

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: April 19, 2021)

This paper focuses on 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 PINNs solver which 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 numerous computational problems. Hence, researchers today incorporate machine learning in several scientific areas, such as extracting physical parameters from experimental data for numerical simulations [18], detecting cracks in the specific regions of a material [17], and, to the interest of this paper, solving a highly nonlinear system of partial differential equations (PDEs) [7].

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. These features make PINNs a competitive numerical tool compared to standard methods, such as the finite volume method [14] for forward problems and nonlinear least-squares approaches [4] for inverse problems. The developers of PINNs constructed the method 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 [11][15][16].

Although PINNs has become a popular method for solving PDEs, it has its limitations as any classical numerical method. 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 [10][12] specifies that neural networks are universal approximators for solutions to PDEs that are *continuous*. As of today, 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. For example, 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-tube problems and explosions. The finite volume method (FVM) (LeVeque, 2011) is often the method of choice to solve these problems, but it needs adjustments to its original numerical formulation in order to propagate shock and rarefaction waves correctly. The so-called *flux-limiting* is introduced to resolve artificial dispersion and dissipation created by the FVM schemes. Similarly, PINNs without modifications develop similar phenomena when computing for solutions that include shocks and rarefaction fans. Therefore, for PINNs to be competitive with the FVM with flux-limiting, we must modify the method to solve complicated problems in fluid dynamics, particularly hydrodynamic shock-tube problems.

To the best of my knowledge, (Michoski et al., 2019) and (Patel et al., 2020) are the only other publications introducing modifications to PINNs to solve the classic hydrodynamic shock-tube test problem, the *Sod Shock-Tube Problem* [13]. Both papers discuss modifications to rectify for spurious oscillations near shocks and contact discontinuities, generated by the original PINNs formulation during training. (Michoski et al., 2019) uses artificial viscosity to dissipate oscillations, by adding viscous terms to the Euler equations. Introducing artificial viscosity is in itself a non-physical adjustment to the mathematical formulation. 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 a FVM total variation diminishing formulation and artificial viscosity with PINNs. The merger further complicates

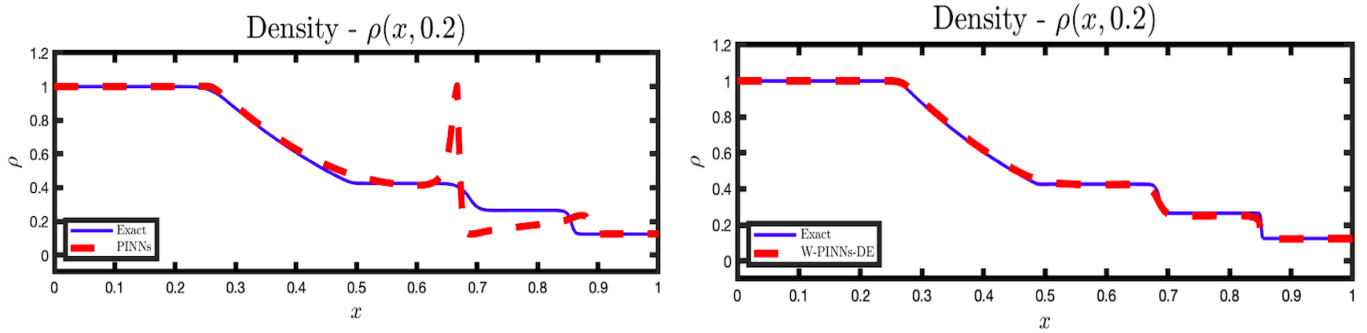


Figure 1 – The left figure is the solution of the density of the Sod Shock-tube problem using the standard formulation of PINNs. The right figure is the solution of the density of the Sod Shock-tube problem using W-PINNs-DE

the PINNs algorithm and, thus, takes away from the simplicity of the PINNs method. Both of these adjustments are designed to solve the specific shock-tube problems addressed in the corresponding papers and do not seem to be easily generalizable to a broader class of shock-tube problems. In this work, we 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 the two papers mentioned above. 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 Sod shock-tube problem.

We will test the proposed approach on six hydrodynamic shock-tube test problems. The proposed modifications consists of two components:

- Incorporate weights to the total loss function, such that the loss of the initial condition is minimized faster than the loss of the PDE.
- Extend the original spatial domain to introduce additional training points to the neural network in order to reduce spurious oscillations.

Allowing the loss of initial condition to decrease faster than that of the PDE ensures the neural network learns the initial condition data accurately. Having the neural network learn the initial condition impeccably is essential to minimize the error of the solution at future time, $t > 0$. Domain extension smooths out the spurious oscillations by introducing the neural network with more points to the left and right of the initial point of discontinuity, x^* . The additional training points take on the values of the boundary condition since the boundary conditions for each problem presented are equivalent to the boundaries of the initial condition. The spatial domain extends depending on the initial state, \mathbf{U}_0 .

The rest of the paper is organized as follows. Section II will provide a brief introduction to PINNs, its algorithm, and theoretical limitations. In section III, we present W-PINNs-DE by constructing the method and its algorithm. Section III also discusses the architecture of W-PINNs-DE for each test problem presented in this paper. In sections IV, we solve each of the six hydrodynamic shock-tube test problems using the proposed method, and compare it to the original PINNs formulation, as well as the FVM with flux limiting. Here we will present figures of the solutions, the relative L_2 error for each physical quantity, ρ, u , and p , and error analysis. In section V, we discuss results presented in section IV and additional error analysis for W-PINNs-DE. In the appendix, we compare the W-PINNs-DE mach and total energy solutions versus the exact solutions.

II. BACKGROUND

A. Pinkus proved that a continuous function and its derivatives may be approximated by any single layered (hence a multilayered) neural network (Pinkus, 1999). 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*.

To understand the PINNs formulation, let us consider a general nonlinear PDE:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + \mathcal{L}(\mathbf{u}) = 0, & (x, t) \in \Omega \times (0, T] \\ \mathbf{u}(x, t) = \mathbf{g}(x, t), & (x, t) \in \partial\Omega \times (0, T] \\ \mathbf{u}(x, 0) = \mathbf{h}(x), & x \in \Omega \end{cases} \quad (1)$$

where \mathcal{L} is a nonlinear differential operator, $\Omega \subset \mathbb{R}^d$, and $u(x, t)$ is the exact solution to (1). To solve (1) using PINNs, we construct a deep neural network (DNN), $\tilde{\mathbf{u}}(x, t, \theta)$, where $\theta \in \mathbb{R}^k$ are network weights, (x, t) are inputs to the network,

and $\tilde{\mathbf{u}} = [\tilde{u}_1(x, t), \dots, \tilde{u}_n(x, t)]$ is the output. The standard loss (Karniadakis et al., 2019) is defined by:

$$G(\theta) = \frac{1}{N_f} \left\| \frac{\partial \tilde{\mathbf{u}}}{\partial t}(x, t, \theta) + \mathcal{L}(\tilde{\mathbf{u}}(x, t, \theta)) \right\|_{\Omega \times (0, T], \nu_1}^2 + \frac{1}{N_{IC}} \left\| \tilde{\mathbf{u}}(x, 0, \theta) - \mathbf{h}(x) \right\|_{\Omega, \nu_2}^2 + \frac{1}{N_{BC}} \left\| \tilde{\mathbf{u}}(x, t, \theta) - \mathbf{g}(x, t) \right\|_{\partial\Omega \times (0, T], \nu_3}^2 \quad (2)$$

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 . We then minimize (2) to optimize θ through the use of stochastic gradient decent. After an appropriate number of epochs, the neural network should output an approximate solution to (1).

Unfortunately, the theorem of Pinkus does not generalize to solutions to PDEs that are discontinuous. This restricts the current PINNs framework to solve problems with only continuous solutions. Hence, W-PINNs-DE were developed to bypass theoretical and computational limitations of the original PINNs method by approximating solutions of a general class of discontinuous problems referred to as hydrodynamic shock-tube problems.

III. W-PINNS-DE

A. Euler Equations

Consider the 1-D compressible Euler equations in characteristic form, where $\Omega \subset \mathbb{R}$:

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A} \frac{\partial \mathbf{U}}{\partial x} = 0, \quad (x, t) \in \Omega \times (0, T] \quad (3)$$

where,

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

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:

$$\mathbf{U}(x, 0) = \mathbf{U}_0 = \begin{cases} \mathbf{u}_L, & x < x^* \\ \mathbf{u}_R, & x > x^* \end{cases}, \quad \mathbf{u}_L = [\rho_L, u_L, p_L], \quad \mathbf{u}_R = [\rho_R, u_R, p_R] \quad (4)$$

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

B. Domain Extension

First, we will discuss what it means to extend the spatial domain of the problem, Ω . We extend Ω in a manner such that if $\mathbf{u}_L > \mathbf{u}_R$, Ω is extended such that the initial state leans to the left of the newly extended spatial domain, Ω_e . If $\mathbf{u}_R > \mathbf{u}_L$, Ω is extended such that the initial state leans to the right of Ω_e . In the case where we have a mixture of initial states, we will extend the spatial domain such that each initial state leans towards the dominate direction of propagation. Since the initial state, \mathbf{U}_0 , varies but remains constant to the right and left of the point of discontinuity, x^* , we extend Ω so the neural network trains on sufficiently many points that remain constant, and hence are easily learned by the network. By the time the neural network reaches x^* from the right and left, it will encounter a sharp change in gradient (a discontinuous change). Extending Ω smooths out any artificial oscillations that are typically generated when solving within the original spatial domain. By eliminating spurious oscillations in the solutions, we further minimize the total loss of the neural network. 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, extending the spatial domain with regards to the initial condition is a cheaper and more effective way of diffusing spurious oscillations when solving hydrodynamic shock-tube problems using PINNs. In table I, we define each test problems original domain, and subsequently its newly extended domain based on the initial state of the problem.

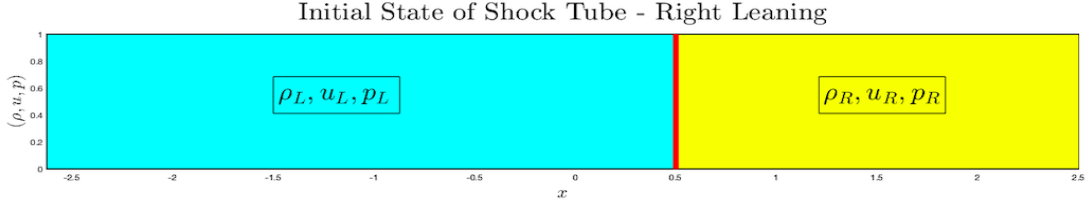


Figure 2 – $u_L < u_R \implies$ Right Leaning Extension, ex) $[0, 1] \rightarrow [-2.625, 2.5]$

Let $\Omega \equiv (a, b)$ be the original spatial domain, and $x^* \in \Omega$ is the point of discontinuity from (4). Then, we have the spatial domain transformation:

$$\Omega \longrightarrow \Omega_e$$

where $\Omega_e = (a_e, b_e)$. In figure 2, we present a right leaning domain extension.

C. Weighting

The second modification we make to PINNs is with regards to the weights of the total loss function. 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. Similarly to (2), the standard loss is defined by:

$$G(\theta) = \frac{1}{N_f} \left\| \frac{\partial \tilde{U}}{\partial t}(x, t, \theta) + \tilde{A} \frac{\partial \tilde{U}}{\partial x}(x, t, \theta) \right\|_{\Omega \times (0, T], \nu_1}^2 + \frac{1}{N_{IC}} \left\| \tilde{U}(x, 0, \theta) - U(x, 0) \right\|_{\Omega, \nu_2}^2 + \frac{1}{N_{BC}} \left\| \tilde{U}(x, t, \theta) - U(x, t) \right\|_{\partial \Omega \times (0, T], \nu_3}^2 \quad (5)$$

We will refer to the first, second, and third components in (5) as $G_f(\theta)$, $G_{IC}(\theta)$, and $G_{BC}(\theta)$, respectively. Since the boundary conditions are induced by the initial conditions, we drop boundary condition term in (5). Hence,

$$G(\theta) = G_f(\theta) + G_{IC}(\theta) \quad (6)$$

Our numerical experiments have shown that $G_f(\theta)$ decreases at a faster rate than the $G_{IC}(\theta)$ when using (5) and (6). This is problematic since the user provides the initial condition data to the neural network. Hence, the neural network should focus its training on data that is provided, then attempt to establish the solution of the PDE at $t > 0$. Moreover, if the initial condition is learned incorrectly, we cannot expect to have the solution at the final time to be accurate. This is trivial since hyperbolic PDEs are highly dependent on its initial state. The solution at future time, $t > 0$, will evolve based on the initial state. Therefore, minimizing $G_{IC}(\theta)$ at a faster rate than that of $G_f(\theta)$ is required to yield fast converging, accurate solutions for shock-tube problems.

Due to the initial discontinuous state, the neural network has much difficulty learning the initial condition since at x^* , the neural network is nondifferentiable in the strong sense. This is to be expected since the universal approximation theorem [10] [12] does not specify whether a neural network can approximate any discontinuous function. To get around this theoretical limitation, we introduce a disproportionate penalty to $G_{IC}(\theta)$, while relaxing the penalty of $G_f(\theta)$, so 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 learned with ease. Therefore, we use the following weighted loss function:

$$G(\theta) = \omega_f G_f(\theta) + \omega_{IC} G_{IC}(\theta) \quad (7)$$

where $\omega_f = 0.1$, $\omega_{IC} = 10$. Numerical experimentation shown that setting ω_{IC} to be 100 times greater than ω_f is optimal for training.

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 $\|U_0\|^2 > 10$, or blast waves), we expect the following inequality for a sufficient number of epochs (training iterations):

$$0 \leq G_{IC}(\theta) \leq G_f(\theta) \quad \forall \theta \in \mathbb{R}^k \quad (8)$$

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. Figure 1 presents the solution of the density of the Sod shock-tube problem. PINNs failure to capture the shock and contact-discontinuity results from an inability to satisfy (8). In figure 3, we compare $G(\theta)$, $G_f(\theta)$, and $G_{IC}(\theta)$ versus training iterations for PINNs and W-PINNs-DE when solving the Sod shock-tube problem. For each iteration, W-PINNs-DE satisfy (8) whereas PINNs do not.

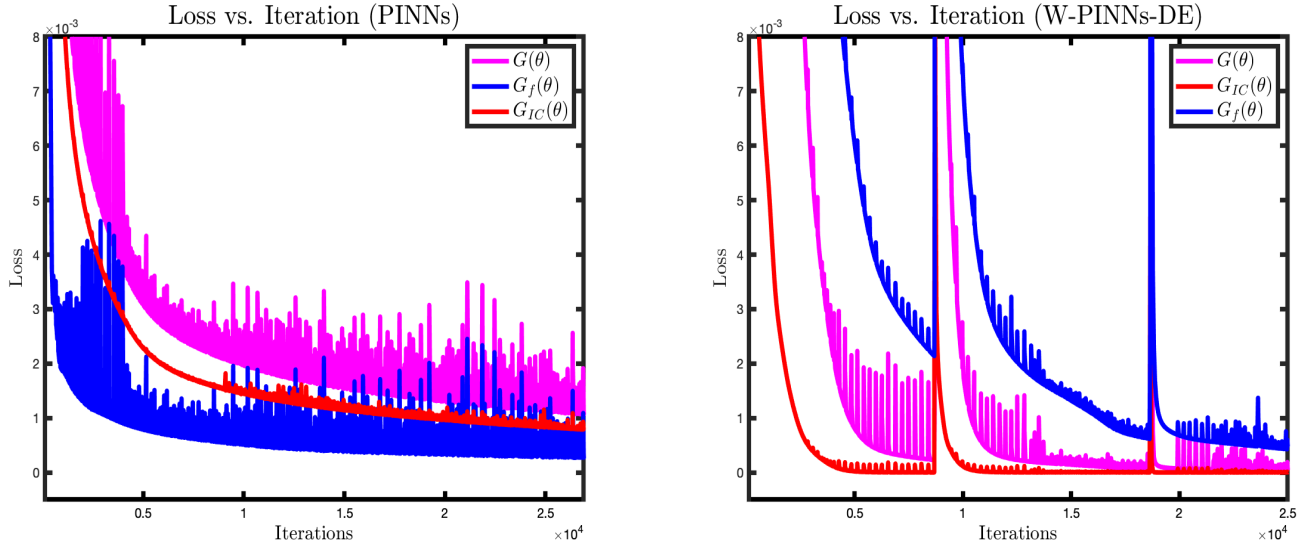


Figure 3 – Loss vs. Iteration: PINNs (Left), W-PINNs-DE (Right)

D. 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 $\tanh(\cdot)$ activation function for non-linear layers, and a learning rate of 0.0005. Taking less than 7 layers resulted in high approximation error and solutions of zero physical meaning. Although prescribing more layers provided accurate solutions, they were computationally expensive. Thus, 7 layers provide accurate results, without being highly computationally intensive. In order to validate the use of the $\tanh(\cdot)$ activation function, various activation functions such as RELU and sigmoid were utilized for each shock-tube problem. However, the $\tanh(\cdot)$ activation function proved most suitable for allowing neural networks to approximate solutions to hydrodynamic shock-tube problems. Lastly, we use the ADAM optimizer for stochastic gradient descent. In table I, 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, $\Omega \times (0, T]$	Extended Domain, $\Omega_e \times (0, T]$
Single Contact Discont.	44,350	$[0, 1] \times [0, 2]$	$[-2.6175, 2.5] \times [0, 2]$
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 I – Test shock-tube problem, the number of epochs for training the neural network, the original domain, and the extended domain

E. W-PINNs-DE Algorithm

To summarize, W-PINNs-DE extends the original domain so the neural network trains on additional points that take on the values of the boundary condition and the boundaries of the initial condition. These additional training points are used to smooth oscillations near points of discontinuity and, thus, minimize the total loss more efficiently. Extending the spatial 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 $G_{IC}(\theta)$ to decrease at a faster rate than $G_f(\theta)$.

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 Define the loss function as:

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

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.

IV. HYDRODYNAMIC SHOCK-TUBE TEST PROBLEMS

In this section, we compare W-PINNs-DE to the original PINNs formulation, as well as the FVM with flux limiting. We will use the HLLE flux solver, with the Osher flux limiter, and RK-3 for time integration for FVM solutions [14]. When solving each test problem using W-PINNs-DE and PINNs, we sample the same number of points from the computational domain $N_{x,t} = \{1000, 1000\}$. The sampling size is presented in each solution figure. Due to the FVM stability dependency on the CFL condition [14], we will partition the spatial computational domain into 1000 points, where as the number of points partitioned in time will vary based on the respective CFL condition.

To ensure that a correct 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 FVM solution. Specifically, we measure:

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

We will also measure the relative L_2 error for other numerical methods presented. Table II establishes each test problem presented in this paper by defining the initial left and right state for the density, velocity, and pressure, respectively.

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

A. Single Contact Discontinuity Problem

The single contact discontinuity problem is the simplest hydrodynamic shock-tube problem we consider. As shown in table II, 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 the density to be a mere translation of the initial state, thus, the initial discontinuity

remains intact, for $t > 0$. In figure 4, we compare W-PINNs-DE with PINNs and the FVM to demonstrate their ability to capture the initial and final state of the density. We then plot the top view of the full W-PINNs-DE and exact solution. Table IV reveals the relative L_2 errors for the density, velocity, and pressure solutions of each method.

While the exact solution to this problem is simple to calculate analytically, it is difficult to solve numerically. This is because numerical methods, typically, introduce dispersion or dissipation near points of discontinuity. The FVM creates immense dissipation at $t = 2.0$. Moreover, the FVM with flux limiting requires a heavily refined mesh to resolve the final contact discontinuity. Figure 4 demonstrates W-PINNs-DE ability to capture the contact discontinuity with minimal dissipation and dispersion, and without an immensely refined mesh. Although the original PINNs method has a smaller relative L_2 error for the density than the FVM, dispersion at the top and bottom of the discontinuity is apparent for each t . In table IV, we find that W-PINNs-DE has the smallest error for the density in comparison to the other methods, resulting from minimal dispersion and dissipation for each t . Despite the fact the FVM solves for the velocity and pressure more accurately than W-PINNs-DE and PINNs, it is important to recognize that the FVM takes the initial data as it is and solves iteratively to obtain the final solutions, where as W-PINNs-DE and PINNs trains to fit the initial data. Thus, the error accumulated from the FVM is merely machine error, where as W-PINNs-DE and PINNs accumulates approximation error from training or fitting the initial conditions.

ρ_L	u_L	p_L	ρ_R	u_R	p_R	$x \in \Omega$	$x \in \Omega_e$	x^*
1.4	0.1	1.0	1.0	0.1	1.0	(0, 1)	(-2.6175, 2.5)	0.5

Table III – Initial conditions for the single contact discontinuity problem

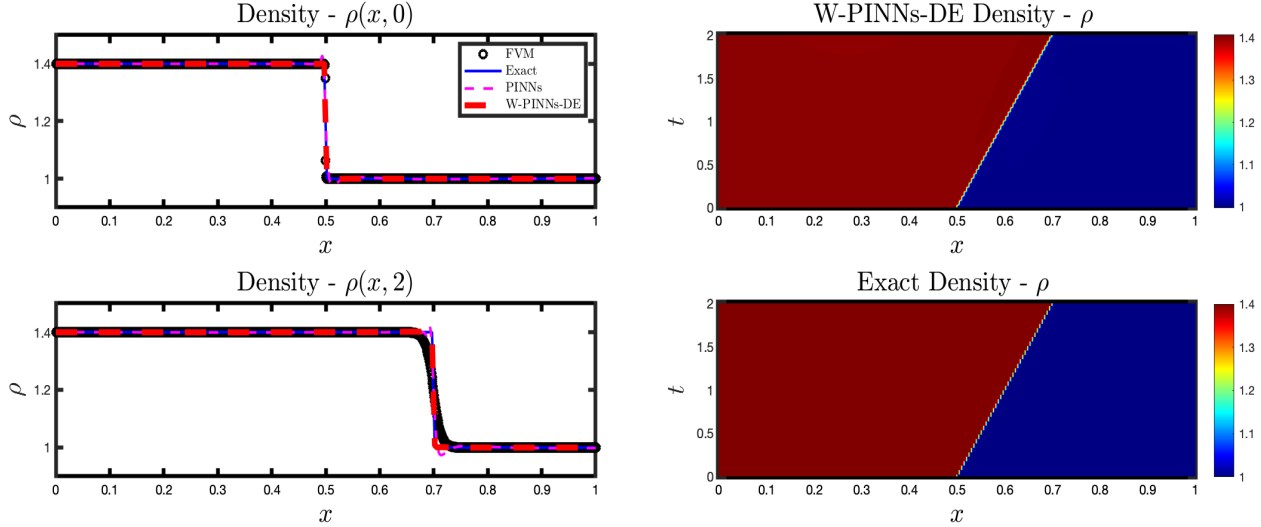


Figure 4 – Solution to the single contact discontinuity problem with sampling, $N_f = 20,000$, $N_{IC} = 1000$

Method	$\frac{\ \rho_{approx} - \rho_{exact}\ _2}{\ \rho_{exact}\ _2}$	$\frac{\ u_{approx} - u_{exact}\ _2}{\ u_{exact}\ _2}$	$\frac{\ p_{approx} - p_{exact}\ _2}{\ p_{exact}\ _2}$
FVM	$3.5e - 03$	$5.3e - 05$	$3.6e - 06$
PINNs	$1.1e - 03$	$1.3e - 03$	$8.4e - 04$
W-PINNs-DE	$1.6e - 04$	$1.2e - 04$	$1.6e - 04$

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

B. Sod Shock-Tube Problem

The Sod shock-tube problem [13] is a standard hydrodynamic shock-tube test problem. Researchers use the Sod problem to validate a numerical methods ability to capture characteristics unique to solving conservation laws. The solution for each physical quantity develops a rarefaction fan, contact discontinuity, and shock. As mentioned in section III.A, numerically capturing the shock, contact discontinuity and rarefaction fan is difficult, as artificial dispersion or dissipation is created

near points of discontinuity by the numerical scheme. Therefore, designing a scheme that captures each discontinuous component is essential in order to obtain accurate numerical solutions.

In figure 5, W-PINNs-DE captures each discontinuous component for both the density and pressure terms with ease. Although the velocity is approximated quite well, we see an acceptable amount of dissipation near the right shock. In table VI, we find the relative L_2 error for the W-PINNs-DE density and pressure is less than 0.8%, while the relative L_2 error for the velocity is less than 6%. Typically, solving numerically for the velocity field of a shock-tube problem is challenging. Hence, a relative L_2 error of less 6% error is acceptable and competitive with respect to classical numerical solvers. W-PINNs-DE performs slightly better than the FVM, whereas the standard PINNs method creates immense dispersion at the contact discontinuity and shock, resulting in a large L_2 error for each physical quantity. Moreover, W-PINNs-DE 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.

ρ_L	u_L	p_L	ρ_R	u_R	p_R	$x \in \Omega$	$x \in \Omega_e$	x^*
1.0	0.0	1.0	0.125	0.0	0.1	(0,1)	(-1.5,3.125)	0.5

Table V – Initial conditions for the Sod shock-tube problem

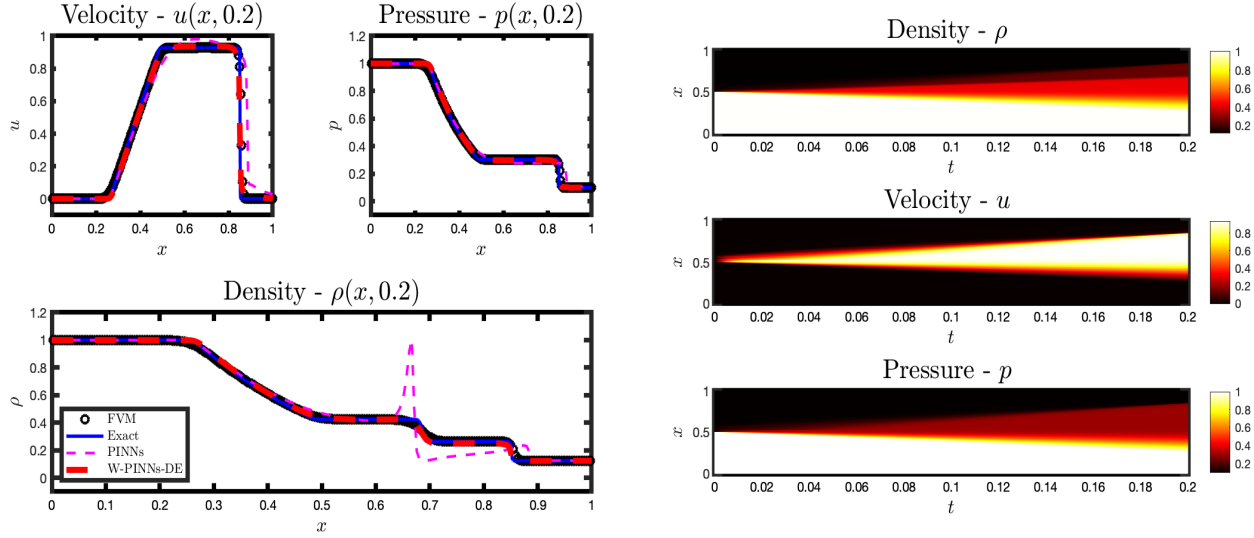


Figure 5 – The Sod shock-tube problem with sampling, $N_f = 11,000$, $N_{IC} = 1000$

Method	$\ \rho_{approx} - \rho_{exact}\ _2$	$\ u_{approx} - u_{exact}\ _2$	$\ p_{approx} - p_{exact}\ _2$
	$\ \rho_{exact}\ _2$	$\ u_{exact}\ _2$	$\ p_{exact}\ _2$
FVM	$8.7e-03$	$4.9e-02$	$8.7e-03$
PINNs	$1.2e-01$	$2.4e-01$	$5.8e-02$
W-PINNs-DE	$8.6e-03$	$5.1e-02$	$7.2e-03$

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

C. Reverse Sod Shock-Tube Problem

The reverse Sod shock-tube problem reflects the initial state of the classical Sod shock-tube problem so that the initial density and pressure are now greater to the right of x^* than the left. Similar to the classical Sod shock-tube problem, W-PINNs-DE captures the shock, contact discontinuity, and rarefaction fan for the density and pressure with ease, whereas the velocity solution experiences an acceptable amount of dissipation near the shock. Table VIII reveals W-PINNs-DE outperforms PINNs and the FVM. PINNs experience immense dispersion at the shock, resulting in a high relative L_2 error. The FVM and W-PINNs-DE add slight dissipation at the contact discontinuity, but W-PINNs-DE proves to capture the shock and rarefaction fan more accurately and without a heavily refined mesh.

Interestingly, PINNs and W-PINNs-DE obtain more accurate results for the reverse Sod shock-tube problem than the original problem, whereas the FVM performs worse. Numerical experiments validated the use of the $\tanh(\cdot)$ activation

function in comparison to other functions such as RELU and sigmoid. The $\tanh(\cdot)$ activation function proved to be most suitable for allowing neural networks to approximate solutions to hydrodynamic shock-tube problems. It is speculated that the geometry of the $\tanh(\cdot)$ activation function allows the neural network to solve the reverse problem more efficiently. It seems that the geometry of the $\tanh(\cdot)$ activation function closely resembles the rarefaction section for each physical quantity, suggesting that activation function geometry may play a roll in achieving smaller approximation error. In future work, we will investigate the geometric effects activation functions have on PINNs when solving hydrodynamic shock-tube problems.

ρ_L	u_L	p_L	ρ_R	u_R	p_R	$x \in \Omega$	$x \in \Omega_e$	x^*
0.125	0.0	0.1	1.0	0.0	1.0	$(0, 1)$	$(-2.625, 2.5)$	0.5

Table VII – Initial conditions for the reverse Sod shock-tube problem

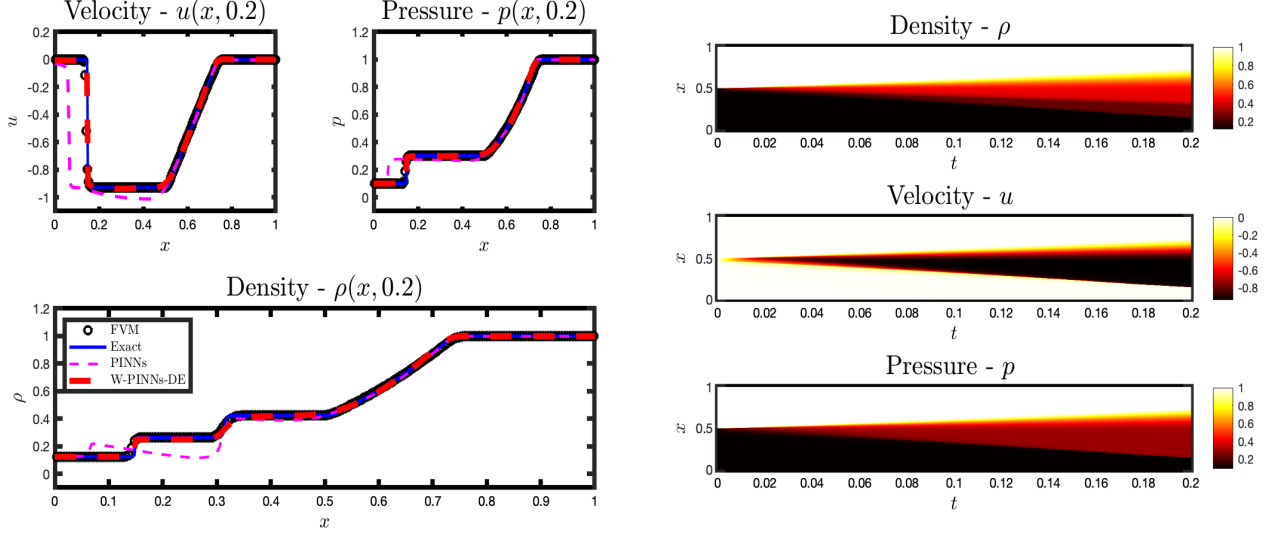


Figure 6 – The reverse Sod shock-tube problem with sampling, $N_f = 10, 500$, $N_{IC} = 1000$

Method	$\frac{\ \rho_{approx} - \rho_{exact}\ _2}{\ \rho_{exact}\ _2}$	$\frac{\ u_{approx} - u_{exact}\ _2}{\ u_{exact}\ _2}$	$\frac{\ p_{approx} - p_{exact}\ _2}{\ p_{exact}\ _2}$
FVM	$9.4e - 03$	$3.0e - 02$	$1.0e - 02$
PINNs	$3.7e - 02$	$2.1e - 01$	$4.1e - 02$
W-PINNs-DE	$8.5e - 03$	$3.0e - 02$	$6.6e - 03$

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

D. Double Expansion Fan Problem

In 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. Instead, each physical quantity develops two, oppositely traveling, expansion fans. Figure 7 demonstrates that each physical quantity evolves to have two expansion fans.

In figure 7, we compare W-PINNs-DE to PINNs and the FVM, as done in previous sections. W-PINNs-DE solves this problem more accurately than PINNs and the FVM, as shown in table X. Remarkably, the original PINNs method shows no difficulty to solve this problem, as it obtains less than 0.6% relative L_2 error for each physical quantity. The neural network does not have difficulty approximating the solution since sharp jumps are not present as t approaches 0.2. However, PINNs dissipate the solution near the smoothed corners towards the boundary, resulting in the highest relative L_2 error in comparison to the other methods. On the other hand, the FVM slightly disperses the solution at $x = 0.5$. At $x = 0.5$, the two fans depart, hence, the solution is prone to higher error at $(0.5, t)$, for $t > 0$. W-PINNs-DE proves to have minimal dispersion and dissipation, resulting in having the smallest relative L_2 error.

ρ_L	u_L	p_L	ρ_R	u_R	p_R	$x \in \Omega$	$x \in \Omega_e$	x^*
1.0	-1.4	0.1	1.0	1.0	0.4	(0, 1)	(-2.625, 2.5)	0.5

Table IX – Initial conditions for the double expansion fan problem

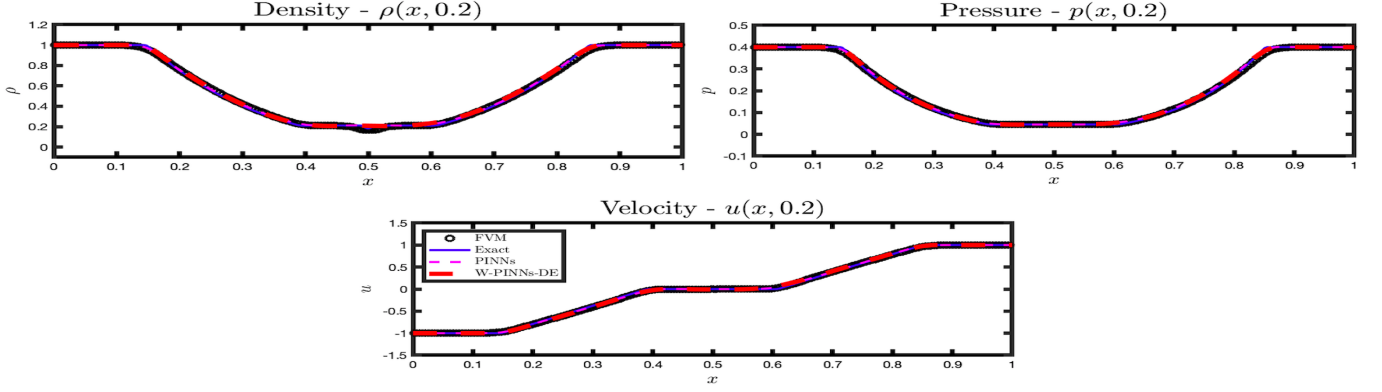
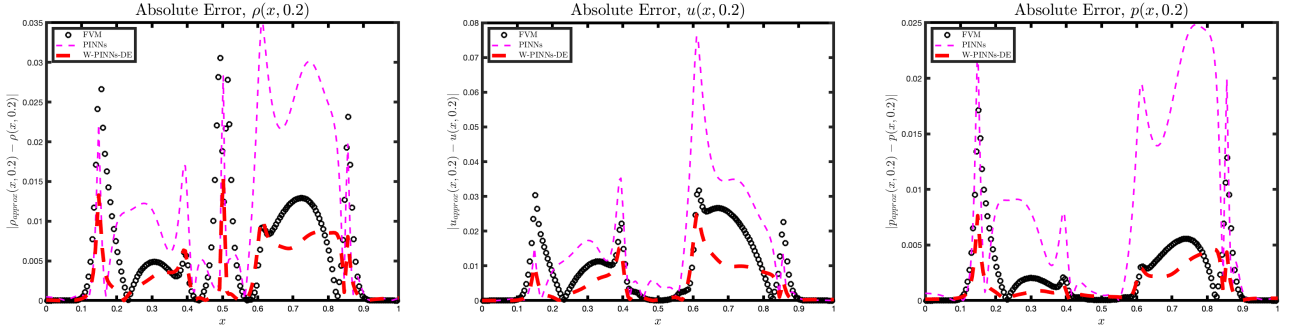
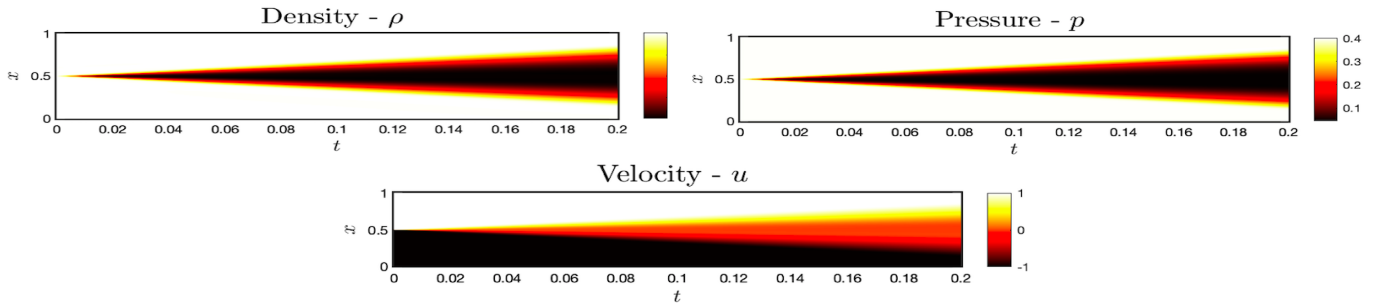
Figure 7 – Solution at $t = 0.2$ for the double expansion fan problem, with sampled points, $N_f = 10,500$, $N_{IC} = 1000$ Figure 8 – Absolute error at $t = 0.2$ 

Figure 9 – Top view of solution to the double expansion fan problem

Method	$\ \rho_{approx} - \rho_{exact}\ _2$	$\ u_{approx} - u_{exact}\ _2$	$\ p_{approx} - p_{exact}\ _2$
	$\frac{\ \rho_{approx} - \rho_{exact}\ _2}{\ \rho_{exact}\ _2}$	$\frac{\ u_{approx} - u_{exact}\ _2}{\ u_{exact}\ _2}$	$\frac{\ p_{approx} - p_{exact}\ _2}{\ p_{exact}\ _2}$
FVM	$2.7e-03$	$2.1e-03$	$1.9e-03$
PINNs	$4.5e-03$	$3.2e-03$	$5.2e-03$
W-PINNs-DE	$1.4e-03$	$1.6e-02$	$1.3e-03$

Table X – Relative L_2 errors for double expansion fan problem

E. High-Speed Flow Problem I

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. Although the modified reverse Sod shock-tube problem has an additional discontinuous initial state, the relative L_2 errors are smaller than that of the reverse Sod shock-tube problem.

W-PINNs-DE and the FVM create some amount of artificial dissipation at the contact discontinuity, whereas PINNs disperses the solution, as shown in figure 10 and 11. PINNs have a higher relative L_2 error for each physical quantity since it disperses the solution in several regions. Although W-PINNs-DE has the smallest relative L_2 error, in figure 10 and 11, we see there is slight dissipation at the shock and contact discontinuity.

ρ_L	u_L	p_L	ρ_R	u_R	p_R	$x \in \Omega$	$x \in \Omega_e$	x^*
0.125	0.0	0.1	1.0	0.75	1.0	$(-0.5, 1.5)$	$(-2.625, 2.5)$	0.3

Table XI – Initial conditions for the high-speed flow problem I

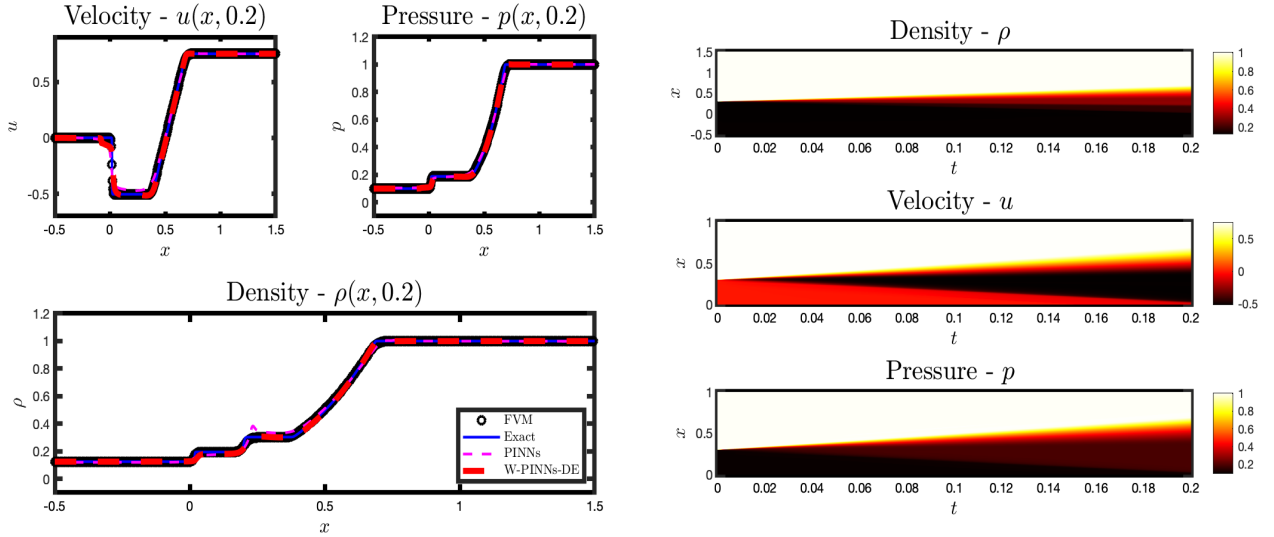


Figure 10 – High-Speed Flow Problem I, with sampled points, $N_f = 10,500$, $N_{IC} = 1000$

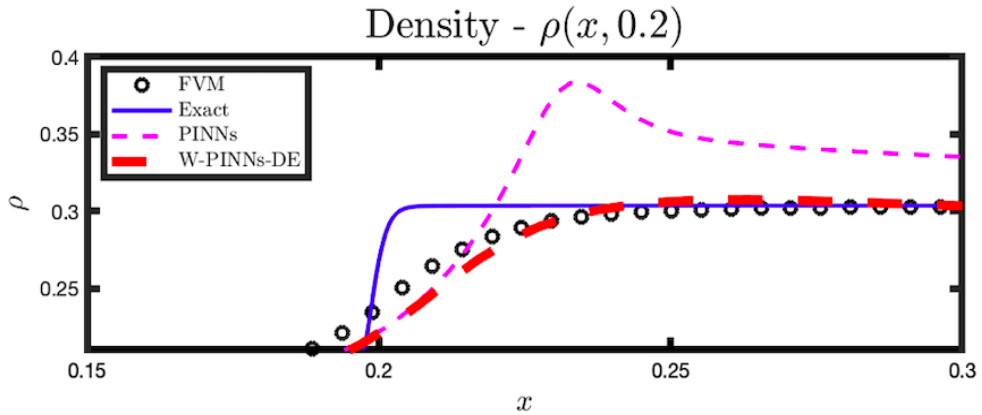


Figure 11 – Close up of the contact discontinuity of the density from figure 10

Method	$\frac{\ \rho_{approx} - \rho_{exact}\ _2}{\ \rho_{exact}\ _2}$	$\frac{\ u_{approx} - u_{exact}\ _2}{\ u_{exact}\ _2}$	$\frac{\ p_{approx} - p_{exact}\ _2}{\ p_{exact}\ _2}$
	$\ \rho_{exact}\ _2$	$\ u_{exact}\ _2$	$\ p_{exact}\ _2$
FVM	$7.7e-03$	$3.5e-02$	$8.1e-03$
PINNs	$2.2e-02$	$6.6e-02$	$1.6e-02$
W-PINNs-DE	$6.7e-03$	$3.2e-02$	$5.0e-03$

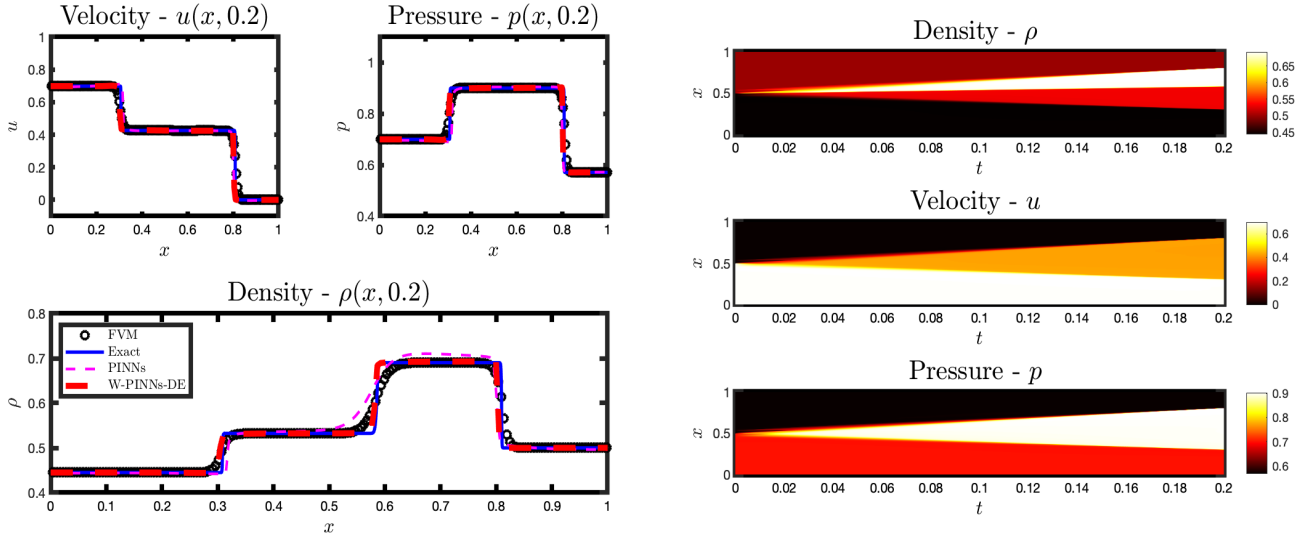
Table XII – Relative L_2 errors for High-Speed Flow Problem I

F. High Speed Flow Problem II

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 12, W-PINNs-DE captures each rarefaction fan and shock with minimal dissipation and dispersion. The FVM creates slight dissipation in the shock regions of the density, where as the original PINNs method dissipates the solution at the bottom of the shock, but disperses the solution at the top. As reported in table XIV, W-PINNs-DE solves for the density and pressure terms with less than 0.8% error, respectively, and the velocity term with less than 2%. Similarly to the first high-speed flow problem, we are surprised to see how well each method performs considering the complexity of the initial-state in comparison to each test problem presented so far.

ρ_L	u_L	p_L	ρ_R	u_R	p_R	$x \in \Omega$	$x \in \Omega_e$	x^*
0.445	0.698	0.7	0.5	0.0	0.571	$(0, 1)$	$(-2.625, 3.125)$	0.5

Table XIII – Initial conditions for the high-speed flow problem II

Figure 12 – High-Speed Flow Problem II, with sampled points, $N_f = 11,000$, $N_{IC} = 1000$

Method	$\frac{\ \rho_{approx} - \rho_{exact}\ _2}{\ \rho_{exact}\ _2}$	$\frac{\ u_{approx} - u_{exact}\ _2}{\ u_{exact}\ _2}$	$\frac{\ p_{approx} - p_{exact}\ _2}{\ p_{exact}\ _2}$
	$\ \rho_{exact}\ _2$	$\ u_{exact}\ _2$	$\ p_{exact}\ _2$
FVM	$8.1e-03$	$3.5e-02$	$3.5e-03$
PINNs	$8.7e-02$	$6.6e-02$	$4.4e-02$
W-PINNs-DE	$7.7e-03$	$1.4e-02$	$6.9e-03$

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

V. DISCUSSION

In this study, we propose W-PINNs-DE and test it on a number of hydrodynamic shock-tube test problems. For each test problem, we compared W-PINNs-DE to the FVM with flux limiting [14] and the original PINNs method [7]. Using W-PINNs-DE provides neural networks with a framework to solve for a general class of discontinuous solutions of the compressible Euler equations. It does so by having an ability to capture physical phenomena such as shocks, contact discontinuities, and rarefaction fans, with an impressive degree of accuracy. By introducing a weighted loss function and domain extension, we've shown that $G_f(\theta)$ decreases at a slower rate than $G_{IC}(\theta)$, resulting in accurate solutions to hydrodynamic shock-tube problems. To demonstrate the necessity of (8), in figure 3, we plotted the $G(\theta)$, $G_f(\theta)$, and $G_{IC}(\theta)$ versus iteration for PINNs and W-PINNs-DE, when solving the Sod shock-tube problem, and presented their solutions in section III.B. $G_f(\theta)$ and $G_{IC}(\theta)$ satisfied (8) for each test problem solved by W-PINNs-DE, whereas PINNs did not. We have shown through each test case that $G_f(\theta)$ and $G_{IC}(\theta)$ must satisfy (8) in order to solve hydrodynamic shock-tube problems using neural networks.

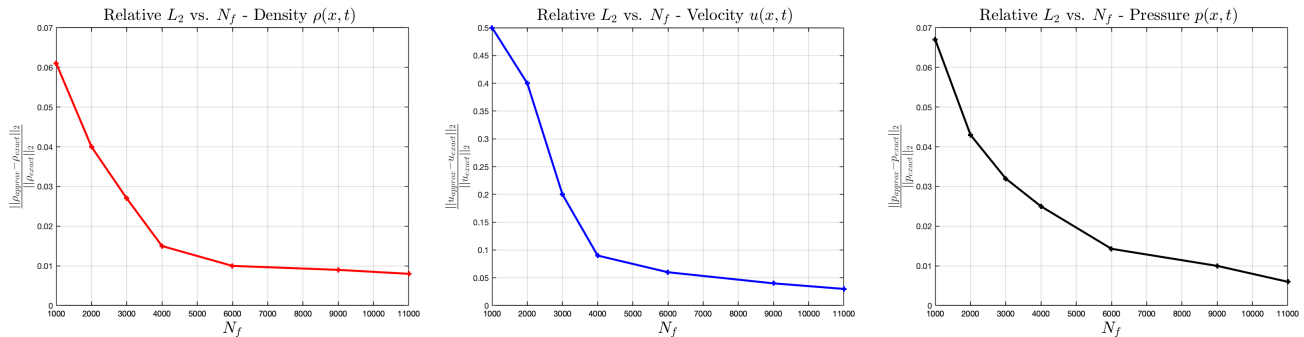


Figure 13 – Relative L_2 errors vs. N_f samples

Although W-PINNs-DE proves to be a competitive numerical tool for solving conservation laws, each test problem demonstrated higher numerical error for the velocity than the density and pressure. In figures 6-10, the W-PINNs-DE velocity typically experienced dissipation near shock corners, whereas W-PINNs-DE captured the rarefaction fan exceptionally. Though this error is acceptable, it is a goal to find as accurate solutions as possible. Hence, understanding this phenomenon is an essential task. 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.

The high-speed flow problems presented in sections III.E and III.F had three discontinuous initial states. Both problems developed more physics in their solutions than the other shock-tube problems. Yet, despite the initial state and final solutions complexity, W-PINNs-DE approximates the solution to each high-speed flow problem with smaller relative L_2 error than W-PINNs-DE solutions to the Sod shock-tube problems. To get an idea of how W-PINNs-DE error decays as N_f increases, in figure 13, we plot the relative L_2 errors versus N_f for the high speed flow problem II. It is to be expected that as N_f increases, the relative L_2 error for each physical quantity will decrease. As the number of samples increases, the neural network resolves the solution accurately. However, figure 13 reveals for $N_f \geq 9000$, the relative L_2 errors reach a saturation point. In future work, we will investigate rigorously the convergence of W-PINNs-DE.

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. In addition to solving hydrodynamic shock-tube problems, W-PINNs-DE can solve isothermal Euler equations and advection-dominated compressible Navier-Stokes equations.

A. 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 C++ and 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>

B. 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.

VI. 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, 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, Wood, M. A. (2020). Thermodynamically consistent physics-informed neural networks for hyperbolic systems. doi:<https://arxiv.org/abs/2012.05343>
- [17] Kim, J, Kim, A, Lee, S. "Artificial Neural Network-Based Automated Crack Detection and Analysis for the Inspection of Concrete Structures." Applied Sciences, vol. 10, no. 22, 2020, p. 8105., doi:10.3390/app10228105.
- [18] Cazzanti, L, Khan, M, Cerrina, F. "Parameter Extraction with Neural Networks." Metrology, Inspection, and Process Control for Microlithography XII, 1998, doi:10.1117/12.308780.

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 (although referred to as PINNs in each figure), and the blue line is the exact solution of the two quantities. In figures 14-19, we see that W-PINNs-DE satisfied the conservation of energy, verifying once more the accuracy of the method.

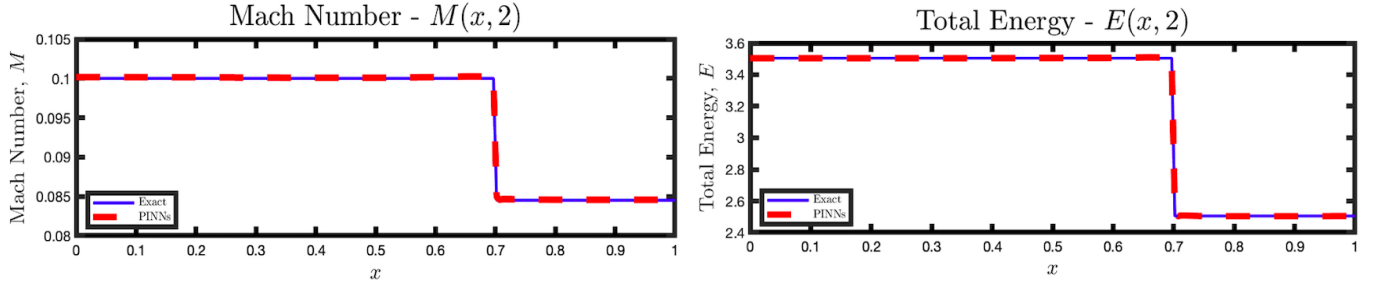


Figure 14 – Mach number and total energy at $t = 2.0$ for the single contact discontinuity problem

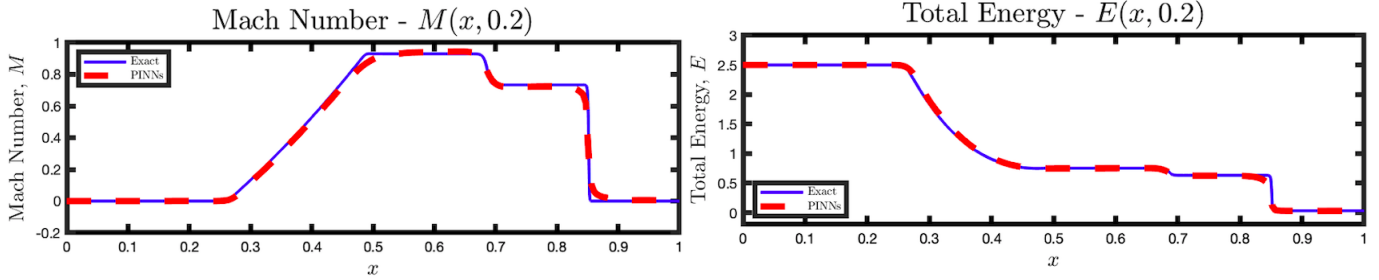


Figure 15 – Mach number and total energy at $t = 0.2$ for the Sod shock-tube problem

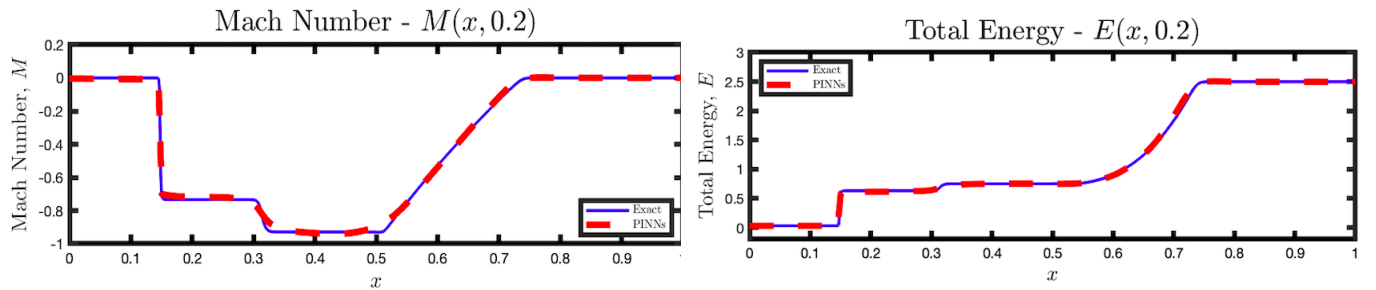


Figure 16 – Mach number and total energy at $t = 0.2$ for the reverse Sod shock-tube problem

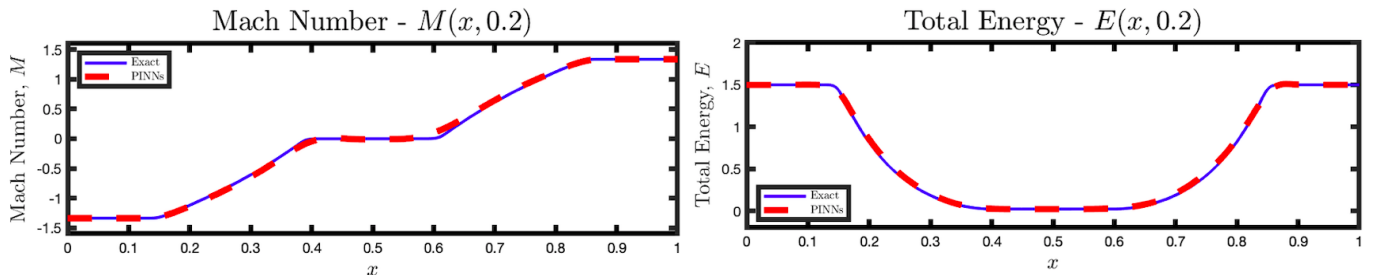


Figure 17 – Mach number and total energy at $t = 0.2$ for the double expansion fan problem

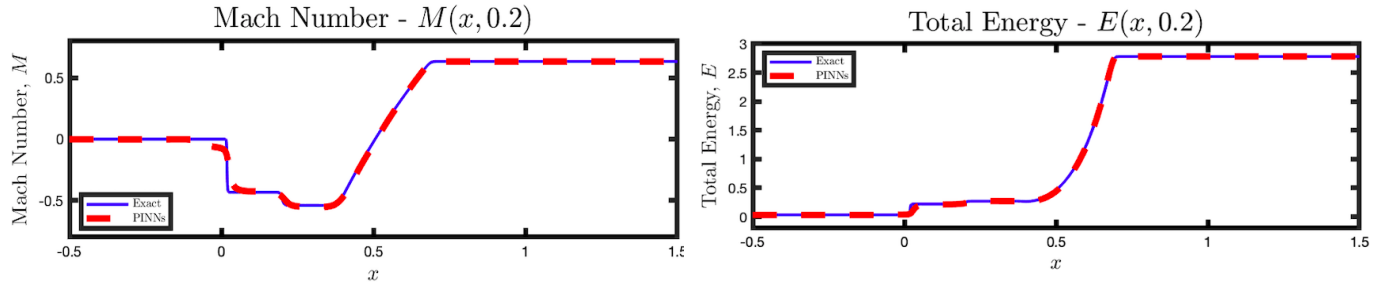


Figure 18 – Mach number and total energy at $t = 0.2$ for the high-speed flow problem I

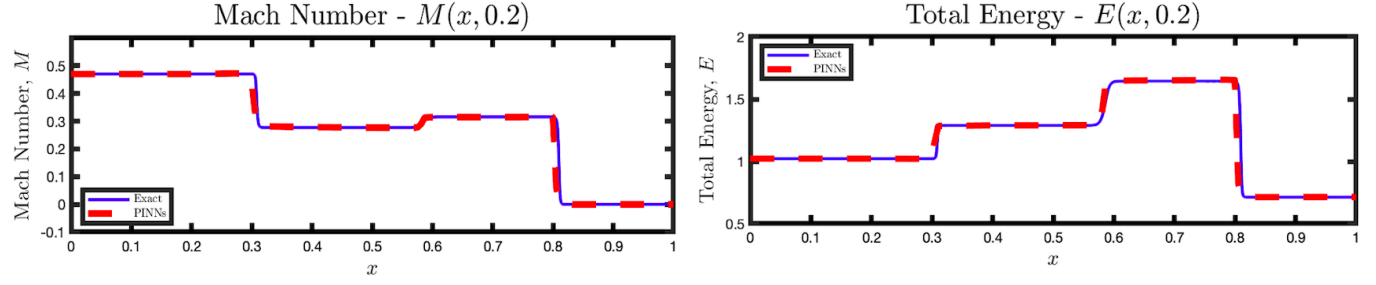


Figure 19 – Mach number and total energy at $t = 0.2$ for the high-speed flow problem II