated. To truly add artificial viscosity, we would need to incorporate ρ_{xx}, u_{xx}, and p_{xx} into the Euler equations. Adding artificial viscosity is more costly since much of the computation lies in calculating the second-derivatives. Therefore, domain-extension and having the initial discontinuity lean in its respective direction is a cheaper and more effective way of diffusing spurious oscillations when solving hydrodynamic shock-tube problems using PINNs.

We will now formally define domain extension. Let $\Omega \equiv [a, b]$ be the original spatial domain, and $x^* \in \Omega$ is the point of discontinuity from (2), Then,

if $\boldsymbol{u_L} > \boldsymbol{u_R}$

$$a_{new} \leq x^* - 2.0, \quad b_{new} \leq x^* + 2.625$$

if $\boldsymbol{u_R} > \boldsymbol{u_L}$

$$a_{new} \geq x^* - 3.125, \quad b_{new} \geq x^* + 2.0$$

Therefore, we have the domain spatial transformation, $[a, b] \to [a_{new}, b_{new}]$. For the first five test problems presented in this paper, the domain extension will be based on the formulation above, for the last example, we will have a mixture of initial states where $\rho_R > \rho_L$, and $[u_L, p_L] > [u_R, p_R]$. In the case where we have a mixed initial state, we will have a mixture of the extended domain such that:

$$a_{new} \geq x^* - 3.125, \quad b_{new} \leq x^* + 2.625$$

In figure 2, we present a right leaning domain extension.

The second modification we make is with regards to the weights of the total loss function. In the original PINNs formulation, one minimizes the total loss, which is a sum of the loss of the PDE, IC, and BCs, in order to optimize the weights of the neural network, $\theta \in \mathbb{R}^k$. To solve the Euler equations using PINNs, we construct a deep neural network (DNN), $\tilde{U}(x, t, \theta)$, where (x, t) are inputs to the network, and $\tilde{U} = [\tilde{\rho}, \tilde{u}, \tilde{p}]$ are the outputs. The standard loss is defined by:

$$G(\theta) = \frac{1}{N_f}\left|\left|\frac{\partial \tilde{\boldsymbol{U}}}{\partial t}(x,t,\theta) + \tilde{\boldsymbol{A}}\frac{\partial \tilde{\boldsymbol{U}}}{\partial x}(x,t,\theta)\right|\right|^2_{\Omega\times(0,T],\nu_1} + \frac{1}{N_{IC}}\left|\left|\tilde{\boldsymbol{U}}(x,0,\theta) - \boldsymbol{U}(x,0)\right|\right|^2_{(\Omega,\nu_2} + \frac{1}{N_{BC}}\left|\left|\tilde{\boldsymbol{U}}(x,t,\theta) - \boldsymbol{U}(x,t)\right|\right|^2_{\partial\Omega\times(0,T],\nu_3}$$

where N_f, N_{IC}, and N_{BC} correspond to the number of points sampled from the interior, initial, and boundary conditions, according to their respective probability densities ν_1, ν_2 and ν_3. Since the initial condition captures the boundary conditions, we are able to dispose the loss of the boundary conditions, leaving us with a condense form for the total loss, as follows:

$$G(\theta) = \frac{1}{N_f}\left|\left|\frac{\partial \tilde{\boldsymbol{U}}}{\partial t}(x,t,\theta) + \tilde{\boldsymbol{A}}\frac{\partial \tilde{\boldsymbol{U}}}{\partial x}(x,t,\theta)\right|\right|^2_{\Omega\times(0,T],\nu_1} + \frac{1}{N_{IC}}\left|\left|\tilde{\boldsymbol{U}}(x,0,\theta) - \boldsymbol{U}(x,0)\right|\right|^2_{\Omega,\nu_2}$$

The issue of using the loss function presented above is that the loss of the PDE decreases at a faster rate than the loss of the IC. This is a major issue since the user provides the initial condition data to the neural network, thus, the neural network should focus its training on data that is provided first, then attempt to establish the solution of the PDE. Moreover, if the initial condition is learnt incorrectly, we cannot expect to have the solution at the final time to be accurate. This is trivial since a time-dependent PDE is highly dependent on its initial state. The solution at later times, $t > 0$, will evolve based on the initial state, at $t = 0$. Therefore, minimizing the loss of the initial condition at a faster rate than that of the PDE is required to yield fast converging, accurate solutions.

Due to the initial discontinuous state, the neural network has much difficulty learning the initial condition since at x^* the neural network cannot differentiate in the strong sense. This is to be expected since the universal approximation theorem does not specify whether neural network can approximate discontinuous functions. Therefore, to rectify this, we introduce a disproportionate penalty to the loss of the initial condition, compared to that of the PDE, such that the neural network focuses its training on the initial condition. Once the initial condition is trained well, the solution of the PDE will be learnt with ease.

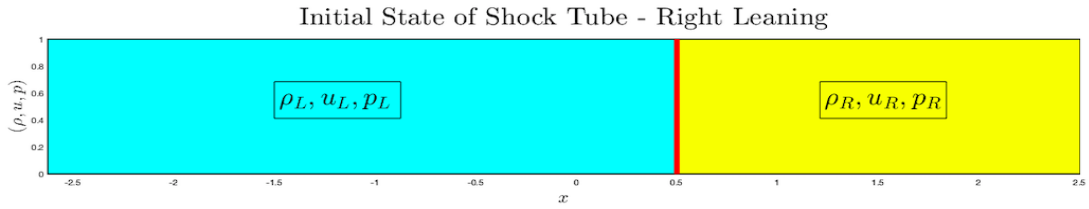

Figure 2 – $\boldsymbol{u_L < u_R} \implies$ Right Leaning Extension, ex) $[0,1] \to [-2.625, 2.5]$

Therefore, we use the following weighted loss function:

$$G(\theta) = \frac{\omega_f}{N_f}\left|\left|\frac{\partial \tilde{\boldsymbol{U}}}{\partial t}(x,t,\theta) + \tilde{\boldsymbol{A}}\frac{\partial \tilde{\boldsymbol{U}}}{\partial x}(x,t,\theta)\right|\right|^2_{\Omega\times(0,T],\nu_1} + \frac{\omega_{IC}}{N_{IC}}\left|\left|\tilde{\boldsymbol{U}}(x,0,\theta) - \boldsymbol{U}(x,0)\right|\right|^2_{\Omega,\nu_2}$$

where $\omega_f = 0.1$, $\omega_{IC} = 10$. With the choice of weights, ω_f, ω_{IC} and the proper domain extension, if the initial state does not contain strong shocks, (i.e $||\boldsymbol{U_0}||^2 > 10$, or blast waves), we guarantee the following inequality for a sufficient number of epochs (training iterations):

$$0 \leq \frac{1}{N_{IC}}\left|\left|\tilde{\boldsymbol{U}}(x,0,\theta) - \boldsymbol{U}(x,0)\right|\right|^2_{\Omega,\nu_2} \leq \frac{1}{N_f}\left|\left|\frac{\partial \tilde{\boldsymbol{U}}}{\partial t}(x,t,\theta) + \tilde{\boldsymbol{A}}\frac{\partial \tilde{\boldsymbol{U}}}{\partial x}(x,t,\theta)\right|\right|^2_{\Omega\times(0,T],\nu_1}, \qquad \forall\theta \in \mathbb{R}^k$$

W-PINNs-DE allows for this inequality by the end of its training, as well as an accurate solution of a hydrodynamic shock-tube problem.

To summarize, W-PINNs-DE extends the original domain so the neural network trains on ghost points that take on the values of the boundary condition and the boundaries of the initial condition. These ghost points are used to smooth oscillations near points of discontinuity and, thus, minimize the total loss more efficiently. Extending the domain acts as artificial viscosity for neural networks, but without introducing non-physical terms to the mathematical model. W-PINNs-DE then penalize the total loss function by introducing weights that allow the loss of initial condition to decrease at a faster rate than the loss of the PDE.

Algorithm 1: W-PINNs-DE Algorithm

1 Extend Ω based on the initial state (left or right leaning)

2 Generate weights $\theta \in \mathbb{R}^k$ and a deep neural network (DNN), $\tilde{U}(x,t,\theta)$, where (x,t) are inputs to the network, and $\tilde{U} = [\tilde{\rho}, \tilde{u}, \tilde{p}]$ are the outputs. The number of layers, neurons per layer, and activation functions for each layer are prescribed by the user.

3 Generate random points (x_n, t_n) from $\Omega \times \mathbb{R}^+$ and w_n from Ω, according to their respective probability densities, ν_1 and ν_2. Let N_f, N_{IC} correspond to the number of points sampled from the interior and initial condition, respectively.

4 Optimize θ by minimizing the loss function, $G(\theta)$, where:

$$G(\theta) = \frac{\omega_f}{N_f}\left|\left|\frac{\partial \tilde{\boldsymbol{U}}}{\partial t}(x,t,\theta) + \tilde{\boldsymbol{A}}\frac{\partial \tilde{\boldsymbol{U}}}{\partial x}(x,t,\theta)\right|\right|^2_{\Omega\times(0,T],\nu_1} + \frac{\omega_{IC}}{N_{IC}}\left|\left|\tilde{\boldsymbol{U}}(x,0,\theta) - \boldsymbol{U}(x,0)\right|\right|^2_{\Omega,\nu_2}$$

where $\omega_f = 0.1$, $\omega_{IC} = 10$

5 Update θ by performing stochastic gradient descent:

$$\theta = \theta - \eta\nabla_\theta G(\theta)$$

where η is the learning rate.

III. VALIDATION

To ensure that a proper W-PINNs-DE solution is obtained, we measure the relative L_2 error between the W-PINNs-DE solution and an analytic solution or a refined numerical solution. Specifically, we measure:

$$\mathcal{E} = \frac{||\tilde{\boldsymbol{U}}(x,t,\theta) - \boldsymbol{U}(x,t)||_2}{||\boldsymbol{U}(x,t)||_2}$$

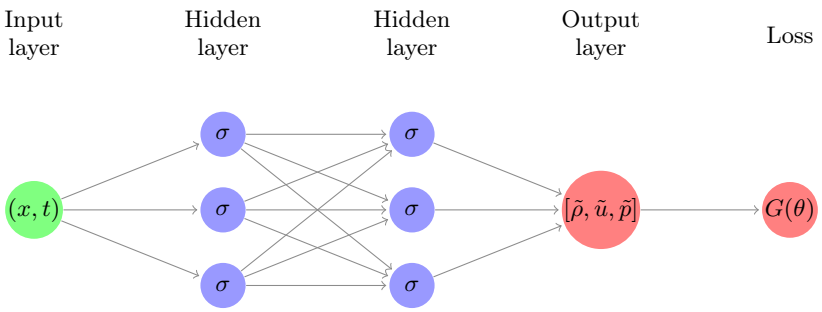

Figure 3 – Schematic of the neural network for the Euler equations

Due to the novelty of PINNs, much of its mathematical theory is in development. Professor Allan Pinkus of the Israel Institute of Technology, proved that a continuous function and its derivatives may be approximated by any single layered (hence a multilayered) neural network. This theorem suggests and proves that it is possible to approximate a solution to a PDE by using neural networks. Hence, PINNs becomes an eligible method for providing solutions to PDEs as well as their *inverse problems*.

Theorem 1. ***(Pinkus, 1999):***

Set $\boldsymbol{m} = m_1 + ... + m_d$ *and*

$$D^{\boldsymbol{m}} = \frac{\partial^{|\boldsymbol{m}|}}{\partial x_1^{m_1} \cdots \partial x_d^{m_d}}$$

Let $\boldsymbol{m}^i \in \mathbb{Z}_+^d$, $i = 1, .., s$, *and set* $m = \max_{i=1,..,s} |\boldsymbol{m}^i$. *Assume* $\sigma \in C^m(\mathbb{R})$ *and is not a polynomial. Then the space of single hidden layer neural nets:*

$$\mathcal{M}(\sigma) = span\{\sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b) : \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

is dense in $C^{\boldsymbol{m}^1,..,\boldsymbol{m}^s}(\mathbb{R}^d)$. *In other words, for any* $f \in C^{\boldsymbol{m}^1,..,\boldsymbol{m}^s}(\mathbb{R}^d)$, *any compact* $K \subset \mathbb{R}^d$, *and any* $\epsilon > 0$, *there exists a* $g \in \mathcal{M}(\sigma)$ *satisfying*

$$\max_{\boldsymbol{x} \in K} \left| D^{\boldsymbol{k}} f(\boldsymbol{x}) - D^{\boldsymbol{k}} g(\boldsymbol{x}) \right| < \epsilon$$

for all $\boldsymbol{k} \in \mathbb{Z}_+^d$ *for which* $\boldsymbol{k} \leq \boldsymbol{m}^i$.

Unfortunately, Theorem 1 does not tell us anything about solutions to PDEs that are discontinuous. In this paper, we demonstrate that for the Euler equations, the W-PINNs-DE method has the ability to approximate a general class of discontinuous solutions, referred to as the hydrodynamic shock-tube problems. We validate the accuracy of W-PINNs-DE by solving the following six hydrodynamic shock-tube test problems:

Problem	ρ_L	u_L	p_L	ρ_R	u_R	p_R	x^*
Single Contact Discontinuity	1.4	0.1	1.0	1.0	0.1	1.0	0.5
Double Expansion Fan	1.0	-1.0	0.4	1.0	1.0	0.4	0.5
Sod Shock-Tube Problem	1.0	0.0	1.0	0.125	0.0	0.1	0.5
Reverse Shock-Tube Problem	0.125	0.0	0.1	1.0	0.0	1.0	0.5
High Speed Shock-Tube I	0.125	0.0	0.1	1.0	0.75	1.0	0.3
High Speed Shock-Tube II	0.445	0.698	0.70	0.5	0.0	0.571	0.5

Table I – Six test problems to validate W-PINNs-DE ability to solve hydrodynamic shock-tube problems

Table I establishes each test problem presented in this paper by defining the initial left and right state for the density, velocity, and pressure, respectively. Sections V-VIII are dedicated to each individual problem.

IV. W-PINNS-DE ARCHITECTURE

For each problem presented in table I, we sample points from the extended computational domain that has been partitioned into 1000 points in x and 1000 points in t. We represent the computational domain by $N_{x,t} = \{1000, 1000\}$. Each neural network has 7 layers with 30 neurons per layer, uses the **tanh(·)** activation function for non-linear layers, and a learning rate of 0.0005. Lastly, we use the ADAM optimizer for stochastic gradient decent. In table 2, we present the test shock-tube problem, the number of epochs for training the neural network, the original domain, and the transformed extended domain.

Problem	**Epochs**	**Original Domain**, (x,t)	**Extended Domain**, (x,t)
Single Contact Discont.	44,350	$[0,1]\times[0,2]$	$[-2.6175.5, 2.5]\times[0,2]$
Double Expansion Fan	40,165	$[0,1]\times[0,0.2]$	$[-2.625, 2.5]\times[0,0.2]$
Sod shock-tube	76,140	$[0,1]\times[0,0.2]$	$[-1.5, 3.125]\times[0,0.2]$
Reverse Sod shock-tube	76,140	$[0,1]\times[0,0.2]$	$[-2.625, 2.5]\times[0,0.2]$
High-speed shock-tube I	55,765	$[-0.5,1.5]\times[0,0.2]$	$[-2.625, 2.5]\times[0,0.2]$
High-speed shock-tube II	67, 200	$[0,1]\times[0,0.2]$	$[-2.625, 3.125]\times[0,0.2]$

Table II – Test shock-tube problem, the number of epochs for training the neural network, the original domain, and the extended domain

V. SINGLE CONTACT DISCONTINUITY PROBLEM

The single contact discontinuity problem is the first and simplest hydrodynamic shock-tube problem we consider. As shown in table I, the initial state of the density is greater on the left of x^*, than on right, while the velocity and pressure remain constant. This allows the solution of density to be a mere translation of the initial state, thus, the initial shock remains intact. In figure 4, we plot the initial condition for the density term, to demonstrate how well the neural network learns the initial state due to weighting and domain extension. We then plot the final solution at $t = 2$, as well as the top view of the full W-PINNs-DE and exact solution. Table III reveals the relative L_2 error for the density, velocity, and pressure solutions.

This first example, although simple in nature, is quite difficult to solve numerically. This is because most numerical methods will introduce some form of dissipation near points of discontinuity, hence smoothing out the shock. For example, when using the FVM with flux limiting, we require a heavily refined mesh to resolve the shock. Figure 4 demonstrates W-PINNs-DE ability to capture the contact discontinuity with ease, with no noticeable dissipation near the point of discontinuity, and without an immensely refined mesh. A quantitative validation to this claim is shown in table III, where the density term achieves an error which is less than 0.02%.

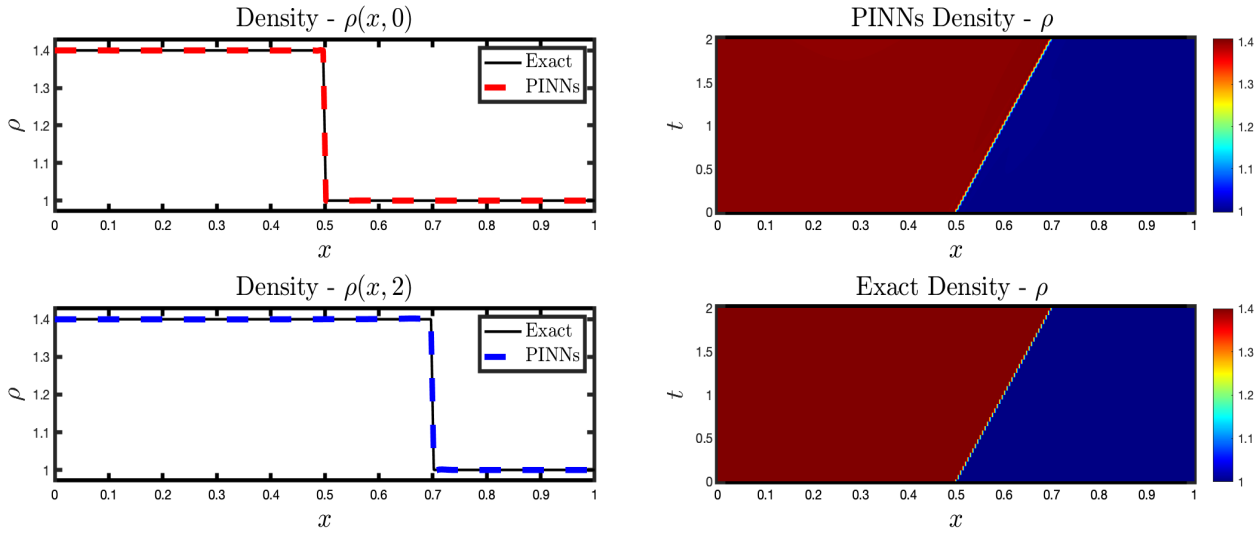

Figure 4 – Solution to the single contact discontinuity problem with sampling, $N_f = 20,000$, $N_{IC} = 1000$

$\frac{\|\rho_{approx}-\rho_{exact}\|_2}{\|\rho_{exact}\|_2}$	$\frac{\|u_{approx}-u_{exact}\|_2}{\|u_{exact}\|_2}$	$\frac{\|p_{approx}-p_{exact}\|_2}{\|p_{exact}\|_2}$
$1.622e-04$	$1.2e-03$	$1.608e-04$

Table III – Relative L_2 error for the single contact discontinuity problem

VI. SOD SHOCK-TUBE PROBLEMS

The Sod shock-tube problem is a standard shock-tube problem, where the initial states of both the density and pressure are greater to the left of x^*, than the right. The solution for each physical quantity develops both a rarefaction fan and shock. Numerically capturing shock and rarefaction fans is difficult, as artificial dispersion or dissipation is created near points of discontinuity by the numerical scheme. In figure 5, the W-PINNs-DE scheme is able to capture the shocks and rarefaction fan with ease for both the density and pressure terms. Although the velocity is approximated quite well, we see an acceptable amount of dissipation near the right shock. In table IV, we see the error for the density and pressure terms is less than 0.9%, while the error for the velocity term is approximately 6%. Typically, solving numerically for the velocity field of a shock-tube problem is quite difficult, thus a 6% error is comparatively small with respect to classical numerical solvers. In fact, the W-PINNs-DE scheme solves this problem more accurately, and in a simpler manner than both the (Michoski et al, 2019) and the (Patel et al, 2020) reported schemes.

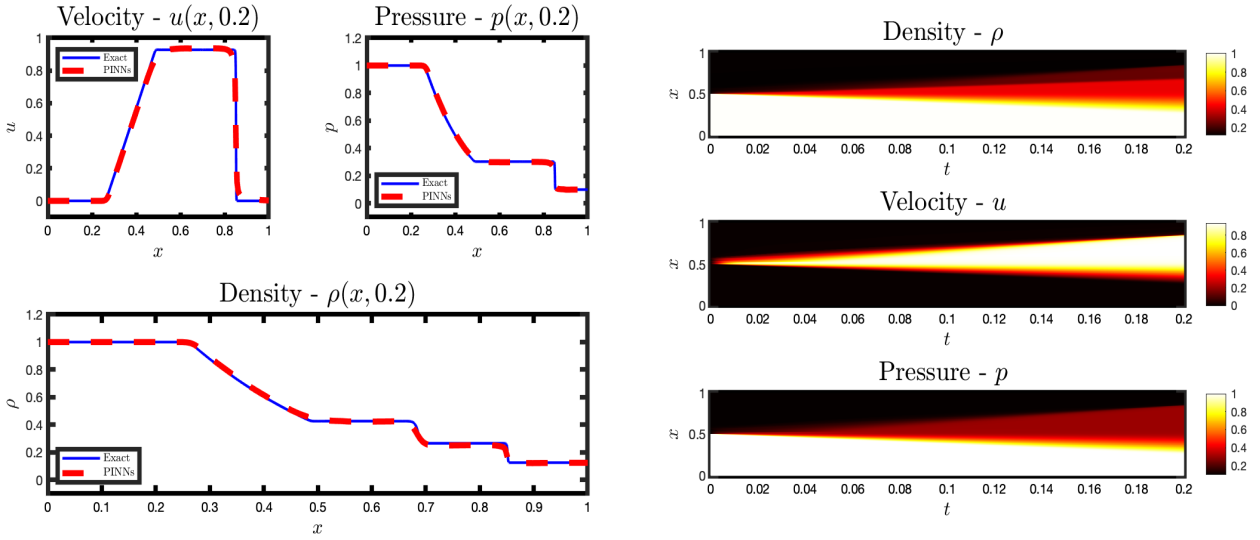

Figure 5 – The Sod shock-tube problem with sampling, $N_f = 11,000$, $N_{IC} = 1000$

$\frac{\|\rho_{approx}-\rho_{exact}\|_2}{\|\rho_{exact}\|_2}$	$\frac{\|u_{approx}-u_{exact}\|_2}{\|u_{exact}\|_2}$	$\frac{\|p_{approx}-p_{exact}\|_2}{\|p_{exact}\|_2}$
$8.6e-03$	$6.29e-02$	$8.2e-03$

Table IV – Relative L_2 error for the Sod shock-tube problem

As we will see in the following shock-tube problems, the velocity is consistently more difficult to learn than the density and pressure terms. In future work, it will be of great interest to resolve the relative velocity error, such that it is as small as that of the relative density and pressure error.

The reverse Sod shock-tube problem reflects the initial states of the classical Sod shock-tube problem with the initial conditions for both the density and pressure now being greater to the right of x^* compared to the left. Similar to the classical Sod shock-tube problem, W-PINNs-DE can capture the shocks and rarefaction fan with ease for the density and pressure terms, whereas the velocity experiences an acceptable amount of dissipation near the shock. Interestingly, we obtain more accurate results for the reverse Sod shock-tube problem than the original problem. When comparing the reverse Sod problem to the classical Sod problem, we see the reverse problem's density term is approximated to the same magnitude of error, whereas the velocity term has 3% less error, and the pressure has 0.2% less error than the original problem. The speculation is that the shape of the **tanh(·)** activation function allows solving this problem more efficiently. When looking at the **tanh(·)** activation function, we see it closely resembles the right portion of the solution for each physical quantity. This is what motivates the hypothesis that the shape of the activation function greatly effects the W-PINNs-DE approximation for hydrodynamic shock-tube problems.

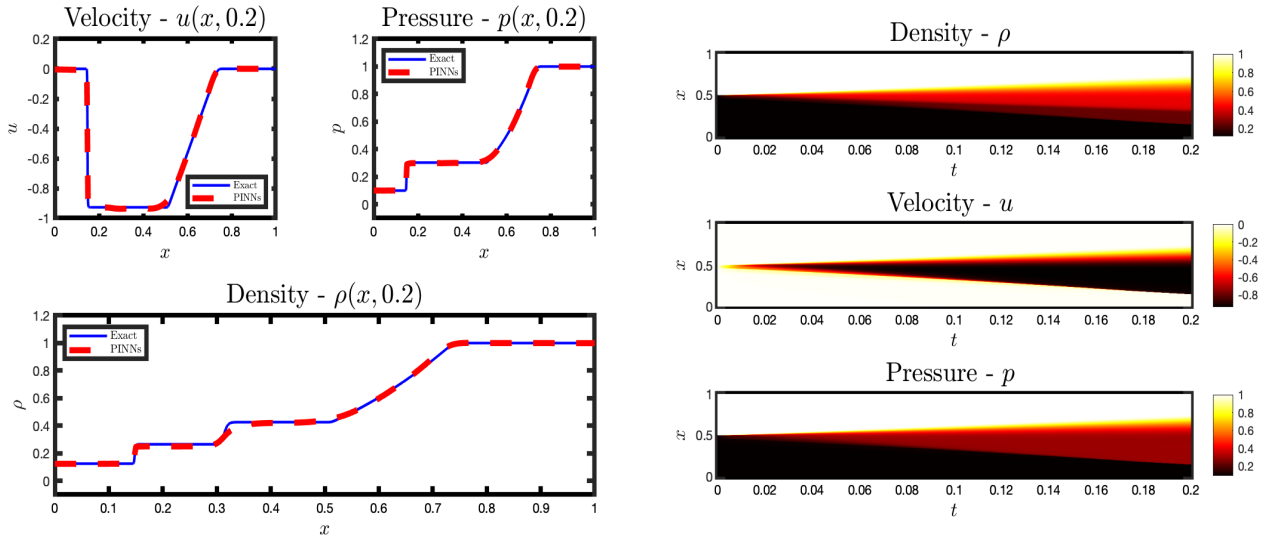

Figure 6 – The reverse Sod shock-tube problem with sampling, $N_f = 10,500$, $N_{IC} = 1000$

$\frac{\|\rho_{approx} - \rho_{exact}\|_2}{\|\rho_{exact}\|_2}$	$\frac{\|u_{approx} - u_{exact}\|_2}{\|u_{exact}\|_2}$	$\frac{\|p_{approx} - p_{exact}\|_2}{\|p_{exact}\|_2}$
$8.5e-03$	$3.04e-02$	$6.6e-03$

Table V – Relative L_2 error for the reverse Sod shock-tube problem

VII. DOUBLE EXPANSION FAN PROBLEM

For the double expansion fan problem, the density and pressure are initially constant and continuous, while the initial velocity is greater to the right of x^* compared to the left. Unlike the single contact discontinuity problem, we observe that the double rarefaction fan problem does not simply translate the initial discontinuity. Figure 7 demonstrates that each of the physical quantity evolves to have two expansion fans. We find that the W-PINNs-DE scheme captures both expansion fans with ease, as this scheme is able to achieve less than 0.3% error for each of the physical quantities, as reported in table VI.

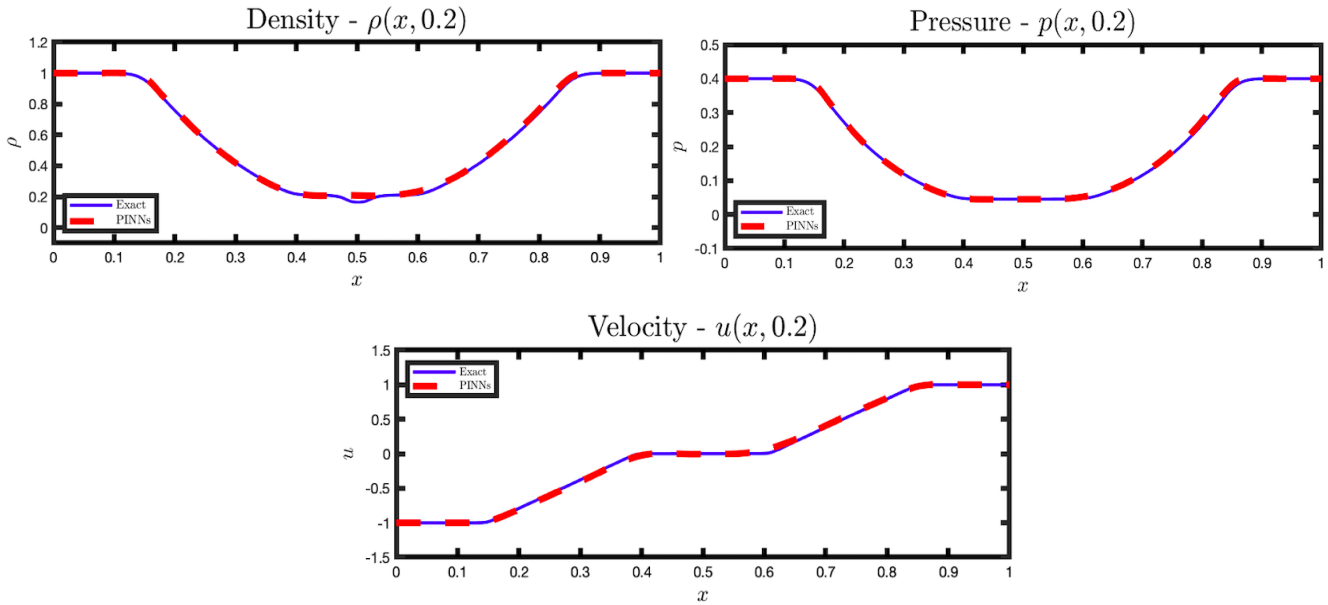

Figure 7 – Solution at $t = 0.2$ for the double expansion fan problem, with sampled points, $N_f = 10,500$, $N_{IC} = 1000$

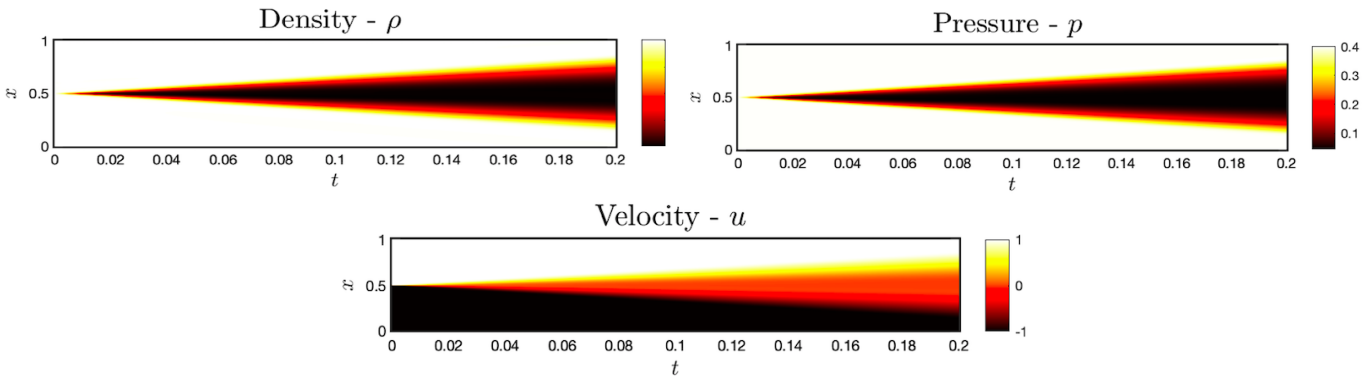

Figure 8 – Top view of solution to the double expansion fan problem

$\frac{\|\rho_{approx}-\rho_{exact}\|_2}{\|\rho_{exact}\|_2}$	$\frac{\|u_{approx}-u_{exact}\|_2}{\|u_{exact}\|_2}$	$\frac{\|p_{approx}-p_{exact}\|_2}{\|p_{exact}\|_2}$
$2.0e-03$	$2.9e-03$	$2.0e-03$

Table VI – Relative L_2 errors for double expansion fan problem

VIII. HIGH-SPEED FLOW PROBLEMS

In this section we present two high-speed flow hydrodynamic shock-tube problems. The first high-speed flow problem we consider is a modified reverse Sod shock-tube problem. The initial state of the density and pressure are equivalent to that of the reverse Sod shock-tube problem, but now the initial velocity is greater on the right of x^* than the left, rather than continuously constant initially. Interestingly, although this problem has a more complicated initial state in comparison to the reverse Sod shock-tube problem, the physics are captured more accurately. We can see from table VII, the relative L_2 errors for the modified reverse Sod shock-tube problem are in-fact smaller than those of the reverse Sod shock-tube problem. Although W-PINNs-DE method captures the solution to the problem accurately, in figure 9, we see there is slight dissipation near the shock on the left.

The second high speed flow problem has the most complex initial state out of all the problems presented in this paper. The reason for this intricacy is due to the fact that this specific shock-tube problem has mixed initial states, where $\rho_R > \rho_L$, and $[u_L, p_L] > [u_R, p_R]$. In figure 10, the W-PINNs-DE method captures each rarefaction fan and shock with little dissipation, and zero dispersion. As reported in Table VIII, the W-PINNs-DE scheme solves for the density and pressure terms with less than 0.8% error, respectively, and the velocity term with less than 2%.

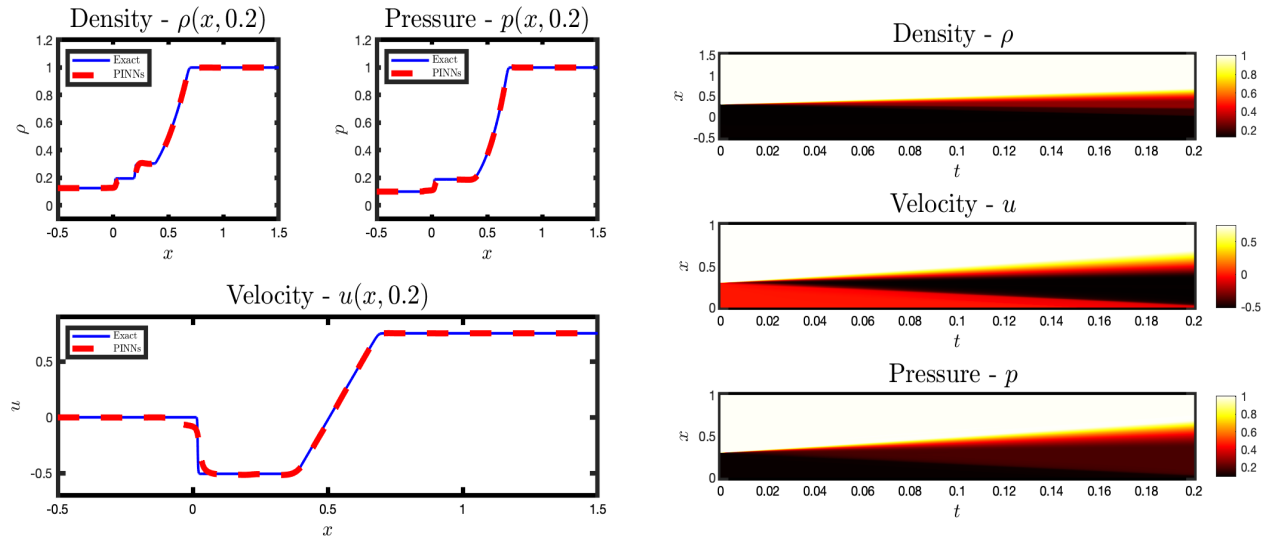

Figure 9 – High-Speed Flow Problem I, with sampled points, $N_f = 10,500$, $N_{IC} = 1000$

$\frac{\|\rho_{approx}-\rho_{exact}\|_2}{\|\rho_{exact}\|_2}$	$\frac{\|u_{approx}-u_{exact}\|_2}{\|u_{exact}\|_2}$	$\frac{\|p_{approx}-p_{exact}\|_2}{\|p_{exact}\|_2}$
$6.7e-03$	$3.29e-02$	$5.0e-03$

Table VII – Relative L_2 errors for High-Speed Flow Problem I

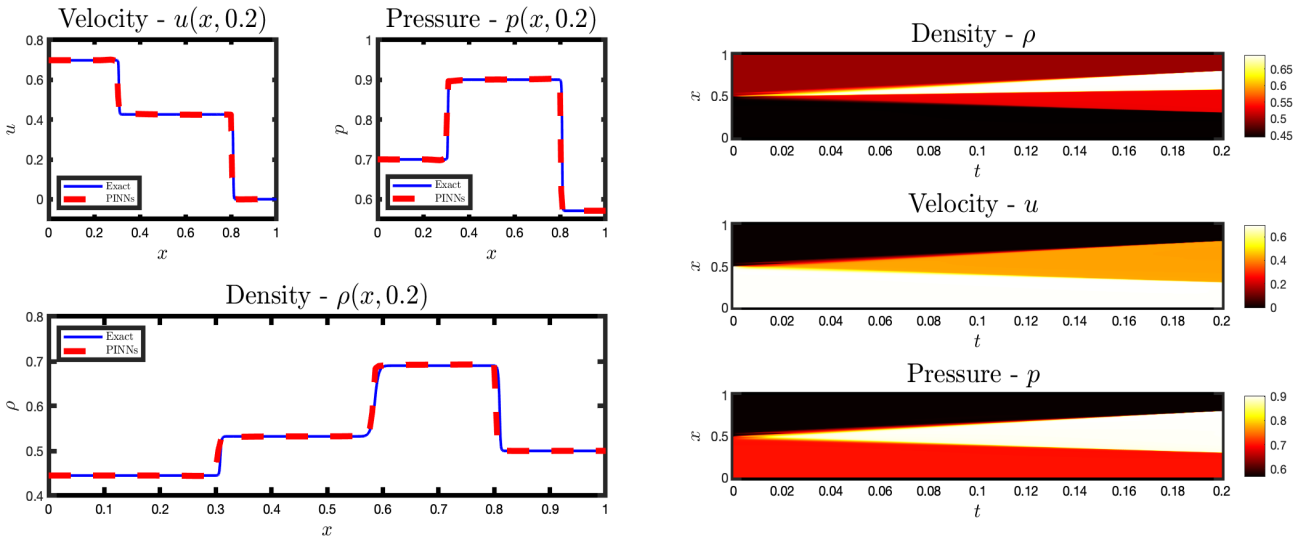

Figure 10 – High-Speed Flow Problem II, with sampled points, $N_f = 11,000$, $N_{IC} = 1000$

$\frac{\|\rho_{approx}-\rho_{exact}\|_2}{\|\rho_{exact}\|_2}$	$\frac{\|u_{approx}-u_{exact}\|_2}{\|u_{exact}\|_2}$	$\frac{\|p_{approx}-p_{exact}\|_2}{\|p_{exact}\|_2}$
$7.7e-03$	$1.4e-02$	$6.9e-03$

Table VIII – Relative L_2 errors for High-Speed Flow Problem II

IX. OBSERVATIONS

In this study, we validated the W-PINNs-DE scheme's ability to solve complicated hydrodynamic shock-tube problems. Using W-PINNs-DE allows for neural networks to solve for discontinuous solutions of compressible Euler equations by capturing physical phenomena such as shocks and rarefaction fans. By introducing a weighted loss function and domain extension, we guarantee that the loss of the PDE decreases at a slower rate than the loss of the initial condition. In figure 11, we plot the total loss, loss of the initial condition, and PDE versus iteration for the Sod shock-tube problem in an effort to demonstrate that the loss of the PDE decreases at a slower rate than that of the initial condition. We have shown through each test case that having this phenomenon occur for the two components of the total loss is essential in solving hydrodynamic shock-tube problems.

Although W-PINNs-DE proves to be a powerful numerical tool for solving conservation laws, each test problem demonstrates higher numerical error for the velocity than the density and pressure. In figures 6-9, the velocity term typically experienced dissipation near the shock corners, whereas W-PINNs-DE captures the rarefaction fan exceptionally. Although this error is acceptable, it is a goal to find as accurate solutions as possible. Hence, understanding this phenomenon is an essential task for the author. Moreover, we desire a rigorous explanation as to why W-PINNs-DE captures these solutions to such accuracy compared to classical PINNs and other modifications to the method. Another curiosity is, the two high-speed flow problems had three discontinuous states and developed more physics in their solutions than the Sod shock-tube problems. Yet, despite the initial state and final answers' complexity, W-PINNs-DE captures the approximate solution to the high-speed flow problems with higher accuracy.

To summarize, this study demonstrated that W-PINNs-DE solves complex shock-tube problems with an impressive degree of accuracy, making it a competitive numerical tool suited in tackling conservation laws based PDEs. In addition to solving hydrodynamic shock-tube problems, W-PINNs-DE can solve isothermal Euler equations and advection-dominated compressible Navier-Stokes equations.

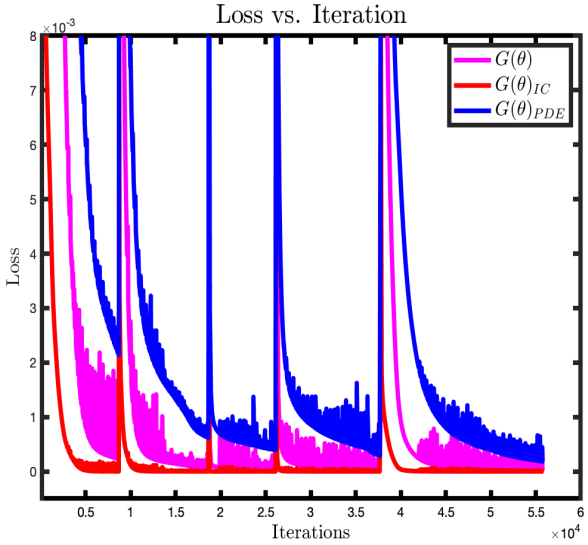

Figure 11 – Loss vs. Iteration: pink is the total loss, blue is the loss of the PDE, and red is the loss of the initial condition

X. SOFTWARE AND CODING LANGUAGES

PyTorch is a useful package for data science and machine learning. It provides easy-to-use functions for training, backpropagation, and auto-differentiation. Hence, all W-PINNs-DE code was written using PyTorch, and therefore, Python was the only coding language used to obtain W-PINNs-DE solutions for the findings of this paper. Numerical solutions using classical numerical methods and analytic solutions were obtained using MATLAB. All W-PINNs-DE code from this paper is available on Github at: https://github.com/alexpapados/W-PINNs-DE-Hydrodynamic-Shock-Tube-Problems

XI. FUTURE WORK

In future work, we aspire to extend W-PINN-DE into higher dimensions and simulate real-life problems, such as high energy release (denotation) problems. It is also essential to have a more robust theoretical background for this method than currently available. Therefore, it will be of great interest to further investigate and refine W-PINNs-DE mathematical formulation. This paper is merely the first reporting of the current research effort. In later work, we will present results for the isothermal Euler and Navier-Stokes equations solved using W-PINNs-DE. The author's future work will include reporting the use of PINNs to solve linear elasticity boundary value problems and provide a comparison to equivalent solutions based on the finite element method.

XII. REFERENCES

[1] Roesner, K. G, Leutloff, D, Srivastava, R. C. (1995). *Computational fluid dynamics: Selected topics.* Berlin: Springer.

[2] Chen, Y, Press, H. H. (2013). *Computational Solid Mechanics Structural Analysis and Algorithms.* Berlin: De Gruyter.

[3] Thomas, J. W. (1999). *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations.* New York: Springer.

[4] Golsorkhi, N. A, Tehrani, H. A. (2014). *Levenberg-marquardt Method For Solving The Inverse Heat Transfer Problems.* Journal of Mathematics and Computer Science, 13(04), 300-310. doi:10.22436/jmcs.013.04.03

[5] Chen, Z. (2010). *Finite Element Methods and their Applications.* Berlin: Springer.

[6] Mao, Z, Jagtap, A. D, Karniadakis, G. E. (2020). *Physics-informed neural networks for high-speed flows.* Computer Methods in Applied Mechanics and Engineering, 360, 112789. doi:10.1016/j.cma.2019.112789

[7] Raissi, M, Perdikaris, P, Karniadakis, G. (2019). *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. doi:10.1016/j.jcp.2018.10.045

[8] Sirignano, J, Spiliopoulos, K. (2018). *DGM: A deep learning algorithm for solving partial differential equations.* Journal of Computational Physics, 375, 1339-1364. doi:10.1016/j.jcp.2018.08.029

[9] Lu, L,Jagtap, A. D, Karniadakis, G. E. (2019). *DeepXDE: A Deep Learning Library for Solving Differential Equations.* ArXiv.org,arxiv.org/abs/1907.04502.

[10] Cybenko, G. (1989). *Approximation by superpositions of a sigmoidal function.* Mathematics of Control, Signals, and Systems, 2(4), 303-314. doi:10.1007/bf02551274

[11] Mishra, S, Molinaro, R. (2020). *Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs II: A class of inverse problems.* https://arxiv.org/abs/2007.01138

[12] Pinkus, A. (1999). Approximation theory of the MLP model in neural networks. Acta Numerica, 8, 143-195. doi:10.1017/s0962492900002919

[13] Sod, G. A. (1978). A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. Journal of Computational Physics, 27(1), 1-31. doi:10.1016/0021-9991(78)90023-2

[14] LeVeque, R. J. (2011). Finite volume methods for hyperbolic problems. Cambridge: Cambridge Univ. Press.

[15] Michoski, C, Milosavljević, M, Oliver, T, amp; Hatch, D. R. (2020). Solving differential equations using deep neural networks. Neurocomputing, 399, 193-212. doi:10.1016/j.neucom.2020.02.015

[16] Patel, R. G, Manickam, I, Trask, N, amp; Wood, M. A. (2020). Thermodynamically consistent physics-informed neural networks for hyperbolic systems. doi:https://arxiv.org/abs/2012.05343

APPENDIX

This appendix is dedicated to displaying the Mach number and total energy of each problem. The red dashed lines are the W-PINNs-DE approximation, and the blue line is the exact solution of the two quantities. In figures 14-17, we see that W-PINNs-DE satisfied the conservation of energy, verifying once more the accuracy of the method

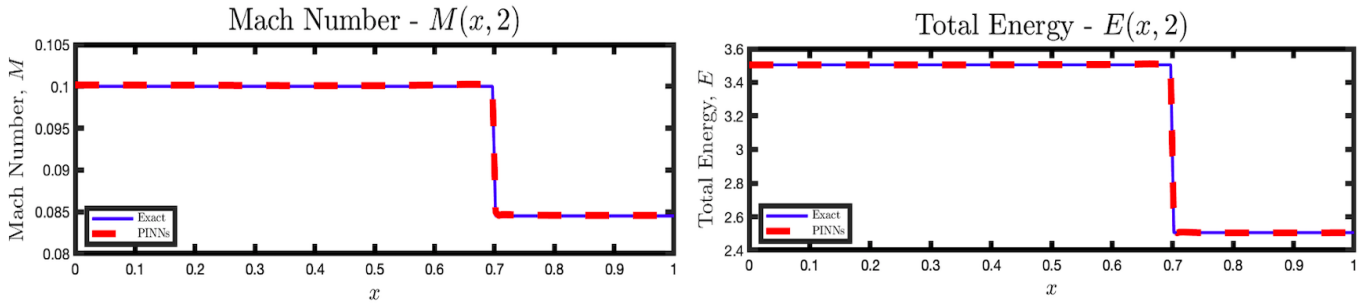

Figure 12 – Mach number and total energy at $t = 2.0$ for the single contact discontinuity problem

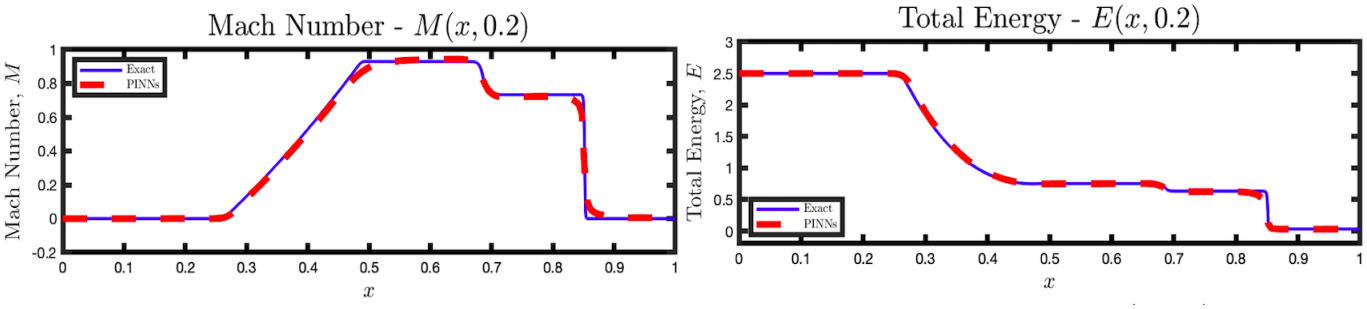

Figure 13 – Mach number and total energy at $t = 0.2$ for the Sod shock-tube problem

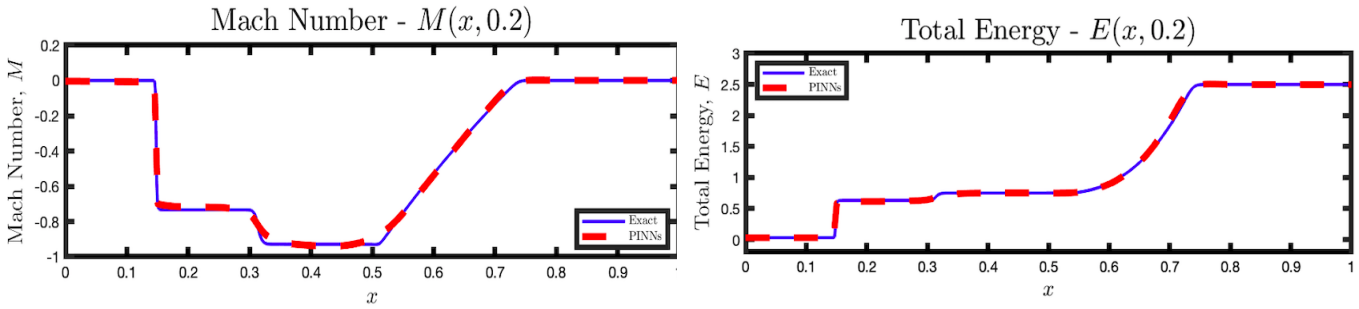

Figure 14 – Mach number and total energy at $t = 0.2$ for the reverse Sod shock-tube problem

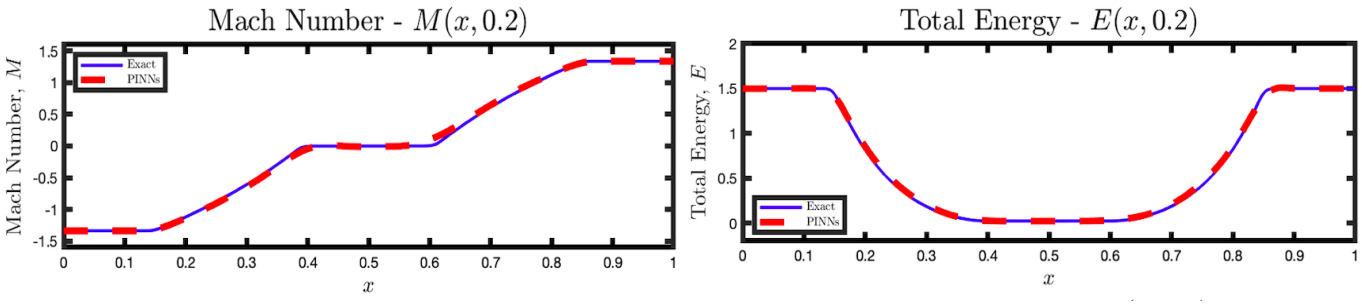

Figure 15 – Mach number and total energy at $t = 0.2$ for the double expansion fan problem

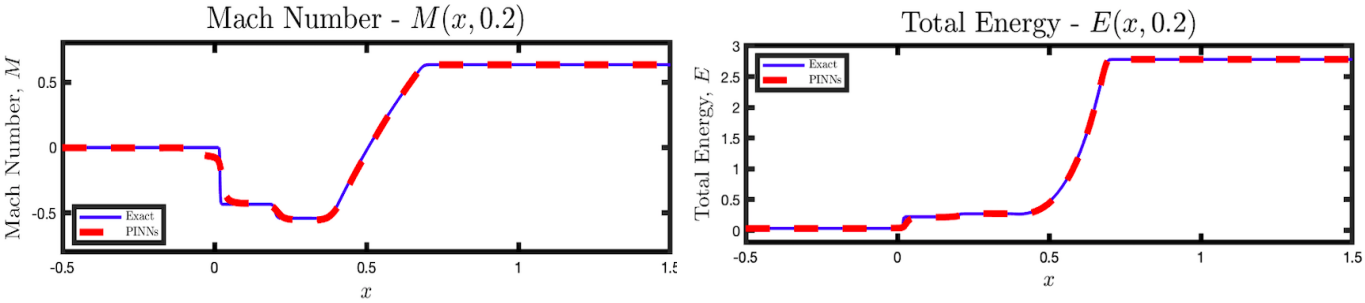

Figure 16 – Mach number and total energy at $t = 0.2$ for the high-speed flow problem I

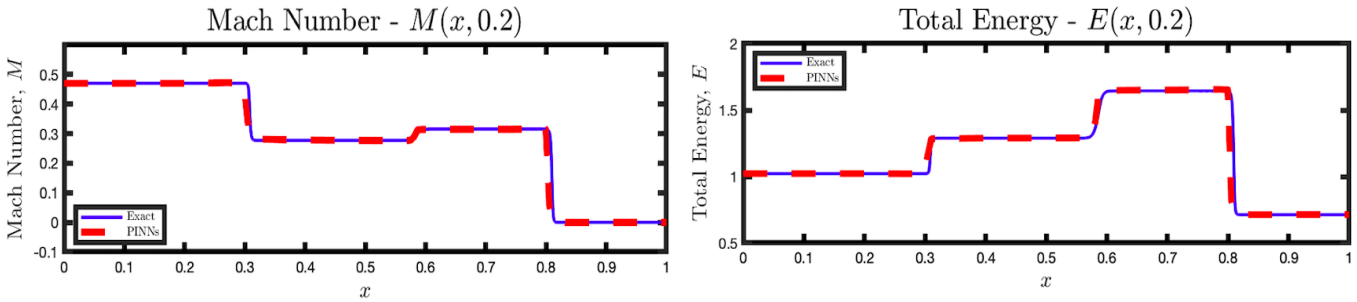

Figure 17 – Mach number and total energy at $t = 0.2$ for the high-speed flow problem II