



Discontinuity Computing Using Physics-Informed Neural Networks

Li Liu¹ · Shengping Liu¹ · Hui Xie¹ · Fansheng Xiong² · Tengchao Yu¹ · Mengjuan Xiao¹ · Lufeng Liu¹ · Heng Yong^{1,2} 

Received: 5 February 2023 / Revised: 28 October 2023 / Accepted: 1 November 2023 /
Published online: 13 December 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Simulating discontinuities has been a long-standing challenge, especially when dealing with shock waves characterized by strong nonlinear features. Despite their promise, the recently developed physics-informed neural networks (PINNs) have not yet fully demonstrated their effectiveness in handling discontinuities when compared to traditional shock-capturing methods. In this study, we reveal a paradoxical phenomenon during the training of PINNs when computing problems with strong nonlinear discontinuities. To address this issue and enhance the PINNs' ability to capture shocks, we propose PINNs-WE (Physics-Informed Neural Networks with Equation Weight) method by introducing three novel strategies. Firstly, we attenuate the neural network's expression locally at 'transition points' within the shock waves by introducing a physics-dependent weight into the governing equations. Consequently, the neural network will concentrate on training the smoother parts of the solutions. As a result, due to the compressible property, sharp discontinuities emerge, with transition points being compressed into well-trained smooth regions akin to passive particles. Secondly, we also introduce the Rankine–Hugoniot (RH) relation, which is equivalent to the weak form of the conservation laws near the discontinuity, in order to improve the shock-capturing performance. Lastly, we construct a global physical conservation constraint to enhance the conservation properties of PINNs which is key to resolve the right position of the discontinuity. To illustrate the impact of our novel approach, we investigate the behavior of the one-dimensional Burgers' equation, as well as the one- and two-dimensional Euler equations. In our numerical experiments, we compare our proposed PINNs-WE method with a traditional high-order weighted essential non-oscillatory (WENO) approach. The results of our study highlight the significant enhancement in discontinuity computing by the PINNs-WE method when compared to traditional PINNs.

Keywords Physics-informed neural networks · Shock capturing · Compressible flow · Euler equations · Discontinuity calculation

✉ Heng Yong
Yong_Heng@iapcm.ac.cn

¹ Institute of Applied Physics and Computational Mathematics, Beijing 100094, China

² Yanqi Lake Beijing Institute of Mathematical Sciences and Applications, Beijing 101408, China

1 Introduction

Efficiently capturing shock waves and other discontinuities is crucial for solving hyperbolic equations. Von Neumann and Richtmeyer pioneered shock-capturing work in 1950 [1], introducing artificial viscosity into a staggered Lagrangian scheme for compressible flow simulations. Today, advanced techniques such as essential non-oscillatory (ENO) [2], weighted ENO (WENO) [3], and discontinuous Galerkin methods [4] offer high-order accuracy in simulating problems involving shock waves. For more in-depth information on the development of shock-capturing methods, you can refer to [5, 6].

Due to the rapid development of machine learning and neural networks (NNs), they have been extensively employed to solve partial differential equations (PDEs) [7–11]. Among these applications, physics-informed neural networks (PINNs) have received significant research attention. PINNs encode PDEs or other model equations as one of their penalty components, making them versatile tools across various fields [12]. In the realm of hyperbolic equations, Mao et al. [13] studied one- (1D) and two-dimensional (2D) Euler equations with shock waves and used clustered training samples around high gradient area to improve the solution accuracy in these regions while preventing error propagation to the entire domain. In another study, Patel et al. [14] introduced a PINN capable of discovering thermodynamically consistent equations that ensure hyperbolicity. This approach is particularly relevant for inverse problems in shock hydrodynamics. Furthermore, Jagtap et al. [15] proposed conservative PINNs that partition the computational domain into smaller subdomains, each employing distinct NNs to tackle Burgers' equation and Euler equations. Additionally, Jagtap et al. [16] delved into inverse problems of supersonic flows.

The aforementioned studies showcase the effectiveness of PINNs in addressing inverse problems involving prior knowledge of flow structure development, such as density gradients [17]. However, when it comes to studying forward problems, the original PINNs have shown limited applicability, primarily restricted to simple problems like tracking moving shock waves. In contrast, Patel et al. [14], based on traditional shock-capturing methods, devised a mesh-based control-volume PINN and incorporated entropy and total-variation-diminishing conditions into the neural network. Additionally, Papados [18] extends the computational domain for simulating the shock-tube problem, yielding remarkable results without the introduction of non-physical viscosity terms into the equations. However, their open-source code suggests that this method is not very stable in the cases under consideration, and good results may only be achieved with fixed hyperparameters under a fixed number of training steps. Nonetheless, this work has demonstrated the potential of using the PINNs method for solving shock wave problems.

We are interested in applying PINNs to the computation of discontinuities, particularly in problems involving the generation of nonlinear strong discontinuities, such as shock waves in hyperbolic equations. Mathematically, these discontinuities possess zero thickness and infinite gradients, rendering them beyond the description of strong form PDEs. Instead, their behavior can be controlled by physical laws or weak form equations. Furthermore, to the best of our knowledge, there is no existing theory that can guarantee neural networks (NNs) accurately approximate C^0 discontinuous functions. Consequently, when residual points fall within a shock region, significant equation losses are incurred due to their steep gradients. These points, located in high-gradient regions where the shock is expected to occur, are referred to as 'transition points' [19]. While an NN may prioritize handling transition points as they contribute the most to the loss. However, an NN cannot increase the gradient to decrease the thickness of the shock because of the aforementioned reason. As a result, transition points

fall into a paradoxical state - whether the gradient increases or decreases, the training loss inevitably rises. What's more concerning is that transition points not only impact the total loss but also influence convergence in smooth regions. We elucidate this phenomenon in Sect. 2.2 through a test involving the Burgers' equation.

To enhance the shock-capturing capabilities of PINNs, this paper presents three key contributions. Firstly, in order to break the paradoxical status at transition points and enable PINNs to represent strong nonlinear discontinuities effectively, we introduce a 'retreat to advance' strategy. This strategy weakens the neural network's expressions in regions of strong compression and large gradients by introducing a physical pointwise equation weight, enabling the network to focus on training in other regions. As a result, depending on the physical compression mechanisms from the well-trained smooth regions, strong discontinuity solutions automatically generate.

Secondly, for the effective control of shock wave solutions and to prevent underdetermined problems, we incorporate the Rankine–Hugoniot (RH) relation, which is equivalent to the weak form of the conservation laws, as constraints in the vicinity of the shock waves. Also, we implement limiters to identify the appearance of shock waves in this part.

Thirdly, in the computation of nonlinear hyperbolic equations, such as the Euler equations, preserving physical conservation is of paramount importance. For instance, it directly impacts the accuracy of shock wave positions. To address this, we introduce a global physical conservation constraint within the new framework.

The remainder of this paper is organized as follows. In Sect. 2, we first analyze the failure of classical PINNs in solving shock waves with the non-dissipative Burgers' equation. Then, in Sect. 3, we detail our proposed PINNs-WE method. Next, various 1D and 2D forward examples are studied to demonstrate the effectiveness of the proposed method in Sect. 4. Finally, conclusions are drawn in Sect. 5. In addition, we also discuss the omissible boundary conditions in PINNs in Appendix A and verify original PINNs in solving problems with linear or weak discontinuities in Appendix B.

2 Classical PINNs and Problem Analysis

2.1 PINNs for Conservative Hyperbolic PDE

We consider the following conservative hyperbolic PDE

$$\frac{\partial \mathbf{U}(\mathbf{x})}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0, \quad \mathbf{x} = (t, x_1, x_2, \dots) \in \Omega, \quad (1)$$

with the initial and boundary conditions (IBCs):

$$\text{IBCs}(\mathbf{U}, \mathbf{x}) = 0 \quad \text{on} \quad \partial\Omega, \quad (2)$$

and we treat the initial condition in the same way with Dirichlet boundary condition.

The classical PINNs typically consist of two parts. The first part is a neural network $\hat{\mathbf{U}}(\mathbf{x}; \boldsymbol{\theta})$ used to approximate the relation of $\mathbf{U}(\mathbf{x})$ with trainable parameters $\boldsymbol{\theta}$. The second part is informed by the governing equations as well as the initial and boundary conditions, which are used to train the network. The calculation of $\partial/\partial t$ and $\nabla \cdot$ in the PDE are carried out through automatic derivative evaluation. More details information about PINNs for convection PDE can be found in [13, 18].

The loss function used to train $\hat{\mathbf{U}}(\mathbf{x}; \boldsymbol{\theta})$ comprised at least two components that define the problem. One component is controlled by the equations, and the other one is given by the

IBCs of the problem,

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \omega_{\text{IBCs}} \mathcal{L}_{\text{IBCs}}. \tag{3}$$

To define the loss, we choose a set of residual points inside the domain Ω and another set of points on $\partial\Omega$ as \mathcal{S}_{PDE} and $\mathcal{S}_{\text{IBCs}}$, respectively. Then

$$\mathcal{L} = \frac{1}{|\mathcal{S}_{\text{PDE}}|} \sum_{\mathbf{x}_i \in \mathcal{S}_{\text{PDE}}} \mathbf{G}_i^2 + \omega_{\text{IBCs}} \frac{1}{|\mathcal{S}_{\text{IBCs}}|} \sum_{\mathbf{x}_i \in \mathcal{S}_{\text{IBCs}}} (\hat{\mathbf{U}}_{e,i} - \mathbf{U}_{e0,i})^2, \tag{4}$$

where $\mathbf{G}_i := \partial_t \hat{\mathbf{U}}(\mathbf{x}_i) + \nabla \cdot \mathbf{F}(\hat{\mathbf{U}}(\mathbf{x}_i))$, $\mathbf{G}_i = 0$ is the governing equations at residual point $\mathbf{x}_i \in \mathcal{S}_{\text{PDE}}$ and $\mathbf{U}_{e0,i}$ represents the given IBCs at residual point $\mathbf{x}_i \in \mathcal{S}_{\text{IBCs}}$. ω_{IBCs} is the weight to adjust the confinement strength of IBCs [16, 18, 20]. Typically, more weight is assigned to points located on initial and boundaries.

In each term of a loss function, it is common to average the residual across all residual points to obtain the total training loss. Averaging is often suitable and convenient for problems with smooth solutions. However, when a nonlinear discontinuity forms under compression, the gradient becomes theoretically infinite and cannot be directly described by differential equations. As a result, transition points defined within the discontinuity may introduce significant errors and loss of function. We will illustrate this in the following example.

2.2 Analysis of Transition Points Based on Inviscid Burgers' Equation

We demonstrate in Appendix B that classical PINNs can sharply capture linear or weak discontinuities without the need of introducing numerical dissipation to maintain stability. However, when it comes to solving nonlinear strong discontinuities, particularly compression discontinuities like shock waves, PINNs encounter significant challenges. To illustrate this phenomenon in the context of PINNs solving problems with compression discontinuities, we consider an inviscid Burgers' equation problem with the following equation and IBCs as

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial(u^2/2)}{\partial x} &= 0, \quad x \in [0, 2], \quad t \in [0, 1], \\ u(0, x) &= -\sin(\pi(x - 1)), \\ u(t, 0) = u(t, 2) &= 0. \end{aligned} \tag{5}$$

In this problem, the initial condition is smooth. However, as the initial velocities on both sides are directed towards the center, a strong discontinuous solution will form in the center after finite time with the compression from both the left and right sides.

We solve this problem using the original PINNs method, which is referred to as 'PINNs' in this paper and was introduced in the previous section. Figure 1 presents the loss history, the variable u , and the residual at $t = 1$ for different training epochs (1000, 3000, 5000, 8000, and 11500). It is evident that the PINNs initially tend to achieve a global smooth solution to satisfy the initial boundary conditions (checkpoint 1). Subsequently, the training focuses on reducing the residual to approach the exact solution of the problem. However, when the transition points reach relatively high gradients (checkpoint 2), they carry nearly all of the function loss \mathcal{L}_{PDE} (Fig. 1c). The training enters a paradoxical state - irrespective of an increase (checkpoint 3) or decrease (checkpoint 4) in the gradient, the training loss always increases. This is because these points cannot be directly controlled by the strong form of the PDE. So a decrease in the gradient will cause the PINNs solution to deviate from the exact solution, while an increase will also result in an increased residual.

As shown in the loss history (Fig. 1a), after 3000 epochs (checkpoint 2), the training encounters difficulties with these transition points and struggles to effectively reduce the total loss. More importantly, as a global method, the total loss also influences the error magnitude in other regions. This stands in contrast to classical finite-volume or finite-difference methods. In those classical methods, when a cell is located within a discontinuous region, the scheme's order automatically reduces to no more than second-order, and significant dissipation is introduced into the cell to achieve a non-oscillatory result. However, the locally large error does not impact the accuracy order of other cells outside the discontinuity.

3 PINNs-WE Framework

We assume that the flow field is continuous except for finite discontinuities. To effectively capture strong nonlinear discontinuities using PINNs, we introduce a new weighted equation framework. The basic idea is to separate the shock waves from other regions and solve them with different constraints. As depicted in Fig. 2, the new framework consists of three key components.

The first part is a local strong form PDE constraint \mathcal{L}_{PDE} for simulating the smooth regions of the computational domain, which also includes linear or weak discontinuities, as explained in Appendix B.

The second part is Rankine–Hugoniot relation constraint \mathcal{L}_{RH} , which equivalent to local weak form of the conservation laws across the shock waves.

The last part is a global conservation component $\mathcal{L}_{\text{CONS}}$ that enforces total conservation.

The algorithm details for solving 2D Euler equations are presented in Fig. 3, while the sample domains for each part are illustrated in Fig. 4.

3.1 Local Strong form PDE Constraint

Based on the aforementioned analysis of the Burgers' equation, the training of PINNs can become challenging due to the paradoxical status of transition points.

Therefore, the first and most important part of our method is to resolve this paradoxical status by introducing a weighted-equation (WE) method. The core idea behind this method is to reduce the impact of points in high-gradient regions by introducing a local positive weight, denoted as λ_1 , into the governing equations.

The effectiveness of this method is based on the fact that strong nonlinear discontinuities are formed by the convergence of characteristic lines. By reducing the influence of transition points through the adjustment of λ_1 , the neural network (NN) will prioritize training in the smooth regions and achieve high accuracy in those areas. Consequently, the transition points, which are inherently influenced by the compression properties of strong discontinuities, will effectively be compressed into a smooth region under the weak form equation constraint introduced in the next part. As a result, a sharp and precise discontinuity will emerge during training.

For a general conservative equation (Eq. 1), if we define

$$\mathbf{G} = \frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}, \quad (6)$$

the weighted equation is defined as follows:

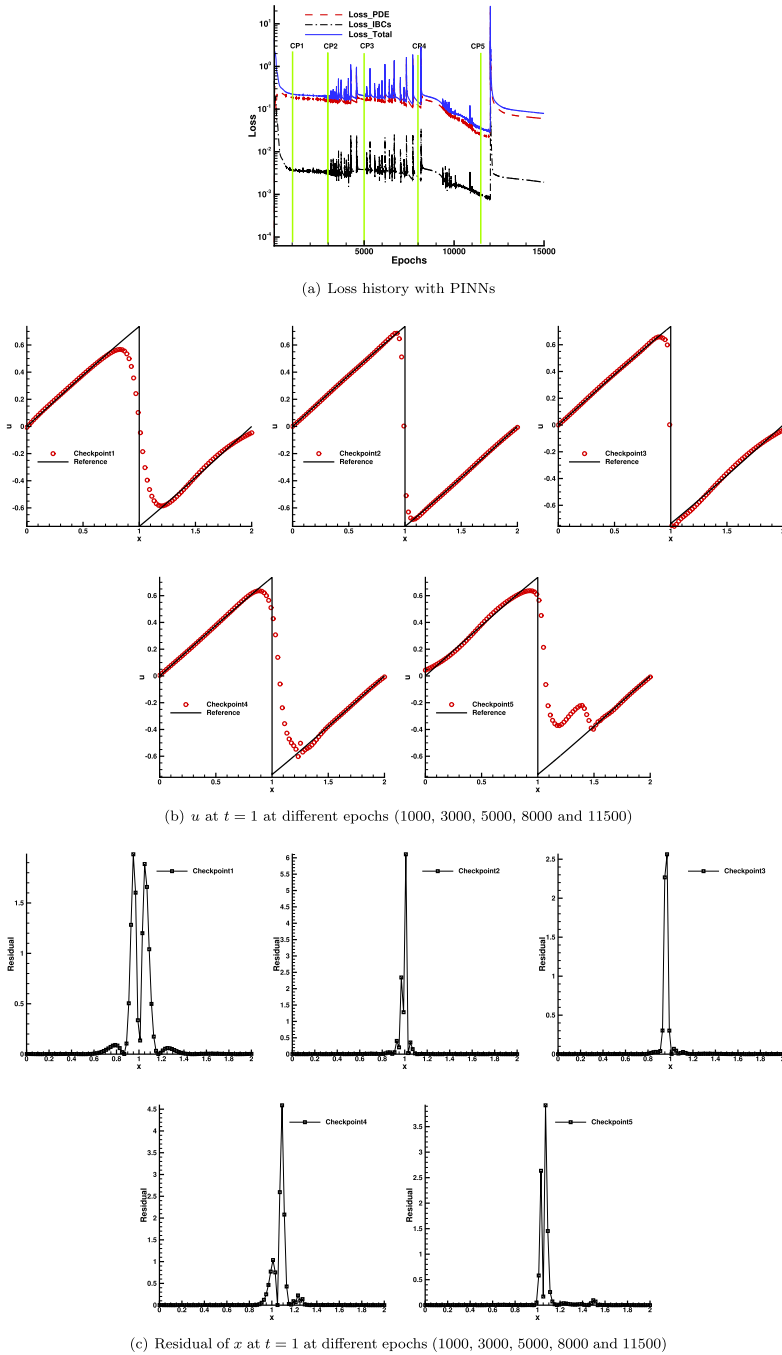


Fig. 1 Results of the Burgers equation with PINNs. The NN has 4 hidden layers and 30 neurons per layer. PDE residual points are distributed uniformly on a 100×100 grid in the $X \times T$ space, and the IBCs points are set on a uniform grid of 100. The optimizer used is Adam with a learning rate of 0.001. The reference result is provided by WENO-Z on a refined mesh with 10,000 spatial grids

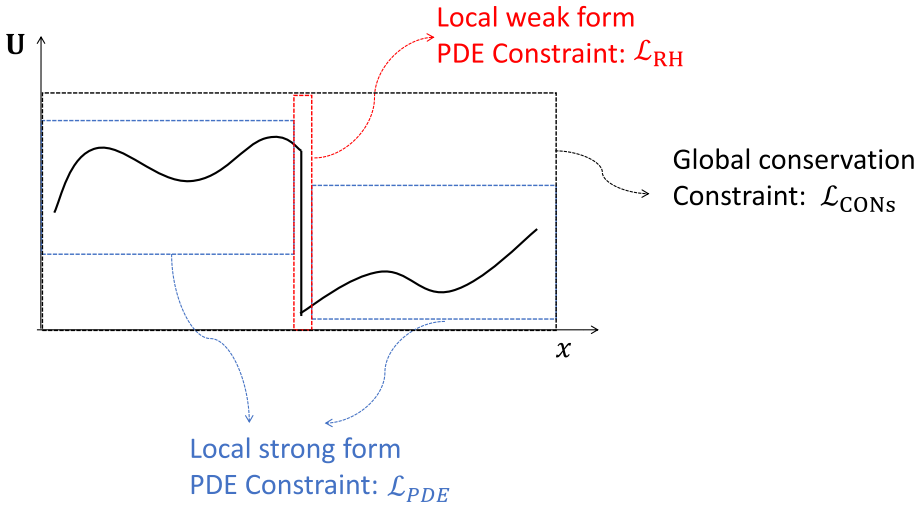


Fig. 2 The basic idea of PINNs-WE framework in solving problems with strong nonlinear discontinuities

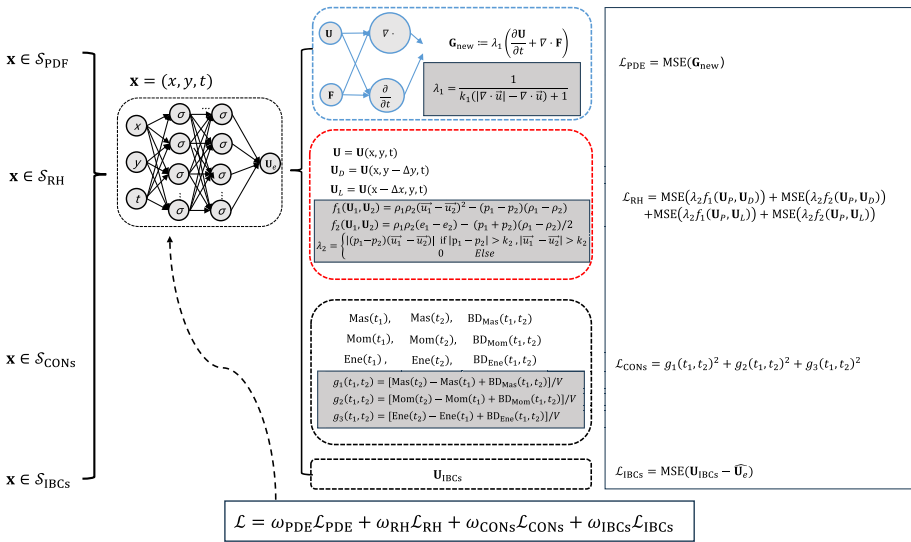


Fig. 3 Architecture of PINNs-WE for solving 2D Euler equations

$$\mathbf{G}_{\text{new}} = \lambda_1 \left(\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} \right). \tag{7}$$

It is evident that $\mathbf{G}_{\text{new}} = 0$ shares the same solutions as $\mathbf{G} = 0$ if λ_1 remains positive. Moreover, we can adjust the NN’s expression at different points through the design of the weight λ_1 . Correspondingly, the new function loss is defined as:

$$\mathcal{L}_{\text{PDE}} = \text{MSE}(\mathbf{G}_{\text{new}}) = \frac{1}{|\mathcal{S}_{\text{PDE}}|} \sum_{\mathbf{x}_i \in \mathcal{S}_{\text{PDE}}} \mathbf{G}_{\text{new},i}^2. \tag{8}$$

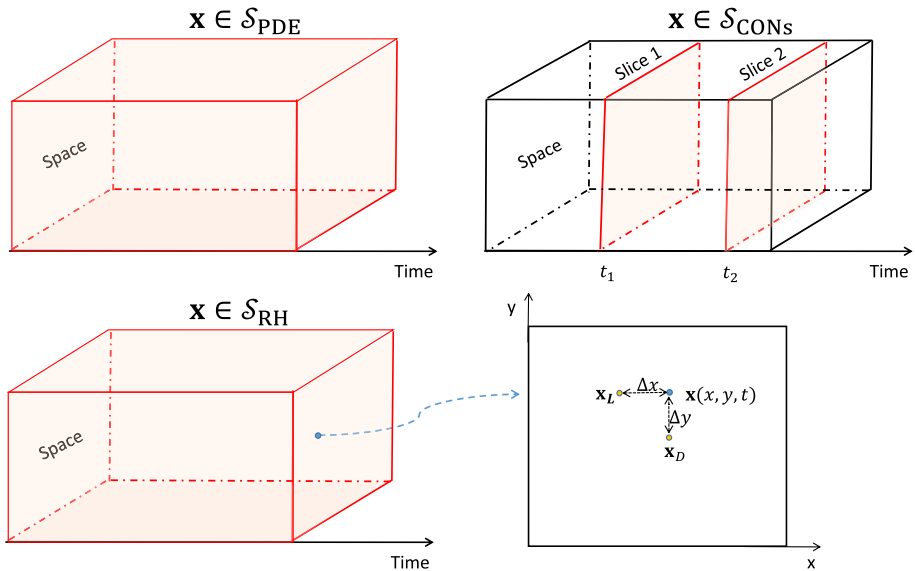


Fig. 4 The sample space of residual points in different loss parts

We would like to note that McClenny and Braga-Neto [21] have also introduced a trainable weight into the loss function to focus the NN on improving approximation in difficult-to-train regions by increasing the loss weights on them. However, based on our analysis, we need to decrease the weight near a physical discontinuity. We believe that this process cannot be effectively controlled by strong form PDEs residual.

Another relative work is proposed by Wang et al. [22], they identify and analyze a fundamental failure mode related to numerical stiffness causing imbalanced gradients during model training. To address this, they introduce a learning rate annealing algorithm that uses gradient statistics to balance the interplay of various terms in composite loss functions. However, the distinction lies in the fact that the computation of shock waves results in an imbalance in spatial derivatives due to physical factors and the non-validity of the strong form PDEs.

We are explaining here that the reduction of sampling points near discontinuities or the extension of the computational domain can also be considered as a method of attenuating the attention on regions close to the discontinuities. However, such methods may, on one hand, reduce the resolution of shock waves and, simultaneously, make it challenging to achieve sufficient weakening strength.

Inspired by the seminal work [1] on artificial viscosity, we define a physics-dependent weight as follows:

$$\lambda_1 = \frac{1}{k_1(|\nabla \cdot \vec{u}| - \nabla \cdot \vec{u}) + 1}, \tag{9}$$

Here, \vec{u} represents the velocity field, and k_1 is a factor used to adjust λ_1 . This factor may vary from case to case; Based on the tests, we suggest using a global value of $k_1 = 0.2$ for this research.

The construction of this weight is based on the fact that velocity divergence $\nabla \cdot \vec{u}$ becomes negative when the field is compressed, and $\nabla \cdot \vec{u} \rightarrow -\infty$ when a shock appears. It's important to highlight that we only utilize velocity divergence to detect shocks and apply λ_1 to the

equations without introducing any numerical dissipation. Since λ_1 is a positive constant, it does not affect the exact solution of the equations.

3.2 Local Weak form of the Conservation Laws Constraint

Since strong form PDEs cannot describe strong discontinuous solutions, the PDE constraints near discontinuities can lead to significant loss errors. Therefore, in the first part, we reduce the weight of the equations near the shock wave regions to eliminate the incorrect constraints in PINNs. However, until now, there have been no appropriate constraints over the shock regions, making the problem underconstrained.

Although strong form solutions do not exist across the shock waves, the physical conservation laws still hold, meaning that the physical flux crossing the shock waves must be conservative. This leads us to the Rankine–Hugoniot (RH) relation of the conservation laws (1):

$$s \cdot \llbracket \mathbf{U} \rrbracket = \llbracket \mathbf{F} \rrbracket, \tag{10}$$

where s represents the normal velocity of the shock wave, and for any given variable f ,

$$\llbracket f \rrbracket = f_1 - f_2, \tag{11}$$

where f_1 and f_2 are the values of f on different sides of the discontinuity.

The RH relation can also be deduced from and is equal to the weak form conservation laws across shock waves. More information can be found in [23]. Here, we provide the RH relation and the corresponding constraints for 1D Burgers equation, as well as 1D and 2D Euler equations.

3.2.1 1D Burgers’ Equation

For Burgers’ equation (5), the RH relation is expressed as follows:

$$s \llbracket u \rrbracket = \llbracket u^2/2 \rrbracket, \tag{12}$$

where the velocity of the discontinuity is given by

$$s = \frac{u_1 + u_2}{2}. \tag{13}$$

In the case of the given initial antisymmetric condition in (5), as the discontinuity forms at the center $x = 1$, the value of s remains zero. Therefore, we have the RH constraint as

$$f_{RH}(x, t) = u(x = 1, t) - u(x = 1, 0) = 0, \tag{14}$$

which can be considered as an additional constraint for the Burgers equation. This constraint is formulated as:

$$\mathcal{L}_{RH} = \text{MSE}(f_{RH}) = \frac{1}{|\mathcal{S}_{RH}|} \sum_{\mathbf{x}_i \in \mathcal{S}_{RH}} f_{RH,i}^2. \tag{15}$$

The settings of \mathcal{S}_{RH} and other sampling sets mentioned in this article can be referred to in Fig. 4. Of course, \mathcal{S}_{RH} and \mathcal{S}_{PDE} can be completely identical here.

3.2.2 Euler Equations

We next consider the 1D and 2D Euler equations in their conservative forms,

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0. \tag{16}$$

For the 1D case, the Euler equations are defined as

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}, \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(E + p) \end{pmatrix}. \tag{17}$$

For 2D cases, we have two flux vectors, \mathbf{F}_1 and \mathbf{F}_2 , with the following definition:

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \mathbf{F}_1 = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E + p) \end{pmatrix}, \mathbf{F}_2 = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E + p) \end{pmatrix}. \tag{18}$$

In these equations, ρ represents density, u is the velocity in the x direction, v is the velocity in the y direction, p is the pressure, and E stands for the total energy. To close the equations, we apply the ideal gas equation of state:

$$E = \frac{1}{2} \rho u^2 + \frac{p}{\gamma - 1}, \tag{19}$$

Here, $\gamma = 1.4$ represents the specific heat ratio.

Now, let's discuss the RH relations. We will first address the 2D Euler equations and subsequently, the 1D case.

For 2D Euler equations, the RH relation is given as

$$\begin{aligned} s \llbracket \rho \rrbracket - \llbracket \rho \vec{u} \cdot \vec{n} \rrbracket &= 0, \\ s \llbracket \rho u \rrbracket - \llbracket (\rho \vec{u} \cdot \vec{n}) u + p n_x \rrbracket &= 0, \\ s \llbracket \rho v \rrbracket - \llbracket (\rho \vec{u} \cdot \vec{n}) v + p n_y \rrbracket &= 0, \\ s \llbracket E \rrbracket - \llbracket (E + p) \vec{u} \cdot \vec{n} \rrbracket &= 0. \end{aligned} \tag{20}$$

Here, $\vec{n} = (n_x, n_y)$ represents the unit normal vector of the discontinuity. To simplify, we can eliminate the variables s and \vec{n} from (20), resulting in the following two equations,

$$\rho_1 \rho_2 [(u_1 - u_2)^2 + (v_1 - v_2)^2] = (p_1 - p_2)(\rho_1 - \rho_2), \tag{21}$$

$$\rho_1 \rho_2 (e_1 - e_2) = \frac{1}{2} (p_1 + p_2)(\rho_1 - \rho_2). \tag{22}$$

These two equations are commonly employed in the field of gas dynamics, with the latter being the renowned Hugoniot relation. These two relations are also valid for 1D case where $v_1 = v_2 = 0$. Then we let

$$\begin{aligned} f_1(\mathbf{U}_1, \mathbf{U}_2) &= \rho_1 \rho_2 [(u_1 - u_2)^2 + (v_1 - v_2)^2] - (p_1 - p_2)(\rho_1 - \rho_2), \\ f_2(\mathbf{U}_1, \mathbf{U}_2) &= \rho_1 \rho_2 (e_1 - e_2) - \frac{1}{2} (p_1 + p_2)(\rho_1 - \rho_2). \end{aligned} \tag{23}$$

To establish the constraint in PINNs, for each RH residual point $\mathbf{x}(x, y, t) \in \mathcal{S}_{RH}$, we introduce two companion points as \mathbf{x}_L and \mathbf{x}_D :

$$\mathbf{x}_L = (x - \Delta x, y, t), \quad \mathbf{x}_D = (x, y - \Delta y, t).$$

Then we utilize them to construct the RH constraint as

$$\begin{aligned} \mathcal{L}_{RH} = & \text{MSE}(\lambda_2(\mathbf{U}, \mathbf{U}_L) f_1(\mathbf{U}, \mathbf{U}_L)) + \text{MSE}(\lambda_2(\mathbf{U}, \mathbf{U}_L) f_2(\mathbf{U}, \mathbf{U}_L)) + \\ & \text{MSE}(\lambda_2(\mathbf{U}, \mathbf{U}_D) f_1(\mathbf{U}, \mathbf{U}_D)) + \text{MSE}(\lambda_2(\mathbf{U}, \mathbf{U}_D) f_2(\mathbf{U}, \mathbf{U}_D)), \end{aligned} \tag{24}$$

where $\mathbf{U} = \mathbf{U}(\mathbf{x})$, $\mathbf{U}_L = \mathbf{U}(\mathbf{x}_L)$ and $\mathbf{U}_D = \mathbf{U}(\mathbf{x}_D)$, respectively.

The hybridization of strong form and weak form equations can be effective, with the key factor being how well we can capture shock waves. In the first part, the added physical weight can adaptively weaken the strong-form PDEs near shock waves with a weak form. However, in the second part, the RH relation is derived under the assumption of shock waves. Therefore, we design a new strong-form filter to detect shock waves using the following expression:

$$\lambda_2(\mathbf{U}_1, \mathbf{U}_2) = \begin{cases} |(p_1 - p_2)(\vec{u}_1 - \vec{u}_2)| & \text{if } |p_1 - p_2| > \varepsilon_1 \text{ and } |u_1 - u_2| > \varepsilon_2, \\ 0 & \text{elsewhere.} \end{cases} \tag{25}$$

Here, ε_1 and ε_2 are two parameters used to detect jumps in shock waves and can be adjusted according to the specific problem. After normalizing and dimensionless scaling of the problem, we set $\varepsilon_1 = \varepsilon_2 = 0.1$ for all the cases in this paper.

3.3 A Total Physical Conservation Constraint

In solving Euler equations, physical conservation laws are of paramount importance, especially when it comes to accurately determining the position of shock waves. We can explain this by reconsidering the RH relation (20). In 1D case, assume we know (ρ_1, u_1, p_1) on one side, there are three equations but four variables left, as the velocity of the discontinuity s is also unknown. As a conclusion, the discontinuity cannot be completely determined solely by RH relation (local conservation laws), so additional conservation relations from the left and right smooth parts are necessary. However, in PINNs, even though we utilize a system of conservation equations, we still cannot achieve the theoretical total conservation properties as with finite difference or finite volume methods. Therefore, in the third part of this work, in order to improve the conservation, we add a soft total conservation constraint to PINNs-WE framework.

Here we consider the conservation of the total mass, momentum and total energy, which are defined as

$$\begin{aligned} \text{Mas}_{\text{Exact}} &= \int_V \rho(\mathbf{x}) dV, \\ \text{Mom}_{\text{Exact}} &= \int_V \rho(\mathbf{x}) \vec{u}(\mathbf{x}) dV, \\ \text{Ene}_{\text{Exact}} &= \int_V E(\mathbf{x}) dV. \end{aligned} \tag{26}$$

The conservation of total mass, momentum, and total energy between time $t = t_1$ and $t = t_2$ can be expressed as follows:

$$\begin{aligned}
 \int_V \rho dV|_{t=t_2} - \int_V \rho dV|_{t=t_1} &= \int_{t_1}^{t_2} \oint_{\partial V} \rho(\vec{u} \cdot \vec{n}_{\partial V}) dt dA, \\
 \int_V \rho \vec{u} dV|_{t=t_2} - \int_V \rho \vec{u} dV|_{t=t_1} &= \int_{t_1}^{t_2} \oint_{\partial V} \rho \vec{u}(\vec{u} \cdot \vec{n}_{\partial V}) + p \vec{n}_{\partial V} dt dA, \\
 \int_V E dV|_{t=t_2} - \int_V E dV|_{t=t_1} &= \int_{t_1}^{t_2} \oint_{\partial V} (E + p)(\vec{u} \cdot \vec{n}_{\partial V}) dt dA,
 \end{aligned} \tag{27}$$

where $E = \rho e + \frac{1}{2} \rho \vec{u}^2$, V is the volume of the computational domain, and $\mathbf{x} = (t = t_k, x)$ for the 1D problem and $\mathbf{x} = (t = t_k, x, y)$ for the 2D problem. ∂V is the boundary of the computational domain, and $\vec{n}_{\partial V}$ is the unit normal vector of the boundary.

As the residual points are random or uniform sampled, we approximate the total conservation constraints at time $t = t_k$ as

$$\begin{aligned}
 \text{Mas}(t_k) &= \frac{V}{|\mathcal{S}_{\text{Con}}(t_k)|} \sum_{\mathbf{x} \in \mathcal{S}_{\text{Con}}(t_k)} \rho(\mathbf{x}) \\
 \text{Mom}(t_k) &= \frac{V}{|\mathcal{S}_{\text{Con}}(t_k)|} \sum_{\mathbf{x} \in \mathcal{S}_{\text{Con}}(t_k)} \rho(\mathbf{x}) \vec{u}(\mathbf{x}) \\
 \text{Ene}(t_k) &= \frac{V}{|\mathcal{S}_{\text{Con}}(t_k)|} \sum_{\mathbf{x} \in \mathcal{S}_{\text{Con}}(t_k)} E(\mathbf{x})
 \end{aligned} \tag{28}$$

The boundary terms can also be approximated as

$$\begin{aligned}
 \text{BD}_{\text{Mas}}(t_1, t_2) &= \frac{(t_2 - t_1)A}{|\mathcal{S}_{\text{BD}}|} \sum_{\mathbf{x} \in \mathcal{S}_{\text{BD}}} \rho(\mathbf{x})(\vec{u}(\mathbf{x}) \cdot \vec{n}_{\partial V}(\mathbf{x})), \\
 \text{BD}_{\text{Mom}}(t_1, t_2) &= \frac{(t_2 - t_1)A}{|\mathcal{S}_{\text{BD}}|} \sum_{\mathbf{x} \in \mathcal{S}_{\text{BD}}} [\rho(\mathbf{x})(\vec{u}(\mathbf{x}))^2 + p(\mathbf{x})] \cdot \vec{n}_{\partial V}(\mathbf{x}), \\
 \text{BD}_{\text{Ene}}(t_1, t_2) &= \frac{(t_2 - t_1)A}{|\mathcal{S}_{\text{BD}}|} \sum_{\mathbf{x} \in \mathcal{S}_{\text{BD}}} [E(\mathbf{x}) + p(\mathbf{x})] \vec{u}(\mathbf{x}) \cdot \vec{n}_{\partial V}(\mathbf{x}),
 \end{aligned} \tag{29}$$

where \mathcal{S}_{BD} is the set of residual points sampled on the boundary over the computational time $t_1 \leq t \leq t_2$, and A is the area of the boundary surface. Then the conservation loss is constructed as

$$\begin{aligned}
 \mathcal{L}_{\text{CONs}} &= (\text{Mas}(t_2) - \text{Mas}(t_1) - \text{BD}_{\text{Mas}})^2 + (\text{Mom}(t_2) - \text{Mom}(t_1) - \text{BD}_{\text{Mom}})^2 \\
 &\quad + (\text{Ene}(t_2) - \text{Ene}(t_1) - \text{BD}_{\text{Ene}})^2.
 \end{aligned} \tag{30}$$

4 Numerical Examples

We employ PINNs-WE to address forward problems involving the Burgers' equation and Euler equations, without relying on prior information from the data, except for the initial boundary conditions (IBCs). We then compare the results with those obtained using traditional high-order finite difference methods, specifically the fifth-order WENO-Z method for spatial discretization [24] and the third-order Runge–Kutta method for time integration [25].

4.1 Inviscid Burgers' Equation

We first solve the inviscid Burgers' equation (5) using the new PINNs-WE method with the same setting as in Sect. 2.2.

In Burgers' equation, as the position of the discontinuity is determined by the antisymmetric condition as discussed in Eq. 14, so the introduction of total conservation of u , that is $\int u dx = 0$, will be redundant. So the total loss is constructed as

$$\mathcal{L} = \omega_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \omega_{\text{IBCs}}\mathcal{L}_{\text{IBCs}} + \omega_{\text{RH}}\mathcal{L}_{\text{RH}}. \tag{31}$$

Here, the ω is weight for each term to adjust the confinement strength [16, 18, 20]. Typically, more weight is assigned to points located on initial and boundaries and near the shocks. We set

$$\omega_{\text{PDE}} = 1, \quad \omega_{\text{IBCs}} = 10, \quad \omega_{\text{RH}} = 10, \tag{32}$$

the velocity is represented by u itself. Therefore, the weight in \mathbf{G}_{new} is given by:

$$\lambda_1 = \frac{1}{k_1(|\frac{\partial u}{\partial x}| - \frac{\partial u}{\partial x}) + 1}. \tag{33}$$

The RH constraint is defined as in (15). All the residual points are uniformly distributed. The RH points are sampled at $x = 1$ for $0 < t < 1$.

Figure 5 displays the loss history over epochs, the evolution of u , the residual, and λ_1 as functions of x at $t = 1$ for different training epochs (1000, 3000, 5000, 8000, 11500, and the end of training).

Compared to the results in Fig. 1, in the first 1000 epochs (Slice 1), PINNs-WE produce results similar to the original PINNs. This is because the gradient of the transition point is not sufficiently large to cause problems. However, after 3000 epochs, PINNs-WE can effectively reduce the total loss as it resolves the paradoxical situation at the transition points. Subsequently, all the transition points are compressed into a smooth region, except for the one where $u = 0$ and always has zero velocity.

As shown in Fig. 5d, after 3000 epochs, all the weights change to nearly 1, except for the central one.

We then proceed to compare PINNs-WE against the original PINNs and the traditional high-order WENO-Z method, considering various factors, including the number of mesh-based residual points, network structure, and the factor k_1 . All the presented results are the averages of ten random runs to mitigate the influence of hyperparameter randomness. We employed the L-BFGS algorithm for training to convergence.

Table 1 provides the L_2 errors, L_2 errors within smooth regions ($x \in [0, 0.95] \cup [1.05, 2], t = 1$), and within the shock region ($x \in (0.95, 1.05)$), along with L_∞ errors. Additionally, the training losses after convergence in different scenarios are displayed in the same table. We also give a rough comparison of the computational cost between different methods. The comparison illustrates that:

1. For PINNs-WE, when the network parameters are sufficient, the network structure (depth and width) does not have a significant impact on the results. The variation of parameter k_1 within the range of 0.1 to 0.4 also does not significantly affect the results.
2. The accuracy of the new method, PINNs-WE, consistently outperforms the original PINNs in this case.
3. When there are insufficient cells/sampling points, the accuracy of PINNs-WE is higher than that of the WENO-Z scheme.

4. Traditional methods exhibit mesh convergence, a property that PINNs do not possess. In fact, even with an increase in residual points, the accuracy of PINNs-WE may decrease. This is mainly due to the increased difficulty in optimization with more points.
5. Using PINNs to solve forward problems is much more expensive than traditional methods (Fig. 6).

4.2 Euler Equations

In 1D cases, we examine two classical Riemann problems, namely the Sod and Lax problems, as well as another 1D problem characterized by strong shock waves. Subsequently, we transition to a 2D Riemann problem and a more intricate problem involving moving shock waves. The loss function is constructed as

$$\mathcal{L} = \omega_{\text{PDE}} \mathcal{L}_{\text{PDE}} + \omega_{\text{IBCs}} \mathcal{L}_{\text{IBCs}} + \omega_{\text{RH}} \mathcal{L}_{\text{RH}} + \omega_{\text{CONS}} \mathcal{L}_{\text{CONS}}. \quad (34)$$

and

$$\omega_{\text{PDE}} = 1, \quad \omega_{\text{IBCs}} = 10, \quad \omega_{\text{RH}} = 10, \quad \omega_{\text{CONS}} = 10. \quad (35)$$

4.2.1 Sod Problem

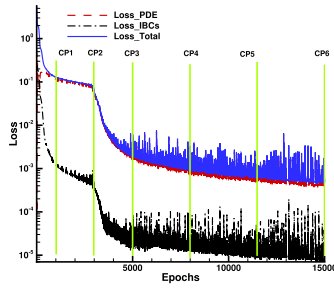
The Sod problem is an extensively studied 1D Riemann problem with the initial constant states in a tube with unit length, formulated as follows:

$$(\rho, u, p) = \begin{cases} (1, 0, 1), & \text{if } 0 \leq x \leq 0.5, \\ (0.1, 0, 0.125), & \text{if } 0.5 < x \leq 1. \end{cases} \quad (36)$$

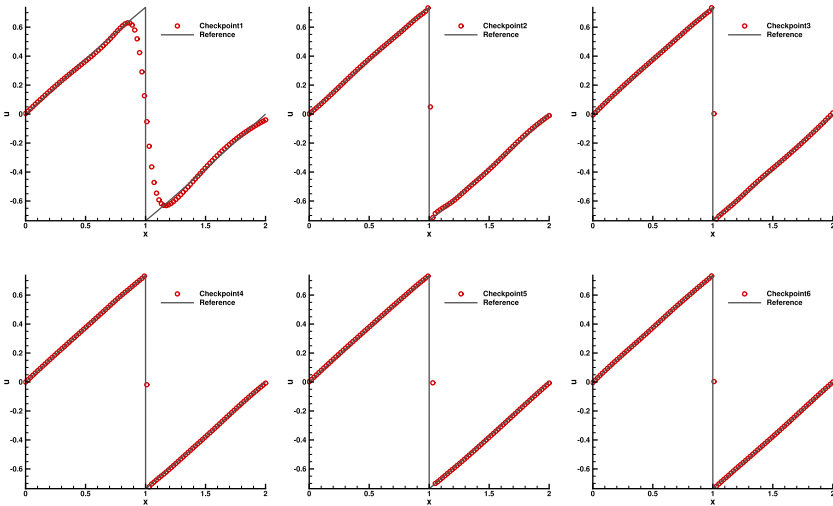
We use a NN with 7 hidden layers, each containing 50 neurons. We then set 5000 function residual points using the Latin Hypercube Sampling (LHS) method in the $X \times T$ space. The number of initial points is $N_{\text{IC}} = 100$. The boundary condition is omitted for the reasons explained in Appendix A. There are 100 residual points at time $t = 0.2$, referred to as N_{RH} . Additionally, we have a total of 100 conservation points at both $t_1 = 0$ and $t_2 = 0.2$, denoted as N_{CONS} . After training, we construct a test set consisting of 100 uniformly spaced points in the range $x \in [0, 1]$ at the final time $t = 0.2$.

First, we compare the results obtained with the well-trained PINNs-WE to the traditional high-order WENO-Z method using 100 cells in space. Figure 7 illustrates that PINNs-WE achieves similar or even superior results when compared to the WENO-Z method. Notably, when capturing shock waves, there are no transition points within the shock region because PINNs-WE does not introduce any dissipation into the equations.

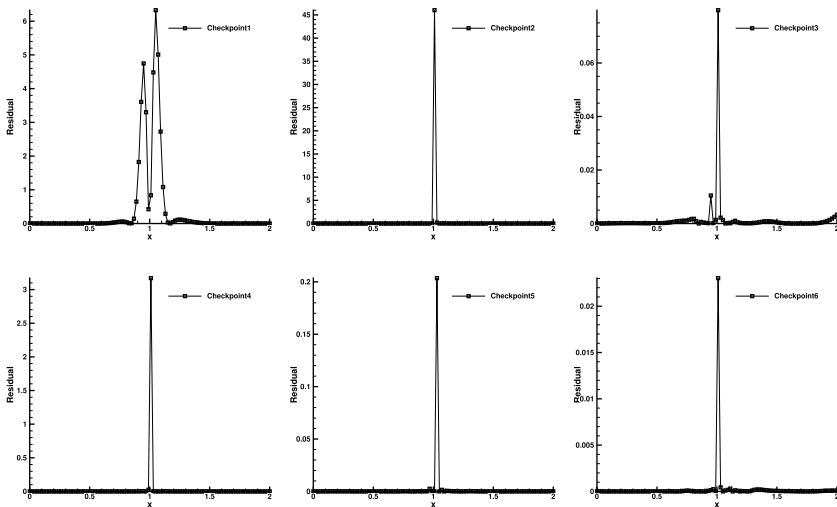
In Fig. 8, we present a result from the network that is still undergoing training. We compare the pointwise function residuals, both with and without the weight λ_1 , and the equation weight λ_1 at time $t = 0.2$ between the well-trained network and the network during training. This comparison demonstrates that, during training, as the shock wave forms, the function residual within the discontinuous region dominates the training process. The design of the equation weight λ_1 is effective in reducing the residuals near the shock, thereby balancing the training process and achieving high accuracy results with sharp shock waves.



(a) Loss history



(b) u at $t = 1$ at different epochs (1000, 3000, 5000, 8000, 11500 and end of the training)



(c) Residual of x at $t = 1$ at different epochs (1000, 3000, 5000, 8000, 11500 and end of the training)

Fig. 5 Results of inviscid Burgers' equation with PINNs-WE (Part 1). Similar computational setting with Fig. 1

Table 1 Accuracy on inviscid Burgers' equation, $\tau = 1$

Hidden layers ⁽¹⁾	$N^{(2)}$	k_1	L_2	Shock L_2	Smooth L_2	L_∞	Final Loss	Runtime(s) ⁽³⁾
PINNs-WE								
3 × 30	10	0.2	1.1×10^{-2}	8.0×10^{-4}	1.2×10^{-2}	2.7×10^{-2}	2.0×10^{-5}	66
3 × 30	20	0.2	2.8×10^{-3}	9.3×10^{-4}	2.9×10^{-3}	1.1×10^{-2}	2.9×10^{-5}	92
3 × 30	40	0.2	2.2×10^{-3}	6.5×10^{-3}	1.5×10^{-3}	1.2×10^{-2}	4.7×10^{-5}	116
3 × 30	80	0.2	2.0×10^{-3}	8.1×10^{-3}	7.0×10^{-4}	1.5×10^{-2}	7.6×10^{-5}	108
3 × 30	160	0.2	2.0×10^{-3}	7.0×10^{-3}	1.0×10^{-3}	1.8×10^{-2}	2.8×10^{-4}	128
3 × 30	320	0.2	4.1×10^{-3}	1.3×10^{-2}	2.9×10^{-3}	2.1×10^{-2}	1.5×10^{-3}	268
4 × 60	80	0.2	2.0×10^{-3}	6.1×10^{-3}	1.4×10^{-3}	1.1×10^{-2}	2.5×10^{-4}	
3 × 30	80	0.4	2.5×10^{-3}	7.8×10^{-3}	1.6×10^{-3}	1.4×10^{-2}	3.0×10^{-4}	
3 × 30	80	0.1	3.0×10^{-3}	8.2×10^{-3}	2.1×10^{-3}	1.6×10^{-2}	6.4×10^{-4}	
PINNs								
3 × 30	10		3.2×10^{-2}	4.7×10^{-2}	2.4×10^{-2}	8.6×10^{-2}	6.7×10^{-4}	61
3 × 30	20		6.7×10^{-3}	7.1×10^{-3}	6.4×10^{-3}	2.6×10^{-2}	5.6×10^{-4}	93
3 × 30	40		6.2×10^{-3}	9.6×10^{-3}	5.8×10^{-3}	1.8×10^{-2}	3.3×10^{-3}	86
3 × 30	80		1.3×10^{-2}	1.8×10^{-2}	1.2×10^{-2}	3.5×10^{-2}	1.1×10^{-2}	76
3 × 30	160		3.1×10^{-2}	8.3×10^{-2}	1.9×10^{-2}	1.8×10^{-1}	3.0×10^{-2}	128
3 × 30	320		3.7×10^{-2}	1.1×10^{-1}	2.3×10^{-2}	2.8×10^{-1}	8.2×10^{-2}	245

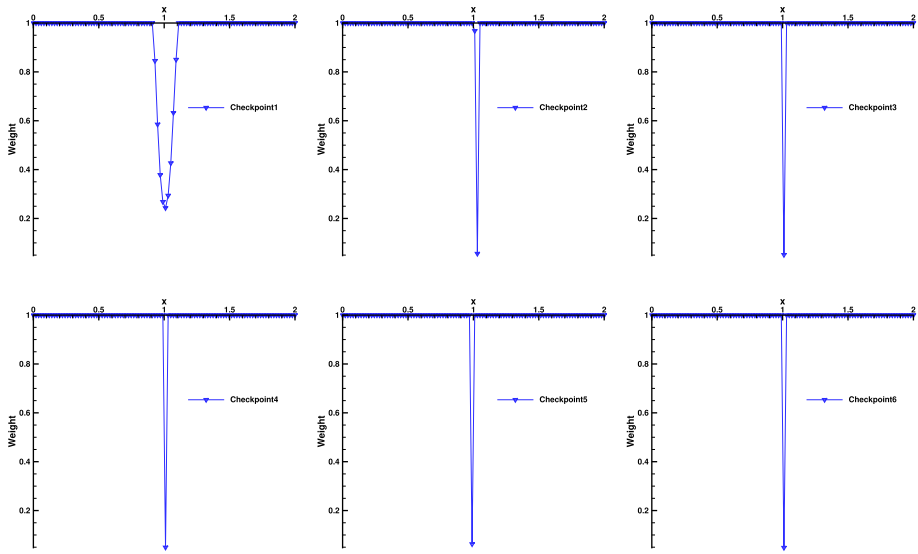
Table 1 continued

Hidden layers ⁽¹⁾	$N^{(2)}$	k_1	L_2	Shock L_2	Smooth L_2	L_∞	Final Loss	Runtime(s) ⁽³⁾
WENO-Z								
	10		5.6×10^{-2}	1.0×10^{-1}	5.0×10^{-2}	1.3×10^{-1}		3×10^{-4}
	20		1.3×10^{-2}	3.3×10^{-2}	1.1×10^{-2}	3.3×10^{-2}		4×10^{-4}
	40		3.9×10^{-3}	6.9×10^{-3}	3.6×10^{-3}	8.9×10^{-3}		1×10^{-3}
	80		2.0×10^{-3}	5.9×10^{-3}	1.6×10^{-3}	9.7×10^{-3}		6×10^{-3}
	160		1.2×10^{-3}	4.2×10^{-3}	7.8×10^{-4}	9.6×10^{-3}		2×10^{-2}
	320		7.9×10^{-4}	3.0×10^{-3}	4.3×10^{-4}	9.8×10^{-3}		6×10^{-2}

¹ $n \times m$ stands for n hidden layers and m neurons per layer

² N is the number of points in each dimension, the function points $N_f = N^2$ and the IBCs points $N_{IBC_s} = N$

³ Just a rough comparison, calculations are based on different devices, PINNs and PINNs-WE are on GPU while WENO-Z is based on CPU



(d) Weight λ of x at $t = 1$ at different epochs (1000, 3000, 5000, 8000, 11500 and end of the training)

Fig. 6 Results of Burgers’ equation with PINNs-WE (Part 2). Similar computational setting with Fig. 1

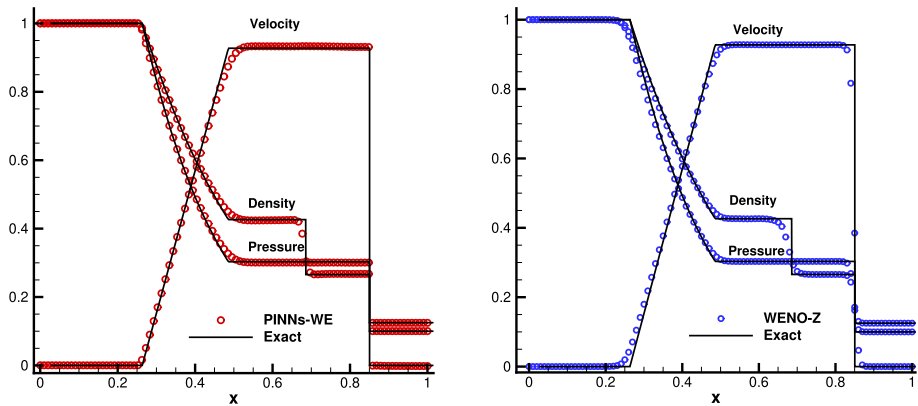


Fig. 7 Evaluation of the Sod problem at $t = 0.2$

4.2.2 Lax Problem

The Lax problem is another a Riemann problem featuring a strong shock and strong contact discontinuity. The initial conditions are defined as follows:

$$(\rho, u, p) = \begin{cases} (0.445, 0.698, 3.528), & \text{if } 0 \leq x \leq 0.5, \\ (0.5, 0, 0.571), & \text{if } 0.5 < x \leq 1. \end{cases} \quad (37)$$

The computational domain is defined as $t \in [0, 1.4]$ and $x \in [0, 1]$ within the $T \times X$ space. Similar to the Sod problem, you have used the same neural network (NN) architecture. In this case, you randomly selected 50,000 interior points from a uniform 1000×5000 mesh in

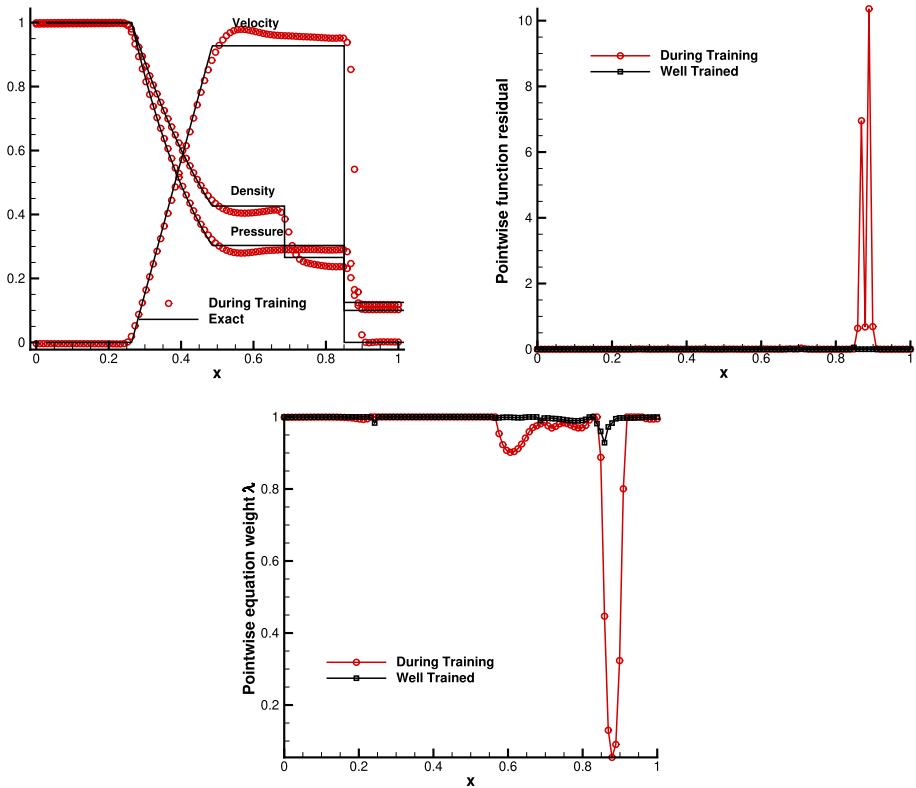


Fig. 8 The pointwise function residual and equation weight distribution at an epoch during training for the Sod problem

the $X \times T$ space. There are 1000 initial points, and the number of residual points, RH points, and conservation points is the same as in the Sod case.

We compare the results with those obtained from the high-order WENO-Z method. Figure 9 shows that the PINNs-WE can accurately simulate this problem and gains a sharper shock than that by WENO-Z. We also give a result from the network at one epoch that is still during training, and we take a comparison of the pointwise function residuals without the weight λ_1 and the equation weight λ_1 at time $t = 1.4$ between the well trained and during training network. It illustrates that the design of the equation weight λ_1 can balance the training process and achieving high accuracy results with sharp shock waves.

4.2.3 Two Shock Waves Problem

At last of the 1D cases, we consider a problem with left and right running strong shock waves. The initial condition is given as

$$(\rho, u, p) = \begin{cases} (1, 0, 1), & \text{if } -0.1 \leq x - 0.5 \leq 0.1, \\ (1, 0, 0.01), & \text{otherwise.} \end{cases} \quad (38)$$

This problem can be viewed as a normalization and simplification of the classical 1D blast problem. The shock waves are generated due to a pressure difference 100 times greater

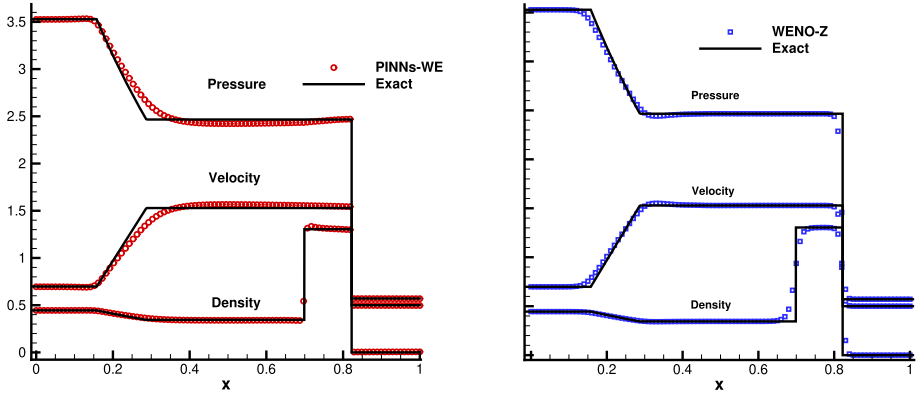


Fig. 9 Results for the Lax problem

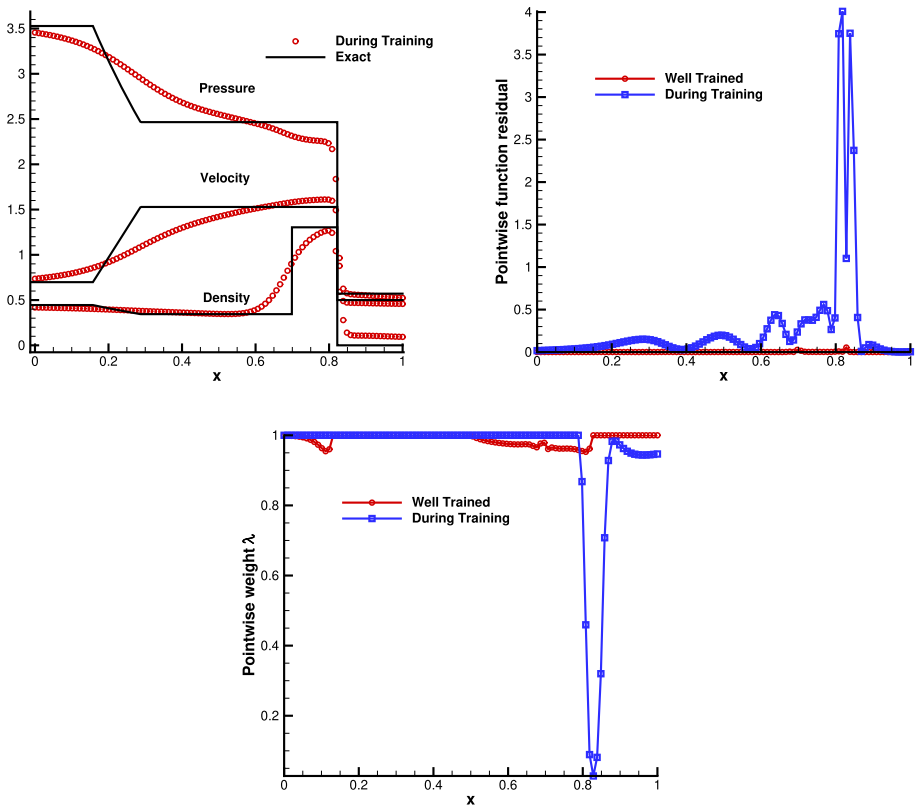


Fig. 10 The pointwise function residual and equation weight distribution at an epoch during training for the Lax problem

between the center and the left and right sides. The computational time is $t = 0.036$. Other settings are same to those in Sod problem.

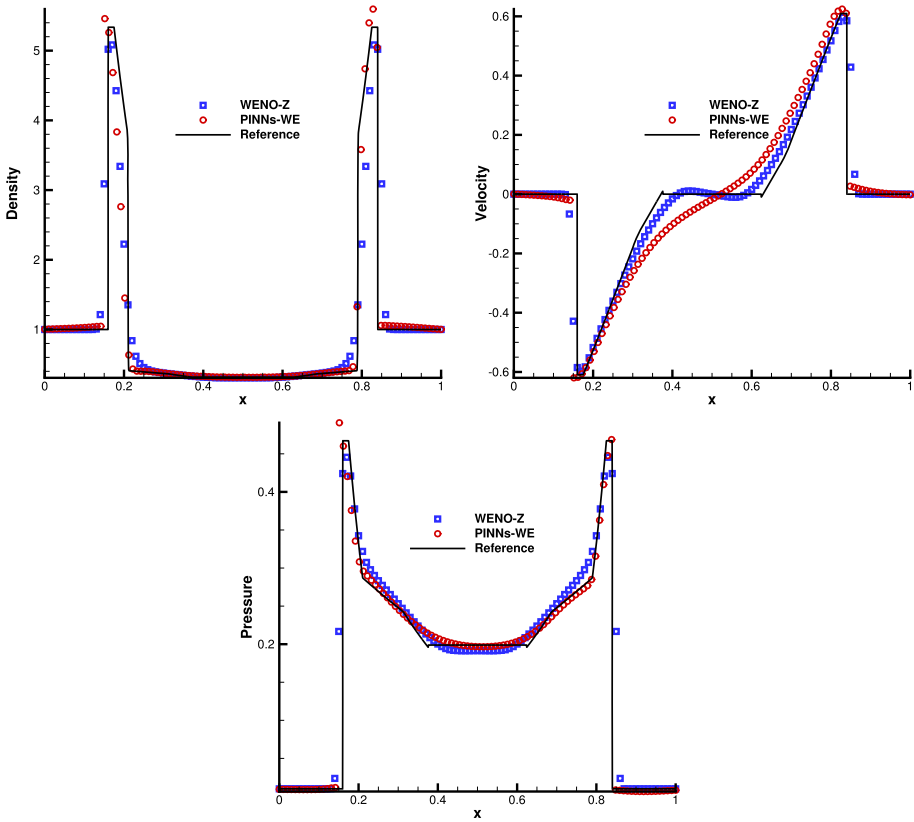


Fig. 11 Evaluation of the two shock waves problem at $t = 0.32$

In Fig. 11, we give a comparison at time $t = 0.32$. They illustrate that PINNs-WE can accurately capture the left and right shock waves. Compared to WENO-Z with a similar number of grid points, despite a slightly lower accuracy in velocity, the shock resolution of the new method still is higher.

4.2.4 2D Riemann Problem

Next, we consider a 2D problem with strong discontinuities. The basic settings are retrieved from case 8 in [26]. The initial condition is given by

$$(\rho, u, v, p) = \begin{cases} (1, -0.75, 0.5, 1) & \text{if } 0 \leq x \leq 0.5, 0 \leq y \leq 0.5, \\ (2, 0.75, 0.5, 1) & \text{if } 0 \leq x \leq 0.5, 0.5 < y \leq 1, \\ (3, -0.75, -0.5, 1) & \text{if } 0.5 < x \leq 1, 0 \leq y \leq 0.5, \\ (1, 0.75, -0.5, 1) & \text{if } 0.5 \leq x \leq 1, 0.5 < y \leq 1. \end{cases} \quad (39)$$

And $t \in [0, 0.4]$. We use an NN with 6 hidden layers and 60 neurons per layer. The training points are obtained by Latin hypercube sampling, with 200,000 interior points in the $T \times X \times Y$ space and 10,000 initial points in the $X \times Y$ space. The final training loss is 0.009.

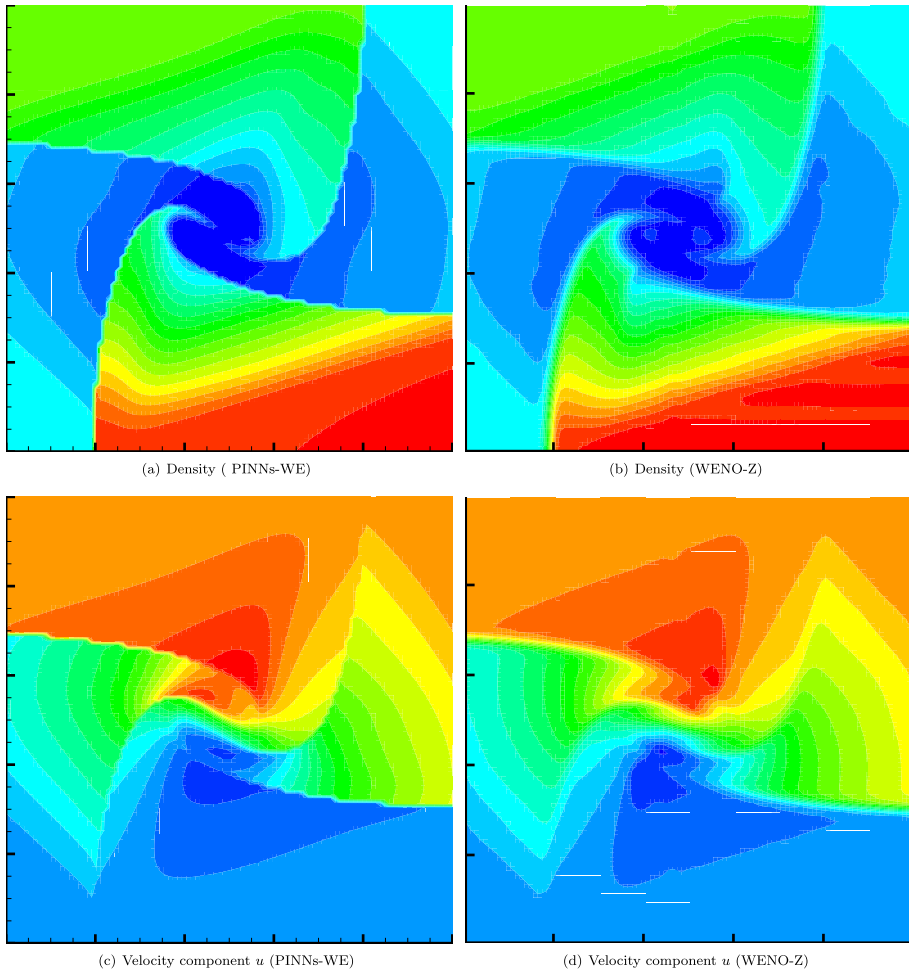


Fig. 12 Results of 2D Riemann problem (part 1) with 100×100 test points for PINNs-WE and the same number of mesh grids for WENO-Z

We provide test results for 100×100 mesh points, which slightly outnumber the training points (approximately 60) along each dimension. Comparison results are shown in Figs. 12 and 13 with the WENO-Z method in the same 100×100 mesh. The proposed method can capture contact discontinuities more sharply and nearly without transition points, which are unavoidable by the traditional high-order method.

Then, we increased the test points to meshly 400×400 , which is substantially larger than the training data along each dimension. And we performed an unfair comparison with the WENO-Z method in a 400×400 mesh in Figs. 14 and 15. The computation of the detailed structure, particularly in the middle region, is weaker in the PINNs-WE because of the available training data, but discontinuities are still sharper than those computed by the WENO-Z. These results illustrate the advantages of the PINNs-WE in high dimensions given its mesh-free feature.

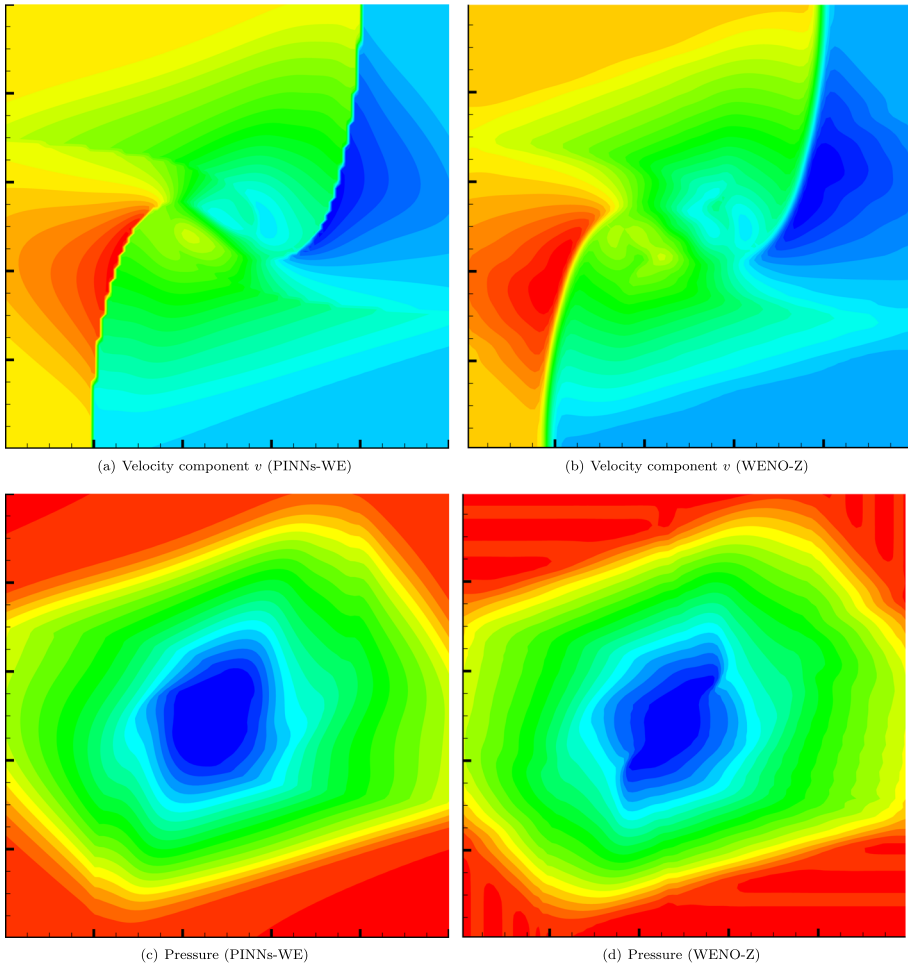


Fig. 13 Results of 2D Riemann problem (part 2) with 100×100 test points for PINNs-WE and the same number of mesh grids for WENO-Z

4.2.5 Transonic Flow Around Circular Cylinder

In our final analysis, we delve into a 2D problem involving a bow shock. Specifically, we examine a transonic flow with a Mach number of 0.728 passing around a stationary circular cylinder. A similar problem, along with detailed analysis, can be found in [27]. The initial condition is defined by a uniform flow with the parameters $(\rho, u, v, p) = (2, 112, 1.028, 0, 3.011)$. The computational domain spans t from 0 to 0.4, x from 0 to 1.5, and y from 0 to 2 within the $T \times X \times Y$ domain. The center of the cylinder, with a radius of 0.25, is positioned at coordinates (1, 1).

For this analysis, we employ a neural network (NN) consisting of 7 hidden layers, each with 90 neurons. The residual points for our computations are obtained through Latin hypercube sampling within the 3D domain of $T \times X \times Y$, totaling 300,000 points. Additionally, we randomly select 15,000 boundary points along the cylinder, and another 15,000 initial points

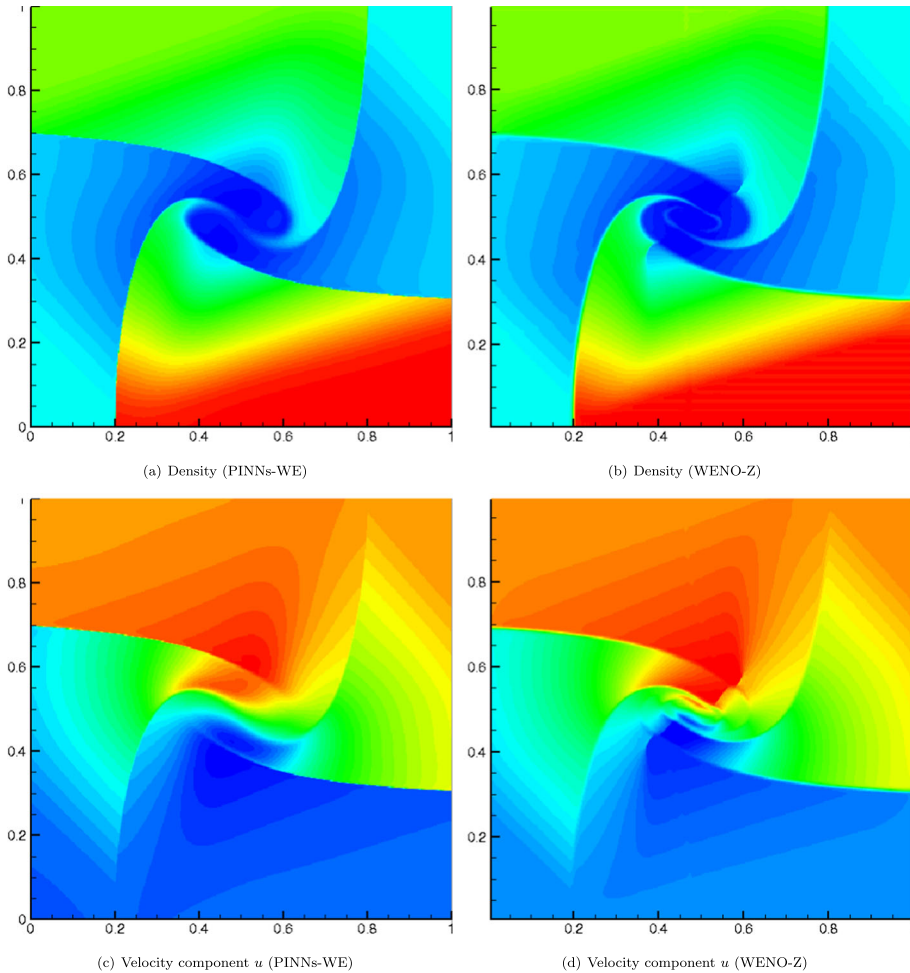


Fig. 14 Results of 2D Riemann problem II (part 1), an unfair comparison, with 400×400 test points for PINNs-WE and the same number of mesh grids for WENO-Z

are obtained using Latin hypercube sampling. After 2000 optimization steps using the L-BFGS algorithm, we achieve a total loss of 0.028.

Figures 16, 17, 18 and 19 present a comparative analysis of the results obtained using PINNs-WE and the WENO-Z method. The results from PINNs-WE exhibit sharpness similar to those of WENO-Z, but they display a smoother profile. Notably, our approach effectively captures the vortex located behind the cylinder.

It's important to acknowledge certain limitations stemming from the number of residual points and the use of single-precision calculations, which are imposed by the hardware (Nvidia 1080ti) used in our study. These constraints prevent further reduction of the loss and hindered obtaining more precise results.

Nevertheless, our work underscores the promising capability of PINNs-WE in simulating complex transonic and supersonic flows. Despite these constraints, the method consistently

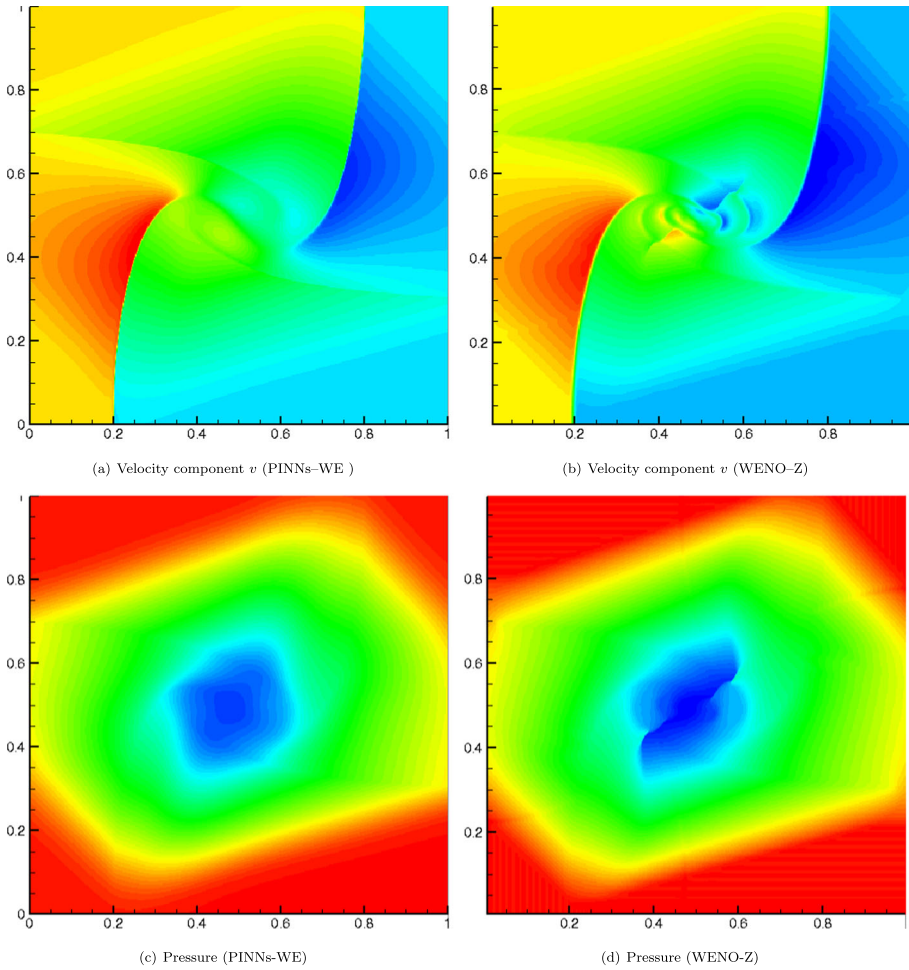


Fig. 15 Results of 2D Riemann problem II (part 2), an unfair comparison, with 400×400 test points for PINNs-WE and the same number of mesh grids for WENO-Z

produces competitive and visually appealing results, highlighting its potential in addressing challenging fluid dynamics problems.

5 Conclusions

In this paper, we introduce a Physics-Informed Neural Networks with Equation Weights (PINNs-WE) framework designed to capture strong nonlinear discontinuities, particularly shock waves, when solving hyperbolic equations. Despite the versatility of PINNs for solving inverse problems combining equations with data, our focus in this work is on forward problems. This choice allows us to analyze the fundamental characteristics of PINNs without the influence of prior data.

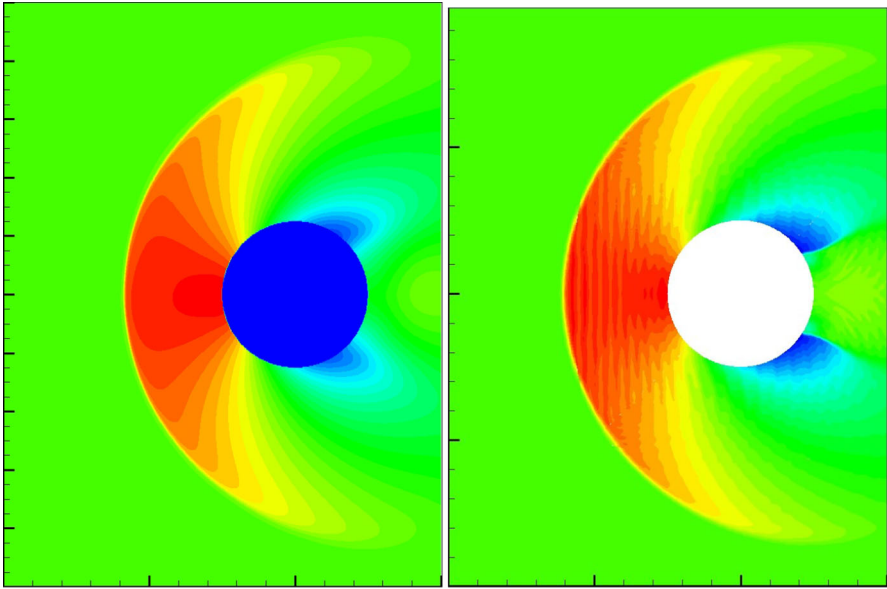


Fig. 16 Resulting pressure for transonic flow through circular cylinder using proposed PINNs-WE (left) and WENO-Z method (right)

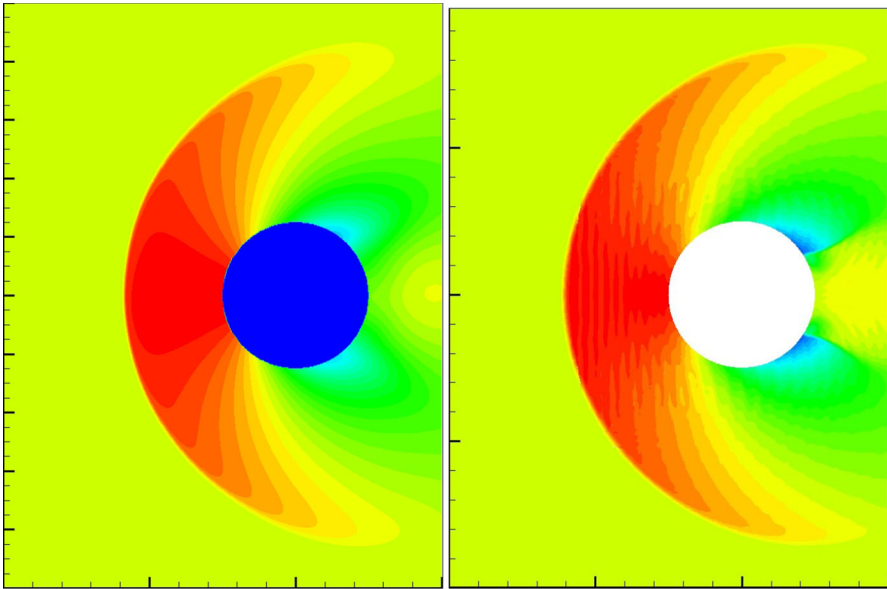


Fig. 17 Resulting density for transonic flow through circular cylinder using proposed PINNs-WE (left) and WENO-Z method (right)

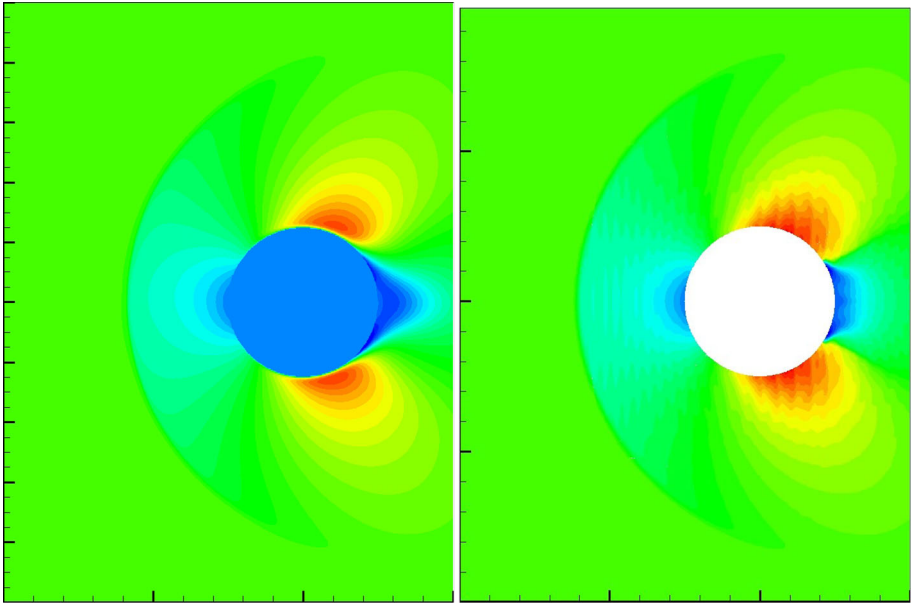


Fig. 18 Resulting velocity u for transonic flow through circular cylinder using proposed PINNs-WE (left) and WENO-Z method (right)

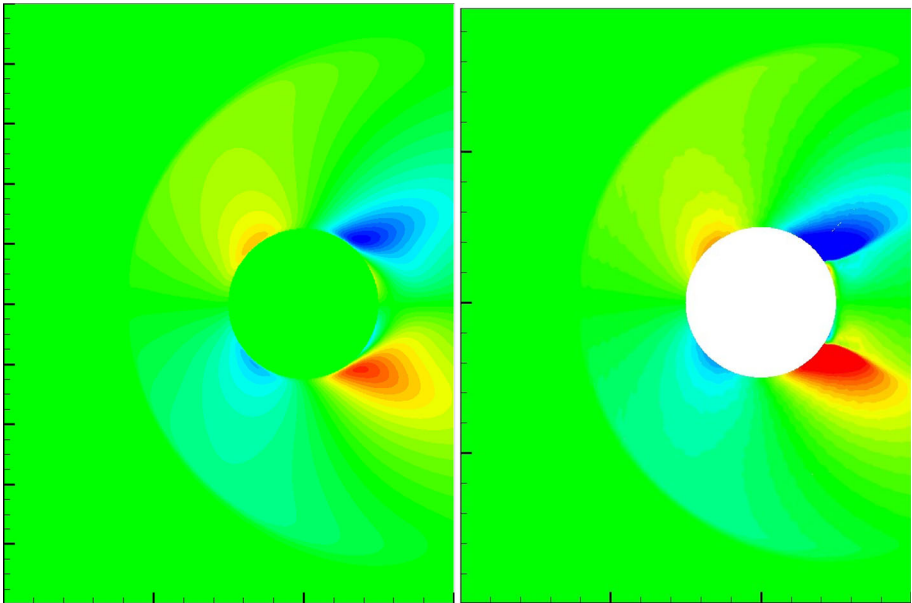


Fig. 19 Resulting velocity v for transonic flow through circular cylinder using proposed PINNs-WE (left) and WENO-Z method (right)

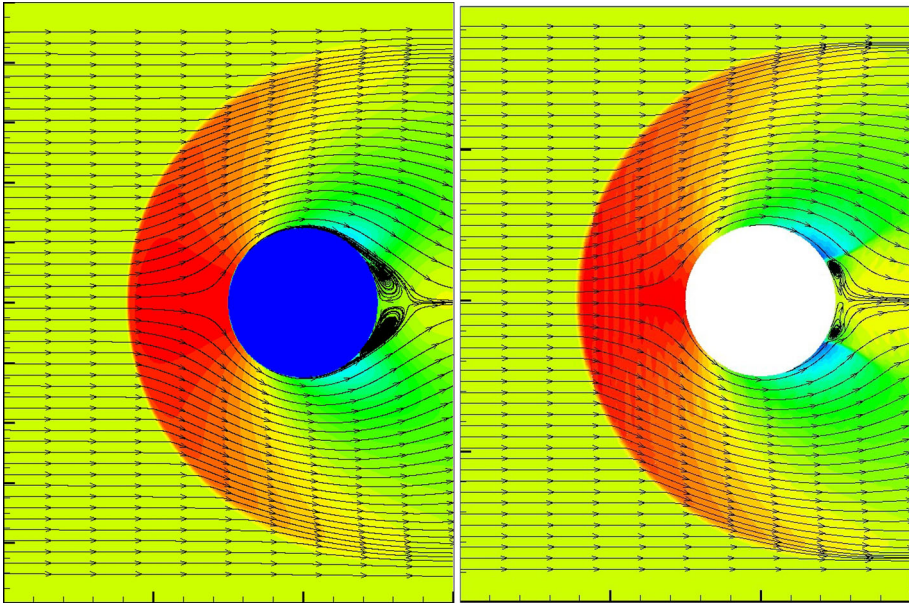


Fig. 20 Resulting streamline for transonic flow through circular cylinder using proposed PINNs-WE (left) and WENO-Z method (right)

One of the key contributions of our approach is the recognition of a paradoxical issue within transition points inside shock waves. Regardless of whether gradients are increased or decreased, these points tend to increase the total loss, potentially leading to conflicts during neural network training. To address this, we adopt three novel strategies:

1. In our framework, we first incorporate a positive physics-dependent weight into the governing equations to adapt the behavior of PINNs in regions with varying physical features. For solving the Euler equations, we construct a weight that is inversely proportional to the local physics compression, measured through the velocity divergence. By introducing these Weighted Equations (WEs) into PINNs, the neural network training primarily focuses on smoother regions, as shock regions receive small weights. Relying on the inherent physics compression learned from these smooth regions, discontinuities naturally emerge as transition points move out into smoother regions, analogous to passive particles.
2. Recognizing that strong form PDEs are not suitable for describing strong discontinuous solutions, we address the underconstrained nature of the problem by incorporating the Rankine–Hugoniot (RH) relation, which is equivalent to the weak form of conservation laws, as new constraints near the shock waves.
3. For nonlinear hyperbolic equations, such as the Euler equations, preserving physical conservation is of utmost importance. It directly impacts the accuracy of shock wave positions. Therefore, we integrate a conservation constraint into our new framework.

Furthermore, we provide a comparison between PINNs-WE and the traditional shock-capturing method, WENO-Z, in this paper. Some of the key findings include:

1. Due to the nonlinear nature of neural networks, PINNs have the potential to capture discontinuities sharper than mesh-based methods.

2. PINNs-WE can capture shock waves without obvious transition points and accurately solve rarefaction waves as there is no additional dissipation introduced, as demonstrated in the 123 problem presented in Appendix B.
3. When residual points are sparse, PINNs may outperform traditional methods in terms of accuracy.
4. However, it is important to note that PINNs involve online training, which can be more computationally expensive than traditional methods for forward problems.
5. A significant challenge with PINNs and similar methods is their lack of grid or sampling point refinement convergence. This limitation can result in significant errors when dealing with complex flows featuring fine or high-frequency structures.

Acknowledgements The authors would like to thank all the members from the corresponding author's team "AI++" for their help and fruitful discussions.

We acknowledge the financial support from the National Key R&D Program of China under Grant No 2022YFA1004500, the NSAF under Grant Number U2230208, and the Key Laboratory of Nuclear Data foundation under Grant Number JCKY2022201C155.

The author would like to thank all referees for their constructive comments and suggestions which greatly improve the paper.

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Code Availability Code is available at https://github.com/bfly123/PINN_WE.

Declaration

Conflict of Interests We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

Appendix A: Omissible Boundary Condition in PINNs

In traditional numerical methods used for solving initial boundary value problems, such as finite element and finite difference methods, boundary conditions play essential roles. They serve two primary purposes: mathematically defining the problem and numerically closing the discretization near the boundaries of the schemes. However, in PINNs, which do not rely on discretization or logical relations between sampling points, the need for boundary conditions to serve the latter purpose is eliminated. This simplifies the process of setting boundary conditions significantly.

In our experimentation with PINNs, we have observed that two types of boundary conditions may can be omitted when solving initial boundary value problems. The first type is constant boundary conditions, where the boundary values remain unchanged over time after being determined by the initial conditions. The second type is outflow boundary conditions, where information purely flows out of the domain.

Traditionally, setting outflow boundary conditions can be challenging because they require closed and discrete boundary conditions to exist while also ensuring that the set boundary conditions do not influence the internal flow. In PINNs, these boundary conditions can be omitted, yet they still ensure complete outflow characteristics at the boundary.

Table 2 Accuracy comparison with and without constant boundary constraint

	L_∞ Error	L_2 Error
Without boundaries	2.0e−3	4.8e−4
With boundaries	4.9e−3	7.3e−4

Table 3 Accuracy comparison with different set of outflow boundary constraint

	L_∞ Error	L_2 Error
Extrapolation	5.2e−3	1.9e−3
Zero gradient	4.9e−3	1.8e−3
Omitting	4.6e−3	1.3e−3

Appendix A.1: 1D Linear Transport Equation Problem

To illustrate the influence of omitting these two types of boundary conditions, we consider two problems solved with PINNs. The governing equation for both problems is chosen as:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \quad -1 < x < 2,$$

with the first initial condition as

$$u(x, 0) = \begin{cases} \sin(2\pi x) + 1 & \text{if } 0 \leq x \leq 1, \\ 1 & \text{else.} \end{cases}$$

The computational time is set as $0 \leq t \leq 0.5$. Both left and right boundary conditions are kept constant throughout the computation time. We compare the results with and without the inclusion of boundary conditions in Table 2.

Our results indicate that when boundary conditions are constant, there is minimal influence when omitting them from the loss function. The number of residual points sampled within the computational domain is denoted as $N_f = 10000$, while $N_{IC} = 100$ represents the number of residual points sampled on the initial condition. Additionally, $N_{BC} = 100$ is the number of residual points sampled on the left and right boundary conditions. All these residual points are uniformly distributed.

For the second initial condition, it is defined as:

$$u(x, 0) = \sin(2\pi x) + 1,$$

and the computational time is extended to $0 \leq t \leq 5$. The left boundary condition is set as an inflow boundary condition:

$$u(-1, t) = \sin(2\pi(-t - 1)) + 1.$$

On the other hand, the right boundary condition is chosen as an outflow condition. In traditional methods with outflow boundaries, there are various approaches to closing the discretization. One common method is to extrapolate the value at $u(2 + \Delta x, t)$ using $u(2, t)$. Here, Δx represents the mesh size and can be calculated as $\Delta x = L_x/N_x = 0.03$, where L_x denotes the length in space, and N_x is the number of points in the x direction. Alternatively, more complex zero-gradient outflow conditions, such as setting $u_x(2 + \Delta x, t) = u_x(2, t)$, can be employed to minimize the impact of boundary conditions. In this context, as you mentioned, using uniform points simplifies the choice of Δx .

In our analysis, we have tested three cases: extrapolated boundary conditions, zero-gradient boundary conditions, and omitting boundary conditions altogether with PINNs. To minimize the impact of network randomness, the results reported in Table 3 represent the averages of ten separate runs with different random seeds.

The results demonstrate that even when using zero-gradient boundary conditions, the boundary conditions still influence the accuracy of internal flows. Conversely, omitting the outflow boundary condition appears to be a better fit for representing the true physical behavior.

Appendix A.2: 2D Vortex Evaluation Problem

Then we test a 2D vortex problem governed by 2D Euler equations (16). The initial condition is considered as

$$\begin{pmatrix} \rho \\ u \\ v \\ p \end{pmatrix} = \begin{pmatrix} (1 + \delta T)^{1/(\gamma-1)} \\ 1 + (l_y/2 - y) \frac{\sigma}{2\pi} e^{0.5(1-r^2)} \\ 1 + (x - l_x/2) \frac{\sigma}{2\pi} e^{0.5(1-r^2)} \\ (1 + \delta T)^{\gamma/(\gamma-1)} \end{pmatrix}. \tag{A.1}$$

Here δT is the perturbation in the temperature and is given by

$$\delta T = -\frac{(\gamma - 1)\sigma^2}{8\gamma\pi^2} e^{(1-r^2)},$$

where $r^2 = (x - x_c)^2 + (y - y_c/2)^2$ and the vortex strength $\sigma = 5$, $x_c = y_c = 2$. The computational domain is given as $[0, 5] \times [0, 5]$. Here γ is 1.4. The initial conditions lead advection of a non-linear vortex at an angle of 45° with the x-axis and the numerical solutions are obtained after $t = 1$. The left and bottom boundaries are constant inflow while right and top boundaries are outflows. The $N_f = 400000$ and $N_{IBC_s} = 10000$. Result at $t = 1$ is compared with the exact solution in Fig. 21. And convergence of L_2 and L_∞ relative errors are presented in Table 4. We show that omitting the boundary conditions in this case can obviously improve the accuracy.

In summary, the PINNs method offers the significant advantage of simplifying the handling of boundary conditions. By not relying on a discrete grid and having the capability to capture temporal behavior, it becomes feasible to omit constant boundary conditions and simplify the setting of outflow boundary conditions. This flexibility and ease of handling boundary conditions make the PINNs method a practical choice for solving hyperbolic equations.

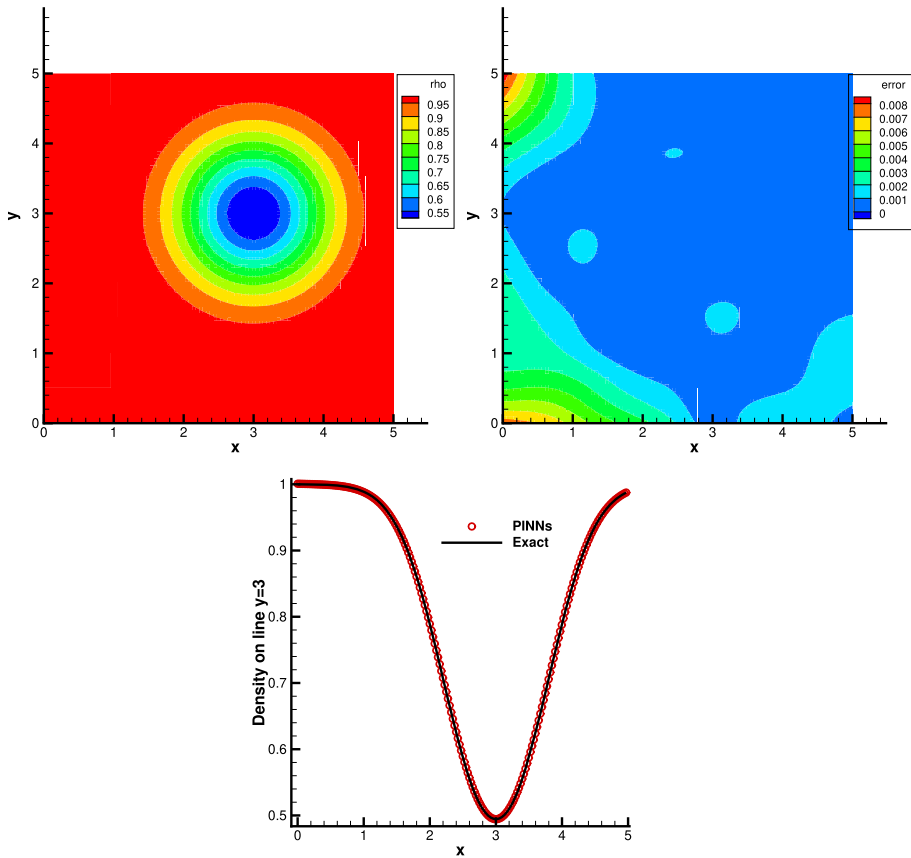


Fig. 21 The result and point-wise error of the 2D vortex evolution problem at $t = 1$

Table 4 Accuracy comparison with and without constant boundary constraint, 2D vortex evolution problem

	L_∞ Error	L_2 Error
Without boundaries	$8.7e-3$	$9.0e-4$
With boundaries	$1.0e-1$	$2.0e-2$

Appendix B: Classical PINNs in Computing Linear and Weak Discontinuities

The effectiveness of Physics-Informed Neural Networks (PINNs) in solving problems with smooth solutions has been extensively studied and demonstrated. However, research on solving problems involving discontinuities is still relatively limited. In this section, we aim to first validate the performance of the classical PINNs approach in solving linear discontinuities and weak discontinuities. Here, we consider several test cases, include linear transport equation problems involving linear discontinuities to verify the solution with linear discontinuities, and one-dimensional Riemann problems with rarefaction waves to verify the solution with derivatives discontinuities.

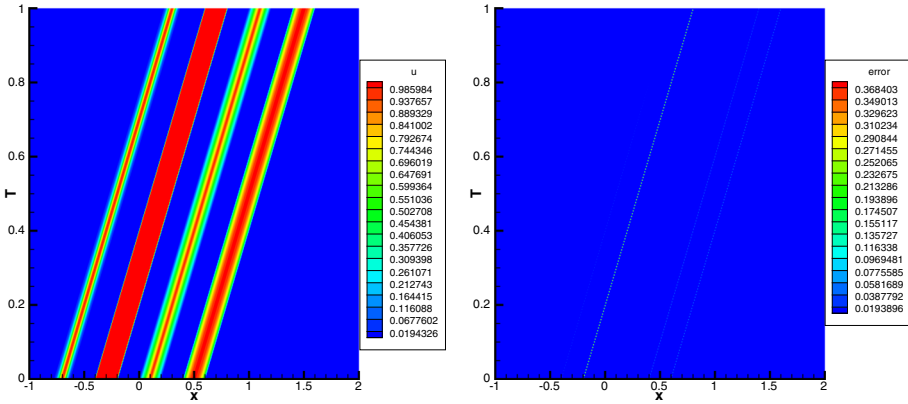


Fig. 22 The result and point-wise absolute error of the 1D linear transport equation with complex waveforms

Appendix B.1: 1D Linear Transport Equation with Complex Waveforms

The equation is given as

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \quad -1 < x < 2,$$

with the initial condition

$$u(x, 0) = \begin{cases} \frac{1}{6}(G(x, \beta, z - \delta) + G(x, \beta, z + \delta) + 4G(x, \beta, z)), & -0.8 \leq x \leq -0.6 \\ 1, & -0.4 \leq x \leq -0.2 \\ 1 - |10(x - 0.1)|, & 0 \leq x \leq 0.2 \\ \frac{1}{6}(F(x, \alpha, \alpha - \delta) + F(x, \alpha, \alpha + \delta) + 4F(x, \alpha, a)), & 0.4 \leq x \leq 0.6 \\ 0, & \text{otherwise} \end{cases}$$

As in Ref. [19], the constants have been assigned specific values: $a = 0.5, z = -0.7, \delta = 0.005, \alpha = 10,$ and $\beta = \log 2/36\delta^2$. The solution exhibits a diverse set of discontinuities, including a smooth combination of Gaussians, a square wave, a sharp triangle wave, and a half ellipse. The equation possesses an exact solution of the form $u(x, t) = u(x - t, 0)$.

Here we use the classical PINNs with the loss function of the mean square error from two part

$$\text{Loss} = \text{MSE}_f + 10\text{MSE}_{IC}$$

and ignoring the boundary conditions as we talked in Appendix A.

The function residual points is taken as $N_f = 10000$ and the initial residual points is $N_{IC} = 1000$, and all they are sampled uniformly.

We use the ADAM optimizer with a learning rate of 0.01 then follows L-BFGS optimizer with a learning rate of 1 until the loss converges.

The solution and the error solved by PINNs are present in Fig.. Then we test the trained network with 100 uniform sampling points at time $t = 0.5$ and compare it we the exact solution. And the L2 relative error is 0.005 (the average of ten cases with different random seeds to eliminate the influence randomness).

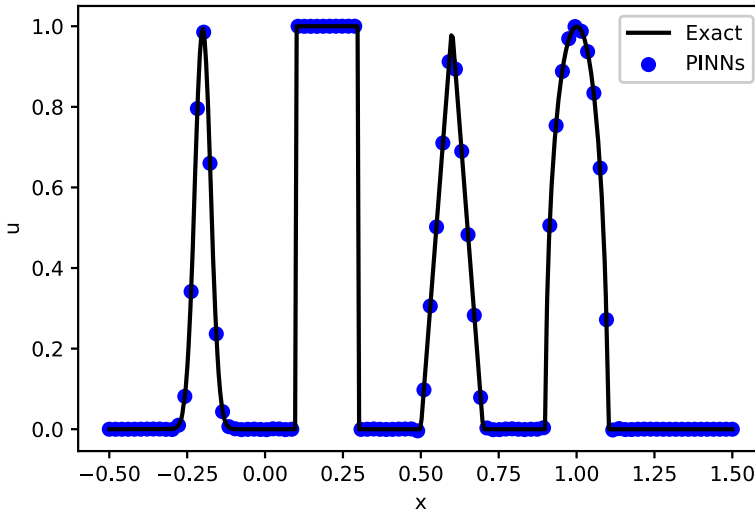


Fig. 23 Test result at $t = 0.5$ of the 1D linear transport equation with complex waveforms

Appendix B.2: 2D Linear Transport Equation with Initial Interface Evolution

Interface tracking is a widely encountered scientific and engineering problem that often requires solving the linear transport equation with a given velocity field:

$$\frac{\partial u}{\partial t} + a_x(t, x, y) \frac{\partial u}{\partial x} + a_y(t, x, y) \frac{\partial u}{\partial y} = 0.$$

We consider a complex case introduced by [28]. The velocity field is given as

$$\begin{cases} a_x = \sin^2(\pi x) \sin(2\pi y), \\ a_y = -\sin^2(\pi y) \sin(2\pi x), \end{cases}$$

when $t < T/2$. Then we take an opposite velocity field to rotate it back. and the initial conditions are

$$u_0(x, y) = \begin{cases} 0, & \sqrt{(x - x_r)^2 + (y - y_r)^2} > 0.15, \\ 1, & \text{else.} \end{cases}$$

The computation domain is $0 \leq x, y \leq 1$.

The numbers of residual points sampled in the computational domain and on the initial condition are $N_f = 10000$ and $N_{IC} = 1000$, respectively. They are sampled with the Latin hypercube sampling (LHS) method.

We have test three networks with different t . After the training, we take two test sets with 10000 uniform points at $T = t/2$ and $T = t/2$ in each case, respectively. The result are shown in Fig. 24. We can see PINNs capture the interface sharply with little dissipation as there is no necessary dissipation introduced to capture the discontinuity. So the possible dissipation is the error from the approximation that can be controlled by the convergence of the loss function.

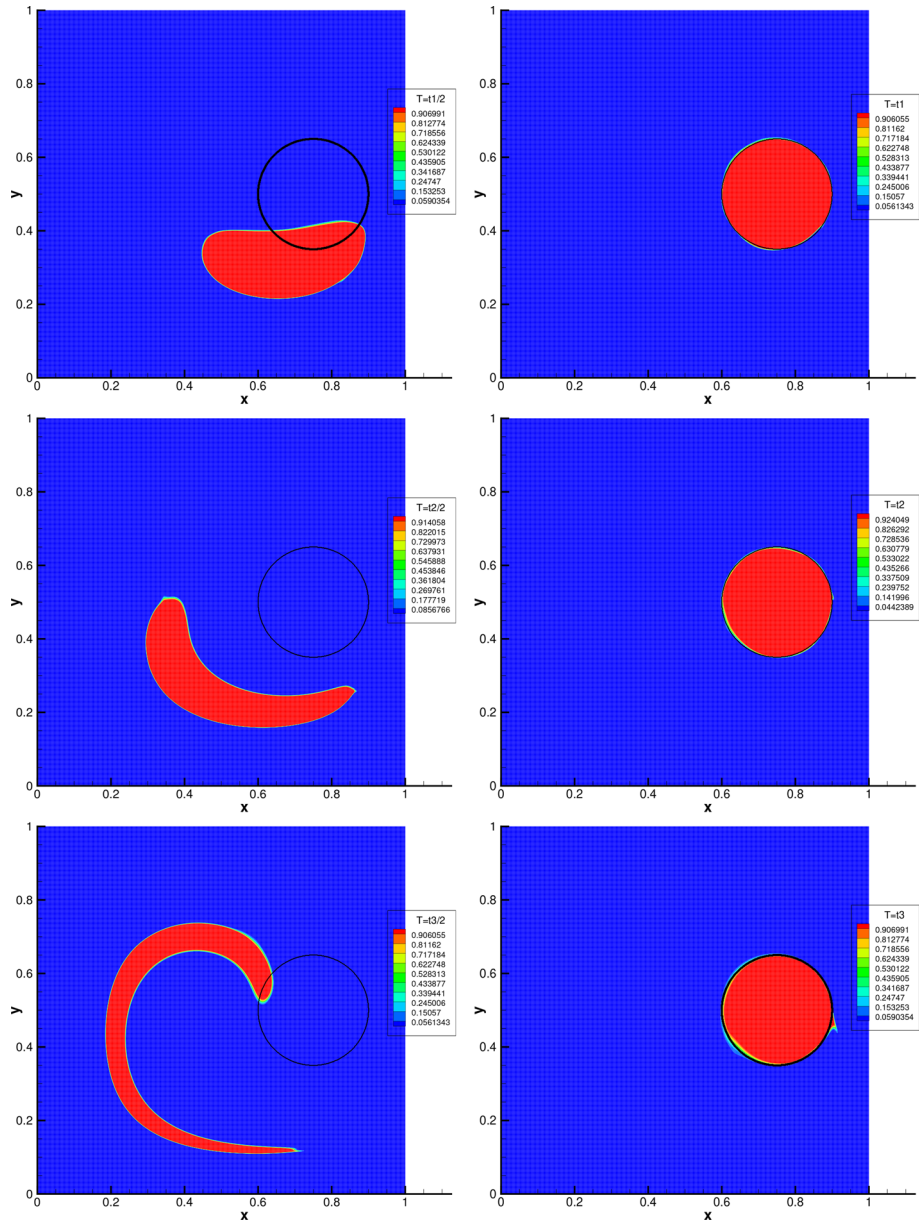


Fig. 24 The result of the 2D linear transport equation with vortex stretching, $t_1 = 0.5$, $t_2 = 1$ and $t_3 = 2$, black line is the initial position of the interface

Appendix B.3: 1D Riemann Problem with Double Rarefaction Waves

After we test two linear discontinuity cases, then we will test the performance of PINNs in solving weak discontinuities. Rarefaction wave is typically one kind of weak discontinuities that they have only derivative discontinuity and C^0 smoothness.

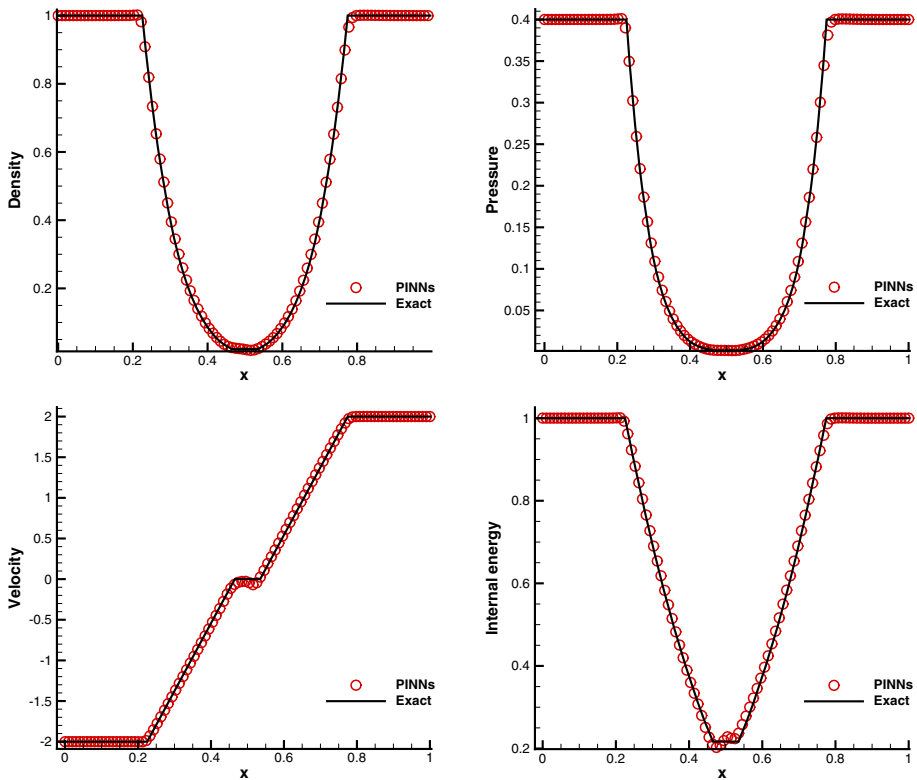


Fig. 25 The result of the double rarefaction problem

We choose a classical double rarefaction problem described by Toro [29], Chapter 6. In this test, the center of the domain is evacuated as two rarefaction waves propagate in each direction, outward from the center.

The initial conditions are: The governing equation is the 1D Euler equation that is given in Sect. 2 with the initial condition as

The initial condition is given as

$$(\rho, u, p) = \begin{cases} (1, -2, 0.4), & \text{if } 0 \leq x \leq 0.5, \\ (1, 2, 0.4), & \text{if } 0.5 < x \leq 1. \end{cases} \tag{B.1}$$

And the computational time is $t = 0.1$. This problem is hard to solve as there are vacuum created at the center after the expansion. Especially the internal energy $e = \frac{p}{\gamma} \rho(\gamma - 1)$ may have large error inside the low pressure/density region.

We use a two step training strategy, the initial residual points $N_{IC} = 100$ and we pre-train the network with $N_f = 2000$ first to convergence and then refine it with more residual points as $N_f = 10000$. The results are shown in Fig. 25. It shows that PINNs performs good in solving rarefaction waves even with vacuum regions.

Then we give a conclusion of appendix B. We test three classical interface tracking cases with the classical PINNs without any optimization. The results show that PINNs is very good at solving linear discontinuous problem, the discontinuous keeps sharp without obviously

dissipation. So PINNs have much potential to be powerful methods for solving multi-materials problems.

References

1. VonNeumann, J., Richtmyer, R.D.: A method for the numerical calculation of hydrodynamic shocks. *J. Appl. Phys.* **21**(3), 232–237 (1950)
2. Harten, A., Engquist, B., Osher, S., Chakravarthy, S.R.: Uniformly high order accurate essentially non-oscillatory schemes, III. In: *Upwind and High-resolution Schemes*, Springer, pp. 218–290 (1987)
3. Jiang, G.-S., Shu, C.-W.: Efficient implementation of weighted ENO schemes. *J. Comput. Phys.* **126**(1), 202–228 (1996)
4. Cockburn, B., Shu, C.-W.: The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Numer. Anal.* **35**(6), 2440–2463 (1998)
5. Pirozzoli, S.: Numerical methods for high-speed flows. *Annu. Rev. Fluid Mech.* **43**, 163–194 (2011)
6. Zhang, D., Jiang, C., Liang, D., Cheng, L.: A review on TVD schemes and a refined flux-limiter for steady-state calculations. *J. Comput. Phys.* **302**, 114–154 (2015)
7. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019)
8. Pang, G., Yang, L., Karniadakis, G.E.: Neural-net-induced Gaussian process regression for function approximation and PDE solution. *J. Comput. Phys.* **384**, 270–288 (2019)
9. Lye, K.O., Mishra, S., Ray, D.: Deep learning observables in computational fluid dynamics. *J. Comput. Phys.* **410**, 109339 (2020)
10. Magiera, J., Ray, D., Hesthaven, J.S., Rohde, C.: Constraint-aware neural networks for Riemann problems. *J. Comput. Phys.* **409**, 109345 (2020)
11. Huang, H., Liu, Y., Yang, V.: Neural networks with local converging inputs (NNLCI) for solving conservation laws, part II: 2D problems, arXiv preprint [arXiv:2204.10424](https://arxiv.org/abs/2204.10424)
12. Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F.: Scientific machine learning through physics-informed neural networks: Where we are and what's next, arXiv preprint [arXiv:2201.05624](https://arxiv.org/abs/2201.05624)
13. Mao, Z., Jagtap, A.D., Karniadakis, G.E.: Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.* **360**, 112789 (2020)
14. Patel, R.G., Manickam, I., Trask, N.A., Wood, M.A., Lee, M., Tomas, I., Cyr, E.C.: Thermodynamically consistent physics-informed neural networks for hyperbolic systems. *J. Comput. Phys.* **449**, 110754 (2022)
15. Jagtap, A.D., Kharazmi, E., Karniadakis, G.E.: Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **365**, 113028 (2020)
16. Jagtap, A.D., Mao, Z., Adams, N., Karniadakis, G.E.: Physics-informed neural networks for inverse problems in supersonic flows, arXiv preprint [arXiv:2202.11821](https://arxiv.org/abs/2202.11821)
17. Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E.: Physics-informed neural networks (PINNs) for fluid mechanics: a review. *Acta. Mech. Sin.* **37**(12), 12 (2021)
18. Papados, A.: Solving hydrodynamic shock-tube problems using weighted physics-informed neural networks with domain extension, <https://doi.org/10.13140/RG.2.2.29724.00642/1>
19. Shen, Y., Liu, L., Yang, Y.: Multistep weighted essentially non-oscillatory scheme. *Int. J. Numer. Meth. Fluids* **75**(4), 231–249 (2014)
20. Yu, J., Lu, L., Meng, X., Karniadakis, G.E.: Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Comput. Methods Appl. Mech. Eng.* **393**, 114823 (2022)
21. Mcclenny, L., Braga-Neto, U.: Self-adaptive physics-informed neural networks using a soft attention mechanism. In: *AAAI-MLPS 2021* (2021)
22. Wang, S., Teng, Y., Perdikaris, P.: Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *Soc. Ind. Appl. Math.* **43**, A3055–A3081 (2021)
23. Dafermos, C.M.: *Hyperbolic Conservation Laws in Continuum Physics*, Hyperbolic Conservation Laws in Continuum Physics. Springer, Berlin (2005)
24. Borges, R., Carmona, M., Costa, B., Don, W.S.: An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. *J. Comput. Phys.* **227**(6), 3191–3211 (2008)
25. Shu, C.-W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes. *J. Comput. Phys.* **77**(2), 439–471 (1988)

26. Kurganov, A., Tadmor, E.: Solution of two-dimensional Riemann problems for gas dynamics without Riemann problem solvers. *Numer. Methods Partial Differ. Equ. Int. J.* **18**(5), 584–608 (2002)
27. Mo, H., Lien, F.-S., Zhang, F., Cronin, D.S.: An immersed boundary method for solving compressible flow with arbitrarily irregular and moving geometry. *Int. J. Numer. Meth. Fluids* **88**(5), 239–263 (2018)
28. Aulisa, E., Manservigi, S., Scardovelli, R.: A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows. *J. Comput. Phys.* **188**(2), 611–639 (2003)
29. Toro, E.F.: *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer, Berlin Heidelberg (2013)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.