Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 360 (2020) 112789
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Physics-informed neural networks for high-speed flows

Zhiping Mao, Ameya D. Jagtap, George Em Karniadakis*

Division of Applied Mathematics, 182 George Street, Brown University, Providence, RI 02912, USA

Received 25 May 2019; received in revised form 16 December 2019; accepted 17 December 2019
Available online 27 December 2019

Abstract

In this work we investigate the possibility of using physics-informed neural networks (PINNs) to approximate the Euler equations that model high-speed aerodynamic flows. In particular, we solve both the forward and inverse problems in one-dimensional and two-dimensional domains. For the forward problem, we utilize the Euler equations and the initial/boundary conditions to formulate the loss function, and solve the one-dimensional Euler equations with smooth solutions and with solutions that have a contact discontinuity as well as a two-dimensional oblique shock wave problem. We demonstrate that we can capture the solutions with only a few scattered points clustered randomly around the discontinuities. For the inverse problem, motivated by mimicking the Schlieren photography experimental technique used traditionally in high-speed aerodynamics, we use the data on density gradient $\nabla\rho(x,t)$, the pressure $p(x^*,t)$ at a specified point $x = x^*$ as well as the conservation laws to infer all states of interest (density, velocity and pressure fields). We present illustrative benchmark examples for both the problem with smooth solutions and Riemann problems (Sod and Lax problems) with PINNs, demonstrating that all inferred states are in good agreement with the reference solutions. Moreover, we show that the choice of the position of the point x^* plays an important role in the learning process. In particular, for the problem with smooth solutions we can randomly choose the position of the point x^* from the computational domain, while for the Sod or Lax problem, we have to choose the position of the point x^* from the domain between the initial discontinuous point and the shock position of the final time. We also solve the inverse problem by combining the aforementioned data and the Euler equations in characteristic form, showing that the results obtained by using the Euler equations in characteristic form are better than that obtained by using the Euler equations in conservative form. Furthermore, we consider another type of inverse problem, specifically, we employ PINNs to learn the value of the parameter γ in the equation of state for the parameterized two-dimensional oblique wave problem by using the given data of the density, velocity and the pressure, and we identify the parameter γ accurately. Taken together, our results demonstrate that in the current form, where the conservation laws are imposed at random points, PINNs are not as accurate as traditional numerical methods for forward problems but they are superior for inverse problems that cannot even be solved with standard techniques.

MSC: 35L65; 65M70; 74S25

Keywords: Euler equations; Machine learning; Neural networks; Conservation laws; Riemann problem; Hidden fluid mechanics

* Corresponding author.
E-mail addresses: zhiping_mao@brown.edu (Z. Mao), ameya_jagtap@brown.edu (A.D. Jagtap), george_karniadakis@brown.edu (G.E. Karniadakis).

https://doi.org/10.1016/j.cma.2019.112789

1. Introduction

The conservation of mass, momentum and energy for compressible flow in the inviscid limit can be modeled by the Euler equations, which can be written in the following conservative form [1–3]:

$$\partial_t U + \nabla \cdot f(U) = 0,\ x \in \Omega \subset \mathbb{R}^d,\ d = 1, 2,\ t \in (0, T], \tag{1.1}$$

where in the one-dimensional case,

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}, \quad f(U) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(\rho E + p) \end{pmatrix},$$

and in the two-dimensional case,

$$U = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{pmatrix}, \quad f = (G_1, G_2),\ \text{with } G_i(U) = \begin{pmatrix} \rho u_i \\ \delta_{i1} p + \rho u_1 u_i \\ \delta_{i2} p + \rho u_2 u_i \\ p u_i + \rho u_i E \end{pmatrix},\ i = 1, 2.$$

Here, ρ is the density, p is the pressure, u is the velocity in one dimension or u_1, u_2 are velocity components in x and y directions in two dimensions, and E is the total energy, δ_{ij} is the Kronecker delta. To close the equations, we need one more equation, i.e., the equation of state describing the relation of the pressure and energy. In this paper, we consider the equation of state for a polytropic gas given by

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho \|\boldsymbol{u}\|^2\right), \tag{1.2}$$

where $\gamma = 1.4$ is the adiabatic index, $\boldsymbol{u} = u$ for the one-dimensional case while $\boldsymbol{u} = (u_1, u_2)$ for the two-dimensional case.

Solutions of these conservation laws often develop discontinuities in some finite time even though the initial conditions may be smooth [4]. In such situation, it is difficult to obtain the analytical solution and thus approximate solutions are sought. Numerical methods are extensively used to solve such equations, targeting non-oscillatory reconstructions near discontinuities. Various numerical methods for conservation laws are available in the literature with early attempts focusing on shock capturing methods [5]. Recently, higher order methods have been developed like the discontinuous Galerkin method [6–10], the essential non-oscillatory method and its variants, the weighted essentially non-oscillatory methods [11,12], the spectral difference methods [13], etc. For more details about numerical methods for conservation laws, see [5] and a summary of recent developments for high-speed compressible flow simulations is presented in [14,15]. However, instead of using the classical numerical schemes, alternative approaches have emerged recently using machine learning algorithms to solve forward and inverse problems for partial differential equations (PDEs), such as Navier–Stokes equation, see [16–20] and references therein. Specifically, a physics-informed neural network (PINN) was proposed by Raissi et al. in [17]. More extensions can be found in [21] for fractional diffusion equation, in [22] for stochastic differential equations, and in [23] using deep neural networks trained by multi-fidelity data. For the classical numerical methods, it is necessary to carefully design stable or structure preserving schemes to prevent the blow up of the numerical solutions or to avoid the non-executability of the numerical schemes [24–27]. However, by using the deep neural network (DNN) approximation, the DNN is able to get a relatively stable solution without any regularization [28]. Moreover, the structure preserving of the solutions can be easily satisfied, for instance, we can *simply* apply an exponential function to the output of the NN to ensure the positivity of the solutions. Another promising application using PINN is the hidden fluid mechanics (HFM), which takes advantage of the physics-informed deep learning framework to infer hidden quantities of interest such as velocity and pressure fields in fluid flows by using only a small data set of auxiliary variables. For example, in [29], HFM is proposed to infer the velocity and pressure fields from spatio-temporal visualizations of a passive scaler by coupling the advection–diffusion equation with the Navier–Stokes equation.

The essential idea of PINNs is to infer the unknown solution (physics of interest) by combining the governing equation and the given data (e.g., initial/boundary conditions or partial and scattered measurements of the any of the states). In the present work, we aim to use PINNs to learn the states of interest for high-speed flows, namely, we shall solve the forward/inverse problems of conservation laws governed by the Euler equations (1.1). Specifically,

we shall learn the density, velocity and pressure fields by using the Euler equations and additional given data, which might be the initial/boundary conditions (IC/BCs) or other available data, such as data from experiments. First, we solve the one-dimensional Euler equations with smooth solutions and with solutions that have a contact discontinuity as well as a two-dimensional oblique shock wave problem using the IC/BCs. This is the so called forward problem. Subsequently, we consider the inverse problems using PINNs in Section 4, where we do not use any IC/BCs. Without using IC/BCs, an interesting and promising inverse problem set up is to predict the hidden fluid mechanics by using partially given data (that may be easily obtained experimentally). In the numerical computation of the high speed flow, it is usually not straight-forward to determine the boundary conditions [30–32]. Motivated by mimicking the Schlieren photography experimental technique used traditionally in high-speed aerodynamics, where we can obtain data on the density gradient, i.e., $\nabla\rho(x,t)$, we aim to infer all states of interest (density, velocity and pressure) and investigate what is the minimum number of additional measurements required, i.e., on the pressure $p(x^*,t)$. Additionally, we incorporate other invariants known *a priori*, e.g., global conservation of mass and momentum. Specifically, for the problem with smooth solutions and the Riemann problems (Sod and Lax), we infer all states by using the density gradient $\nabla\rho(x,t)$, the single point value of the pressure $p(x^*,t)$ at a given point $x=x^*$ and the global conservation of mass and momentum. We provide a description on how to choose the suitable data for the inverse problem. In particular, for the problem with smooth solutions we can randomly choose the position of the point x^* from the computational domain, while for the Sod or Lax problem, we choose the position of the point x^* from the domain between the initial discontinuous point and the shock position at the final time. Moreover, we consider the case of using the Euler equations in characteristic form for the inverse problem. Furthermore, we solve another kind of inverse problem for the two-dimensional oblique shock wave problem, that is we learn the equation of state of the Euler equations by using the data of density, velocity and pressure. A summary and conclusions are given in Section 5. We also include an appendix, where we provide a pedagogical example with many details for the interested readers. In this paper, we consider the aforementioned one-dimensional benchmarks to illustrate the main ideas of the new approach and identify possible technical issues. This is similar to previous literature [33–40], where even for these more mature methods the same one-dimensional benchmarks have been used in the past.

2. Methodology

For a compressible inviscid flow governed by the Euler equations (1.1), we assume that there is a set of data related to the states U (density, velocity and pressure), e.g., IC/BCs or experimental data. By using such data and combining them with the Euler equations (1.1), we would like to design PINNs to infer all the states of interest, i.e., the density $\rho(x,t)$, the velocity $\boldsymbol{u}(x,t)$ and the pressure $p(x,t)$. The basic architecture of a PINN mainly contains two NNs sharing hyper-parameters and both contributing to the loss function, one is the uninformed NN associated with the states U while the other one is the informed one associated with the conservation laws, see Fig. 1 for the schematic of the PINN for the Euler equations. In the present work, we shall learn the density, velocity and pressure fields by using a deep neural network, namely, we shall approximate the density, velocity and pressure using the neural network denoted by $U_{NN}(x,t)$.

Let

$$F(x,t)=\partial_t U_{NN}(x,t)+\nabla\cdot f(U_{NN}(x,t)). \tag{2.1}$$

In the training process of PINN, we minimize the following loss function

$$Loss=Loss_{Data}+Loss_F, \tag{2.2}$$

where $Loss_{Data}$ and $Loss_F$ are loss functions corresponding to the given data and Eq. (2.1), respectively, to learn the shared hyper-parameters of the neural network. In this work, the loss function $Loss_F$ is given by the mean square error loss for the function F:

$$Loss_F=MSE_F=\frac{1}{N_F}\sum_{j=1}^{N_F}|F(x_j^F,t_j^F)|^2, \tag{2.3}$$

where $\{x_j^F,t_j^F\}_{j=1}^{N_F}$ are the training points with N_F being the number of points for evaluating $F(x,t)$. The loss function $Loss_{Data}$ is also given by the mean square errors evaluated on different training points as we will specify later for different problems. To differentiate the neural networks and their compositions, we use the automatic

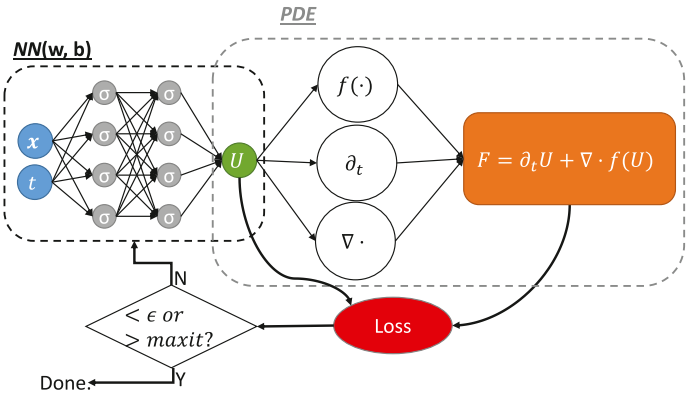

Fig. 1. Schematic of PINN for the Euler equations. The left NN is the uninformed while the right one induced by the conservation law is the informed one. The two NNs share hyper-parameters and they both contribute to the loss function.

differentiation applying the chain rule [41]. This is the same method used to back-propagate the errors in a neural network and hence we do not need any special grids to compute the derivatives as in classical numerical methods.

In PINNs, we have found empirically that we can approximate PDEs even when we do not have proper IC/BC but instead some other measurements somewhere in the domain. This naturally raises the question on how much data is sufficient to produce a unique field of all states for the compressible flow. On the one hand, we know that the Euler equations are well-posed with suitable IC/BCs and the problem can be solved by a number of numerical methods. So the IC/BCs can also be used to solve the corresponding PDEs with PINN by taking the IC/BCs as data. We will refer to solving PDEs with IC/BCs using PINN as the forward problem [17]. In the present work, we solve the forward problem for the Euler equations in Section 3. On the other hand, what if the IC/BCs are not known? If we only have partial data associated with the density, velocity or pressure, can we predict the whole field for all states of interest. Our simulation tests suggest that this is true for many cases. As shown later in Section 4, we can infer all states (ρ, u and p) by using the gradient of the density $\nabla\rho(x,t)$, the value of the pressure $p_*(t) := p(x^*,t)$ at a given point $x = x^*$ and approximating the conservation laws by neural networks. We will refer to this as the inverse problem in the paper. Another kind of inverse problem is to learn the parameterized equations using the given data, e.g. the equation of state.

3. Forward problems

Typically, the Euler equations can be uniquely solved by using IC/BCs, namely, when the forward problem is wellposed. Thus, in this section, we aim to obtain the neural network approximations of the density, velocity and pressure by using IC/BCs, i.e., we solve the forward problem for the Euler equations.

By taking the IC/BCs as data, we have the loss function (2.2) for the forward problem given by

$$Loss = Loss_{Data} + Loss_F = MSE_{IC} + MSE_{BC} + MSE_F, \tag{3.1}$$

where MSE_F is given by (2.3), MSE_{IC} and MSE_{BC} are the mean square errors corresponding to the initial and boundary conditions.

3.1. One-dimensional problems

In this subsection, we consider the one-dimensional forward problem. First, we include in Appendix A a pedagogical example by considering Eq. (1.1) for smooth solutions; we obtain accurate solutions with relative error of the order of $O(10^{-5})$. We provide all the details for this simple example so that the interested reader can repeat our results.

In addition to the smooth solution problem, we present in the following a numerical example for the forward problem in which the solution has a moving contact discontinuity.

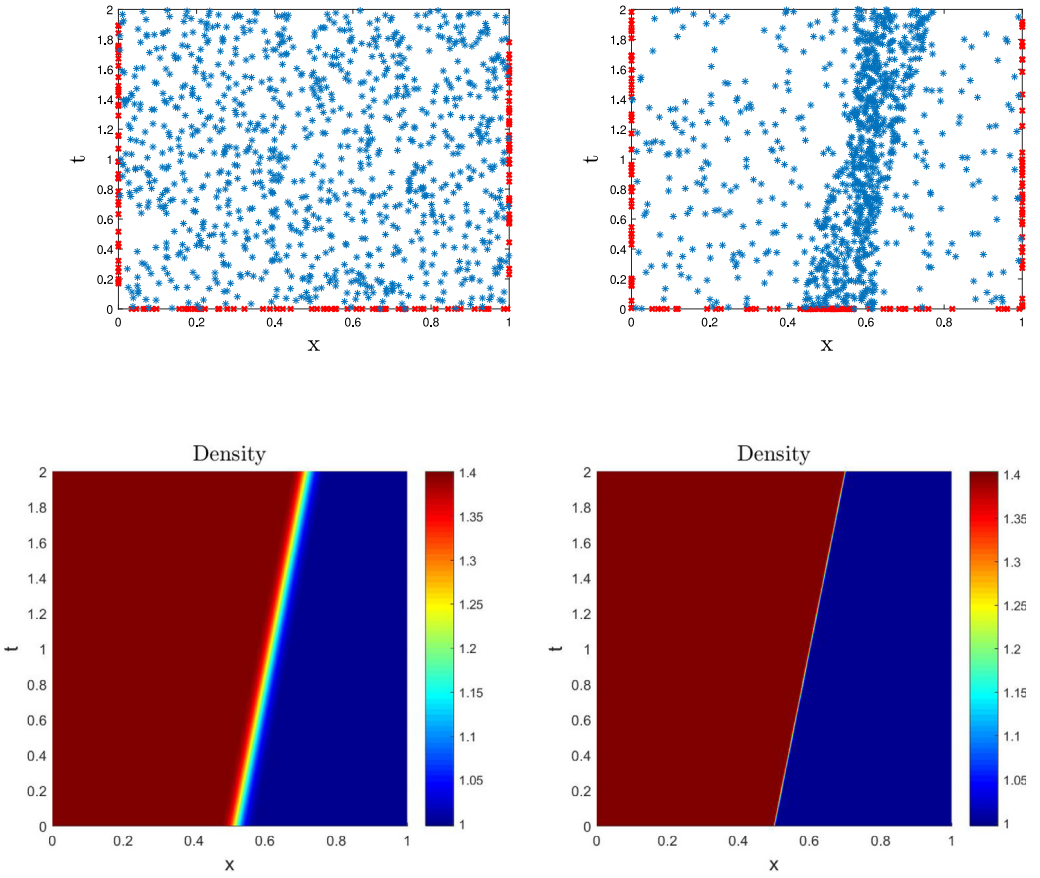

Fig. 2. Forward problem of Example 1: Upper: Distributions of the randomly distributed training and residual points (left) and clustered training and residual points (right). The blue star points are the residual points inside the domain corresponding to the loss associated with the Euler equations, while the red cross points are training points corresponding to the loss associated with IC/BCs. Lower: PINN solutions of the density in the $x-t$ domain obtained by using the loss function (3.1) with randomly distributed training and residual points (left) and clustered training and residual points (right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Example 1. Consider the one-dimensional Euler equation (1.1) with a moving contact discontinuity. The computational domain is [0, 1]. An initial shock is given at $x = 0.5$, and the left and right states are given by

$$(\rho_L, u_L, p_L) = (1.4, 0.1, 1.0), \quad (\rho_R, u_R, p_R) = (1.0, 0.1, 1.0).$$

Dirichlet boundary conditions are used. The exact solutions are given by

$$\rho(x,t) = \begin{cases} 1.4, & x < 0.5 + 0.1t, \\ 1.0, & x > 0.5 + 0.1t, \end{cases} \quad u(x,t) = 0.1, \quad p(x,t) = 1.0.$$

In this case, the mean square errors corresponding to the BCs are as follows:

$$MSE_{BC} = \frac{1}{N_b} \sum_{j=1}^{N_b} \left(|U_{NN}(0, t_j^{BC}) - U(0, t_j^{BC})|^2 + |U_{NN}(1, t_j^{BC}) - U(1, t_j^{BC})|^2 \right).$$

Here we use $N_{BC} = 60$ random distributed training points for the boundaries. The number of training points for the IC and the function F are $N_{IC} = 60$ and $N_F = 1000$, respectively. We test two different sets of training points: randomly distributed points and clustered points. For the clustered points, we have more points around the discontinuous region, see the upper plots of Fig. 2. The search for an efficient NN architecture characterized by its width, depth, activation functions, learning rate, optimizer *etc*, is one of the emerging topics in machine learning

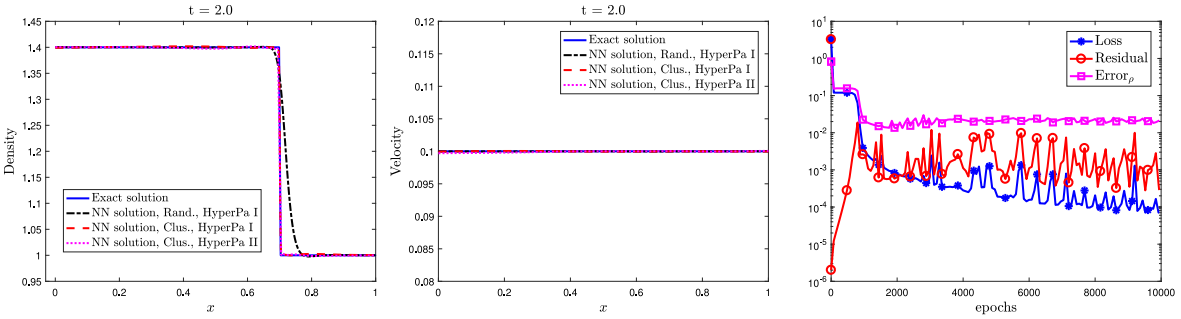

Fig. 3. Effect of the architecture for the forward problem of Example 1: (a)–(b) Comparison of the PINN solutions at time $t = 2$ of the density (left), velocity (middle) obtained by randomly distributed points and clustered points with loss function (3.1) and different types of hyper-parameters. $N_F = 1000$, Rand. means randomly distributed points while Clus. means clustered points. HyperPa I: 7 hidden layers with 20 neurons in each layer, training with 10 000 steps for Adam and 2000 steps for L-BFGS-B; HyperPa II: 4 hidden layers with 40 neurons in each layer, training with 20 000 steps for Adam and 3000 steps for L-BFGS-B. The plot of the pressure at time $t = 2.0$ that is not shown here is almost close to the exact solution (see also the relative error in Table 1). (c) Loss, residual and the error of the density against the epochs using clustered distributed points and HyperPa I. Here "epochs" refers to the number of steps that the Adam optimizer performs.

research, see [42–47] and the references therein. We test two types of neural networks with different sets of hyper-parameters. The first test has the following set of hyper-parameters denoted by HyperPa I: 7 hidden layers with 20 neurons in each layer, and the optimizing procedure is the Adam optimizer with an initial learning rate 0.001 for 10 000 steps followed by the L-BFGS-B optimizer with 2000 steps. The second test has the set of hyper-parameters denoted by HyperPa II as follows: 4 hidden layers with 40 neurons in each layer, and the optimizing procedure is the Adam optimizer with an initial learning rate 0.001 for 20 000 steps followed by the L-BFGS-B optimizer with 3000 steps. We use the function tanh as the activation function, which will be used in all the simulation tests in the present work.

The results of the density in the $x - t$ plane for both types of training and residual points with HyperPa I are shown in the lower plots of Fig. 2. Furthermore, we plot the snapshots of the density, velocity and pressure at time $t = 1.0$ and $t = 2.0$ in Fig. 3. Observe that the result obtained by using the randomly distributed points is diffusive while the one obtained by the clustered points with either set of hyper-parameter captures the discontinuity very well. Moreover, we can see that we do not need to know the exact location of the discontinuity, we only require a rough estimate of the region of discontinuity and use more scattered data around the region of discontinuity.

In addition, we plot the loss, the residual of the equation and the L_2 relative error of the density at time $t = 2$, denoted by $\text{Error}_\rho := \|\rho(x, 2) - \rho_{NN}(x, 2)\|_{L^2} / \|\rho(x, 2)\|_{L^2}$, for Example 1 in Fig. 3 (see the lower right plot). We point out here that the residual of the equation is computed as follows:

$$\text{Residual} := \sqrt{\frac{\sum_{i=1}^{N_g} F^2(x_i, t_i)}{3N_g}}, \tag{3.2}$$

where points $\{x_i, t_i\}_{i=1}^{N_g}$ are the uniformly spaced grid points in the domain Ω with $N_g = |\Omega|/\delta x \delta t$, where $\delta x = \delta t = 0.01$ are the space and time step sizes. If not specified, we would like to use this formula to compute the residual of the equation in this work. We observe that the L_2 relative error of the density decays similarly as that of the loss while the residual of the equation decays fast at first and then increases and then decays again. Moreover, we present the L_2 relative errors for the density, velocity and pressure in Table 1 showing PINN solutions with relative error of the order of $O(10^{-2})$ for density and of almost $O(10^{-5})$ for velocity and pressure. The accuracy for the velocity and pressure is much higher than that for the density, which is due to the fact the velocity and pressure are smooth while the density has a contact discontinuity.

Furthermore, we test two different numbers of residual points (corresponding to the equations), i.e., N_F, for the clustered distributed points with HyperPa I, where we keep dense points around the discontinuous region and use

Table 1
Forward problem of Example 1: Relative L_2 error for the density, velocity and pressure using clustered training points.

State variable	Density	Velocity	Pressure
Relative L_2 error	1.711122e−02	5.196859e−05	3.621883e−06

much less points in the smooth region. The result is shown in Fig. 4. Observe that we can also get good prediction with about half number of the residual points compared to the residual points used previously. This means that in this case we can approximate the solution in the constant region with a few points.

3.2. Two-dimensional steady state problem

We now consider a two-dimensional steady state forward problem, i.e., the forward oblique shock wave problem.

Example 2. In this steady state problem, the computational domain is $[0,\ 1]^2$. An inviscid Mach 2 flow is at an angle of -10° with respect to the horizontal wall, which generates an oblique shock with shock angle 29.3° with the wall. Dirichlet boundary conditions are applied at the left and top boundaries, whereas all primitive variables are extrapolated at the right outgoing boundary. Wall boundary conditions are prescribed at the bottom boundary. The solution for this problem is given by

$$(M,\ \rho,\ u_1,\ u_2, p) = \begin{cases} (2.0,\ 1.0,\ \cos 10^\circ,\ -\sin 10^\circ,\ 0.17857) & \text{before shock,} \\ (1.6405,\ 1.4584,\ 0.8873,\ 0.0,\ 0.3047) & \text{after shock.} \end{cases} \tag{3.3}$$

The loss function for the problem is given by

$$Loss = Loss_{BC} + Loss_F = MSE_{BC} + MSE_F, \tag{3.4}$$

where MSE_F is given by (2.3) and

$$MSE_{BC} = MSE_\rho^D + MSE_p^D + MSE_{\boldsymbol{u}}^D + MSE_\rho^N + MSE_p^N + MSE_{\boldsymbol{u}}^N$$

with

$$\begin{aligned} MSE_\rho^D &= \frac{1}{N_u}\sum_{i=1}^{N_u} |\rho^i - \rho_{NN}(x_u^i, y_u^i)|^2, \qquad MSE_\rho^N = \frac{1}{N_u}\sum_{i=1}^{N_u} |\nabla\rho_{NN}(x_u^i, y_u^i)|^2 \\ MSE_p^D &= \frac{1}{N_u}\sum_{i=1}^{N_u} |p^i - p_{NN}(x_u^i, y_u^i)|^2, \qquad MSE_p^N = \frac{1}{N_u}\sum_{i=1}^{N_u} |\nabla p_{NN}(x_u^i, y_u^i)|^2 \\ MSE_{\boldsymbol{u}}^D &= \frac{1}{N_u}\sum_{i=1}^{N_u} |\boldsymbol{u}^i - \boldsymbol{u}_{NN}(x_u^i, y_u^i)|^2, \qquad MSE_{\boldsymbol{u}}^N = \frac{1}{N_u}\sum_{i=1}^{N_u} |\nabla\boldsymbol{u}_{NN}(x_u^i, y_u^i)|^2. \end{aligned} \tag{3.5}$$

Here $\{x_u^i, y_u^i, \alpha^i\}_{i=1}^{N_u}$, where $\alpha = \rho, u_1, u_2, p$, denotes the boundary training data on primitive variables like density, velocities in x and y directions, and pressure, respectively. It is important to note that the MSE for Dirichlet and Neumann boundaries are used only where these boundary conditions are prescribed in the domain.

We show in the upper plots of Fig. 5 the randomly distributed training points in the domain (left) as well as the clustered training points near the shock (right) while in the lower plots of Fig. 5 the density contours with these two types of training points. These solutions are compared with the exact solution at various x locations shown in Fig. 6. In case of the clustered points set, the number of training points is higher near the shock wave whereas they are sparse away from it. Compared with the PINN solutions obtained by using the random training points, the shock position is much more accurately captured by the PINN solutions by using the clustered training points, even with much less points. Hereafter, we shall use the clustered training points for the PINN simulations.

We show in Fig. 7 the contours of pressure and velocities in x and y directions whereas in Fig. 8 the comparisons of PINN solutions with exact solutions at various x locations. Moreover, we show the loss, the residual of the

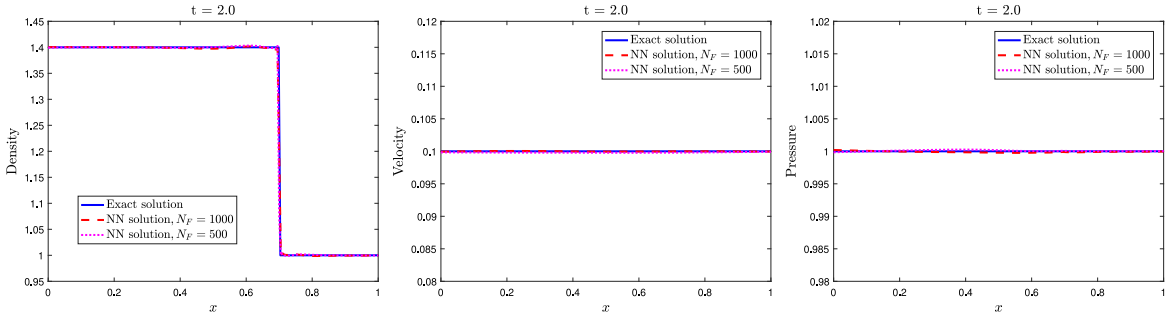

Fig. 4. Effect of the number of the residual points (corresponding to the equation) for the forward problem of Example 1: Comparison of the PINN solutions of the density (left), velocity (middle) and pressure (right) at time $t = 2$ obtained by different numbers of clustered residual points. Here HyperPa I and loss function (3.1) are used. The number of boundary points and initial points (training points) are $N_{BC} = 60,\ N_{IC} = 60$.

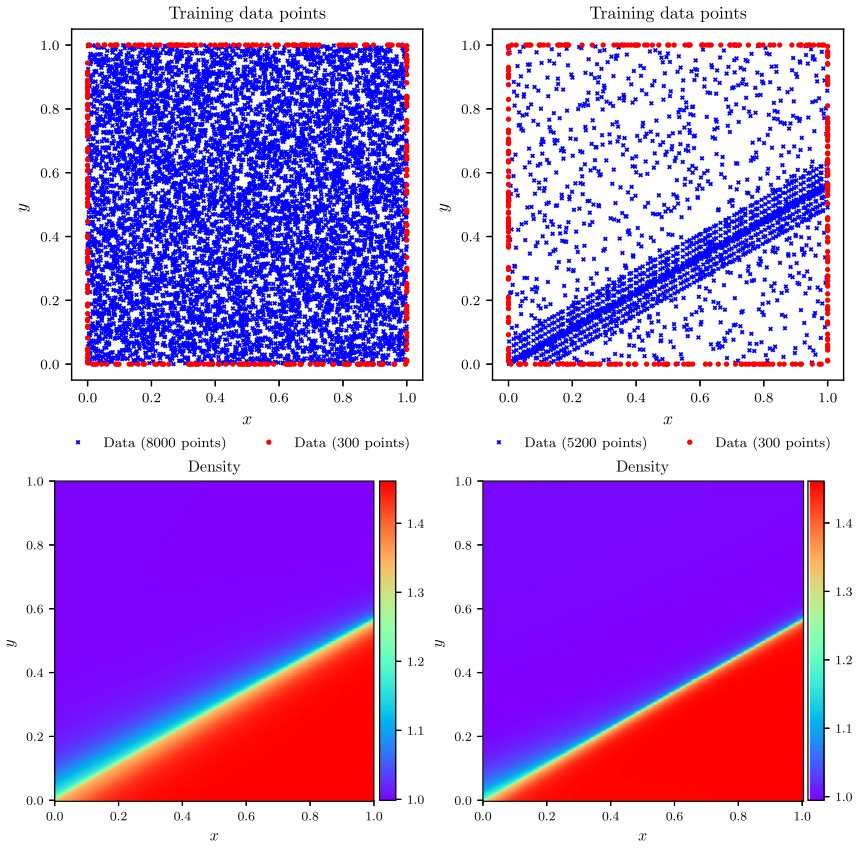

Fig. 5. Forward problem of Example 2, i.e., the forward oblique shock wave problem: Upper: Distributions of the randomly distributed training points (left) and clustered training points (right). The blue cross points are the training points inside the domain corresponding to the loss associated with the Euler equations, while the red dot points are training data corresponding to the loss associated with BCs. Lower: PINN solutions of the density in the $x - y$ domain obtained by using the loss function (3.4) with randomly distributed training points (left) and clustered training points (right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

equation and the L^2 relative error of the density in Fig. 9. The residual of the equation is computed as follows:

$$\text{Residual} := \sqrt{\frac{\sum_{i=1}^{N_F} F^2(x_i, y_i)}{4N_F}},$$

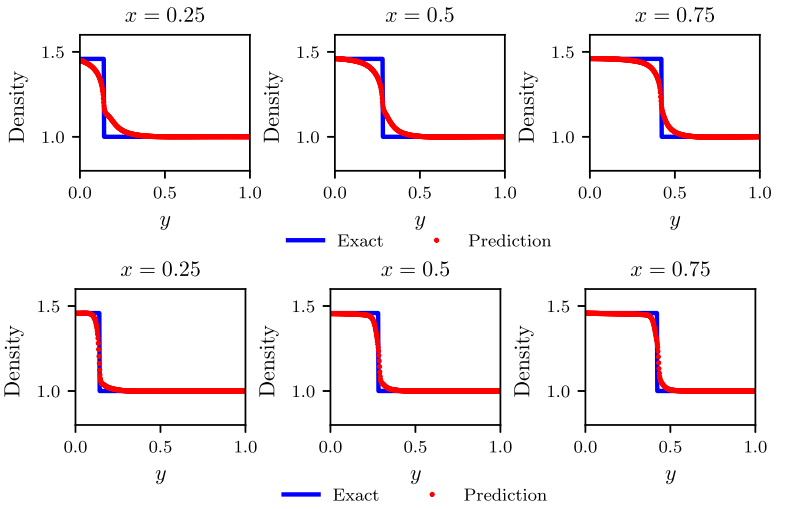

Fig. 6. Forward problem of Example 2, i.e., the forward oblique shock wave problem: Comparison of density with exact solution at various x locations using randomly distributed training points (upper) and clustered training points (lower) and loss function (3.4).

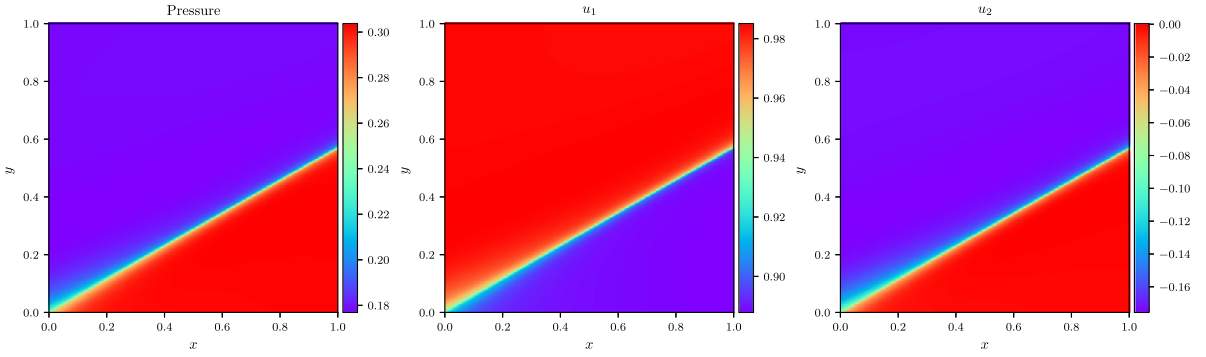

Fig. 7. Forward problem of Example 2, i.e., the forward oblique shock wave problem: PINN solutions of pressure (left) and velocity in x-direction (middle) and y-direction (right) using clustered training points.

Table 2
Forward problem of Example 2, i.e., the forward oblique shock wave problem: Relative L_2 error for various primitive variables using the clustered training points.

State variable	Density	Pressure	Velocity X-dir	Velocity Y-dir
Relative L_2 error	8.23482e−03	2.43650e−03	2.35356e−02	1.04430e−02

where points $\{x_i, y_i\}_{i=1}^{N_F}$ are the training points for evaluating F. We also present in Table 2 the relative L_2 error for density, pressure and velocities in x and y directions. For the two-dimensional problem the accuracy of the velocity and pressure is worse than the one-dimensional case in Table 1 because the velocity and pressure in the one-dimensional case are smooth while the velocity and pressure in the two-dimensional case have discontinuities.

4. Inverse problems

In this section, we turn our attention to inverse problems. Specifically, we aim to solve two types of inverse problems: First, we are going to learn the density, velocity and pressure fields over the domain Ω by using data on the density gradient, i.e., $\nabla\rho(x, t)$, somehow mimicking the Schlieren photography experimental technique used

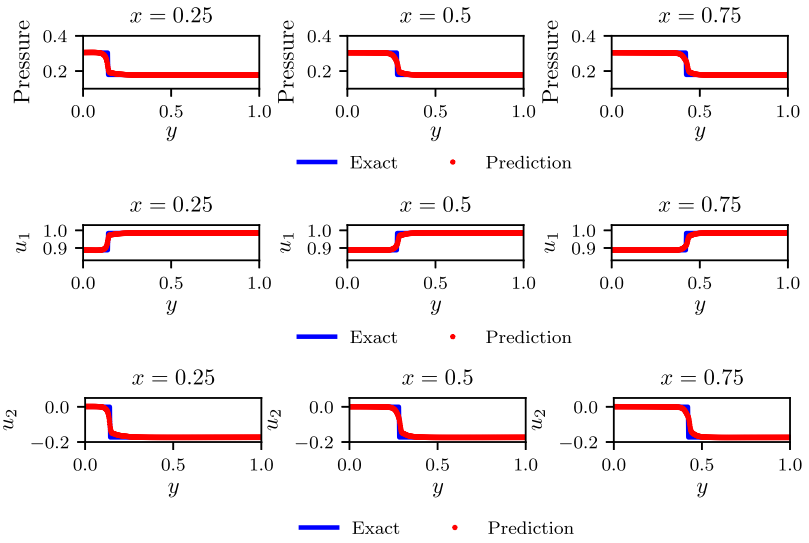

Fig. 8. Forward problem of Example 2, i.e., the forward oblique shock wave problem: Comparison of pressure and velocity fields with exact solution at various x locations using the clustered training points.

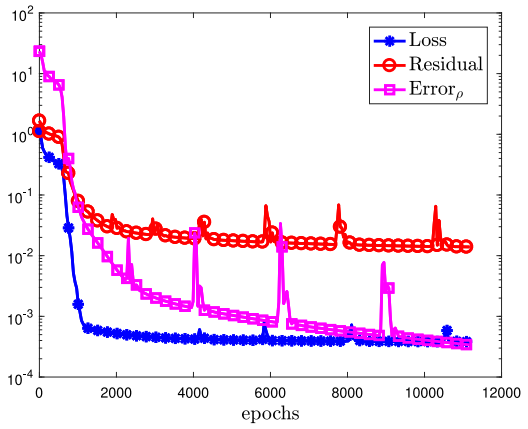

Fig. 9. Forward problem of Example 2, i.e., the forward oblique shock wave problem: Loss and the relative L^2 error of the density against the epochs using the clustered training points. Here "epochs" refers to the number of steps that the Adam optimizer performs.

traditionally in high-speed aerodynamics. In particular, we do not use the initial or the boundary conditions and instead we investigate what is the minimum number of additional measurements required, i.e., on the pressure $p(x^*, t)$. Additionally, we incorporate other invariants known *a priori*, e.g., global conservation of mass and momentum. Second, we are going to learn the equation of state (EOS) from given data for the two-dimensional oblique shock wave problem using measurements of the density, pressure and velocity fields.

4.1. One-dimensional problems

First we solve the inverse smooth problem in Appendix B, as a pedagogical example, where given the density gradient we learn all the fields using the additional constraint of the global mass conservation at the initial time and a single point measurement for the pressure field at all times; the accuracy of the results is of the order of $O(10^{-4})$.

In the following we present results for two well-known benchmarks, namely, Sod's shock tube problem [48] as well as Lax's shock tube problem [49].

4.1.1. Sod problem

Example 3. Consider the following Riemann problem subject to the following initial condition

$$U(x,0) = U_0(x) = \begin{cases} U_L, & x < 0, \\ U_R, & x > 0 \end{cases} \tag{4.1}$$

with

$$U_L = (\rho_L, v_L, p_L)^T = (1, 0, 1)^T, \quad U_R = (\rho_R, v_R, p_R)^T = (0.125, 0, 0.10)^T.$$

The computational domain and the time interval are set to $(x, t) \in [-2.5, 2.5] \times [0, 1]$.

To start with, we use the same setup for the data as the one used in Appendix B in which case we have smooth solutions and obtain good predictions for all states by using the data on the density gradient $\nabla\rho(x,t)$, the pressure $p(x^*,t)$ as well as the global conservation of mass. Furthermore, for the Sod problem, we use a weighted loss function, i.e.,

$$Loss = \omega_{\nabla\rho} MSE_{\nabla\rho} + \omega_p MSE_{p^*} + \omega_m MSE_{Mass_0} + \omega_F MSE_F \tag{4.2}$$

by assigning weights to each term of the loss function. Moreover, for the shock tube problem (Sod problem or Lax problem considered below), since we do not have the first-order derivatives in the strong sense at the discontinuous points, we would like to use the finite difference to represent the differentiation for the density gradient. In particular, we let

$$MSE_{\nabla\rho} = \frac{1}{N_{\nabla\rho}} \sum_{j=1}^{N_{\nabla\rho}} \left| \frac{\rho(x_j^{\nabla\rho} + dx, t_j^{\nabla\rho}) - \rho(x_j^{\nabla\rho}, t_j^{\nabla\rho})}{dx} - \frac{\rho_{NN}(x_j^{\nabla\rho} + dx, t_j^{\nabla\rho}) - \rho_{NN}(x_j^{\nabla\rho}, t_j^{\nabla\rho})}{dx} \right|^2,$$

where dx is the space step size to be given. We set $dx = 0.008$ in all the tests for the Sod problem.

In all the simulations carried out in this subsection, we set $\omega_{\nabla\rho} = \omega_F = 0.1$ and $\omega_p = \omega_m = 1.0$. Moreover, for the distribution of the training points, we adopt the clustered training points, which means that we use more points around the discontinuous regions to compute the residuals while we use less points around the smooth regions; see Fig. 10 for the distribution of the training data. The total number of the training data is $N_{\nabla\rho} = 2334$ for $\nabla\rho$, and $N_F = 3338$ for F. The total number of the training data for the pressure is $N_{p^*} = 200$ that is uniformly distributed in the time domain $[0, 1]$. A comparison between the results obtained by using different types of data will be presented in Section 4.1.3 later. We point out here that in the computation, instead of using single precision, we shall use *double precision* for the shock tube problem. A comparison between the results obtained by using single precision and double precision will be presented in Section 4.1.4 later. We also point out here that, if not specified, we shall use a neural network that has 4 hidden layers with 120 neurons at each layer and train all models with the Adam optimizer with an initial learning rate of 0.0005 for 8000 steps followed by a L-BFGS-B optimizer with 200 000 steps. The training process would stop if the relative difference between two neighboring training steps is less than $\epsilon = 10^{-16}$.

4.1.2. Influence of the position of x^* of the pressure probe

As we mentioned for the case of the Sod problem with smooth solutions (see Appendix B), the choice of the position of the point x^* (at which the data of the pressure would be provided) would effect the learning process depending on the specific physical problem. For the problem with smooth solutions considered in Appendix B, we can randomly choose the position of the point x^* from the computational domain Ω, however, this is not the case for the Sod problem. In particular, we present the following test to illustrate the failure by specifying the position of x^* to be the right boundary, i.e., $x^* = 2.5$, and employing the loss function (4.2).

The results for the inference of the density, velocity and pressure at time $t = 1.0$ are shown in Fig. 11. We can see that although we can obtain a good solution for the density, we cannot obtain good predictions for the velocity and the pressure. It seems that since the flow has not yet reached the region between the shock and the right boundary, combining the information of the pressure in this region with the data and equations provided would not reveal the true fields of the pressure and velocity in the domain of interest.

Now let us choose $x^* = 0.6$ and use the neural network that has the same architecture as the aforementioned one. The results for the density, velocity and pressure at time $t = 1.0$ are shown in Fig. 12, (see the results corresponding

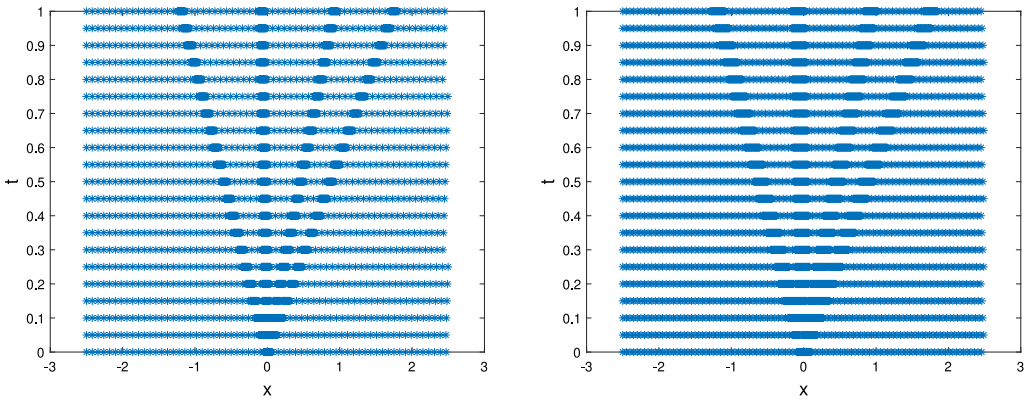

Fig. 10. Inverse problem of Example 3, i.e., the inverse Sod problem: Distribution of the clustered training points. We use more points around the discontinuous regions while use less point in the smooth regions. Left: distribution of training data for the density gradient; Right: Distribution of the training points for F, i.e., the Euler equations.

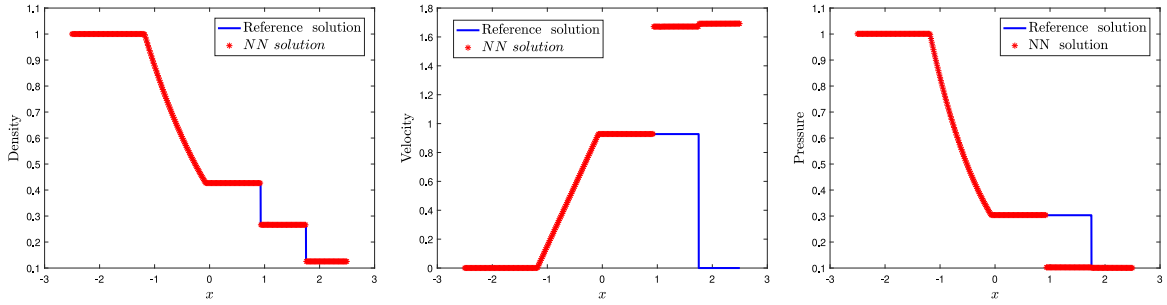

Fig. 11. Inverse problem of Example 3, i.e., the inverse Sod problem, using the data shown in Fig. 10: PINN solutions of the density (left), velocity (middle) and pressure (right) obtained by using the loss function (4.2). Given also is information from a pressure probe located at the right boundary, i.e., $x^* = 2.5$.

to red star points). Observe that unlike the case considered above, we now have our predictions of the density and pressure to be in good agreement with the exact solutions. However, for the velocity, we can infer the corresponding field before the shock (see the middle plot of Fig. 12), but in the region between the shock and the right boundary, there is a gap between the exact solution and the neural network solution. In this case we do not use any data related to the velocity, so we can infer the velocity only by using the Euler equations and the data related to the density and the pressure. Therefore, we can predict the velocity if there exists state changing or otherwise, for instance, steady state (the velocity is equal to 0), we cannot capture the velocity.

To overcome the aforementioned issue, one more condition related to the velocity should be used, and we would like to utilize the "regional" momentum conservation. In particular, assuming that we have *constant* inputs and outputs over a domain $(x_1, x_2) \subset (a, b)$ for $t \in [0, T]$, by integrating equation (1.1) over the domain (x_1, x_2), we have

$$\frac{d}{dt}\int_{x_1}^{x_2} \rho u dx = f_2(x_1) - f_2(x_2) = const.$$

Thus, we shall penalize the momentum over the domain (x_1, x_2). Specifically, as discussed before, the velocity is unclear over the domain where there is no state change, therefore, for the Sod and Lax problems considered in the present work, we shall let the domain (x_1, x_2) be the domain where there is no state change, namely, x_1 is the shock point at final time and x_2 is the right end point. Furthermore, we would like to utilize the momentum $\int_{x_1}^{x_2} \rho u \, dx$ at several time steps given the specific values of x_1, x_2. Thus, we add one more weighted term to the loss function $Loss_{data}$ resulting in

$$Loss = \omega_{\nabla\rho} MSE_{\nabla\rho} + \omega_p MSE_{p^*} + \omega_m MSE_{Mass_0} + \omega_M MSE_{Mom} + \omega_F Loss_F, \tag{4.3}$$

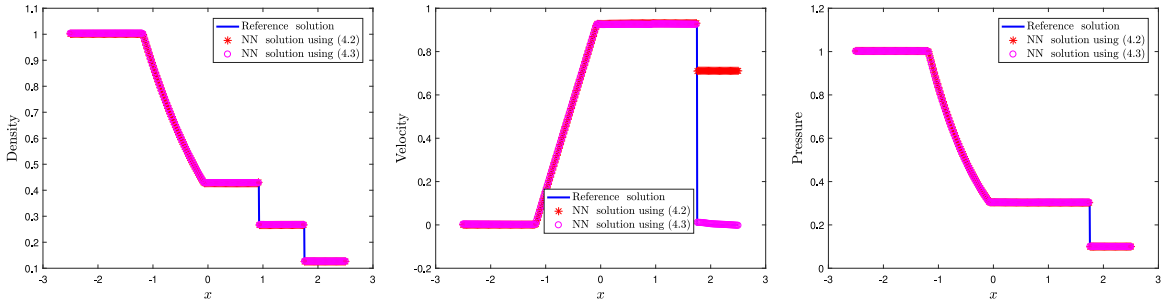

Fig. 12. Inverse problem of Example 3, i.e., the inverse Sod problem, using the data shown in Fig. 10: PINN solutions of the density (left), velocity (middle) and pressure (right) obtained by using the loss function (4.2). Given also is information from a pressure probe located at $x^* = 0.6$. The results corresponding to red star points are obtained by using the loss function (4.2) while the results corresponding to magenta circle points are obtained by using the loss function (4.3).

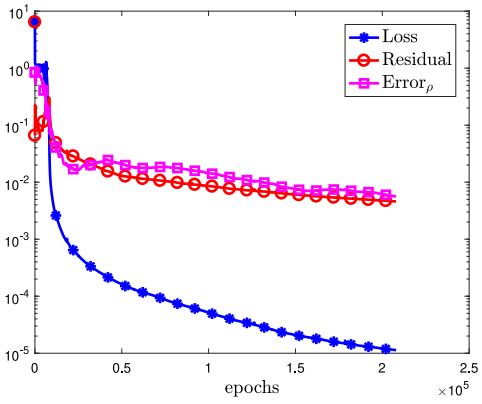

Fig. 13. Inverse problem of Example 3, i.e., the inverse Sod problem, using the data shown in Fig. 10: Loss, residual and the error for density against the epochs. Here "epochs" refers to the number of steps that the Adam and the L-BFGS-B optimizers perform.

where $MSE_{\nabla\rho}$ and MSE_{p^*} are given by (B.2) and MSE_{Mass_0} is given by (B.4), and

$$MSE_{Mom} = \frac{1}{N_M}\sum_{j=1}^{N_M}\left(\int_{x_1}^{x_2}(\rho_{NN}u_{NN})(x,t_j^M)dx - \int_{x_1}^{x_2}(\rho u)(x,t_j^M)dx\right)^2, \tag{4.4}$$

where $\{t_j^M\}_{j=1}^{N_M}$ are the collocations points with N_M being the number of points. In the computation, we set $\omega_M = 1.0$ and use the trapezoidal rule with 100 points to compute each integral.

We present the numerical result in Fig. 12, (see the results corresponding to the magenta circle points). We can see that we now obtain satisfactory predictions for the density, velocity and pressure. The convergence of the loss, the residual of the equation and the L^2 relative error of the density at time $t = 1$ are shown in Fig. 13. The residual is computed by using the formula (3.2) where the points are the training points used in the learning process.

For the choice of the point x^*, we also test several other cases, for example, a point between the left boundary and the rarefaction wave at the final time or, a point between the rarefaction wave at the final time and the initial discontinuous point, and we cannot get good predictions for the velocity and pressure. So we suggest to choose the point x^* from the domain that is between the initial discontinuous point and the shock point at the final time.

4.1.3. Effect of the distribution of the training points

Previously, we use the clustered training points, namely, we use more points around the discontinuous region while use less points in the smooth region. We compare in this test the result obtained by using clustered training

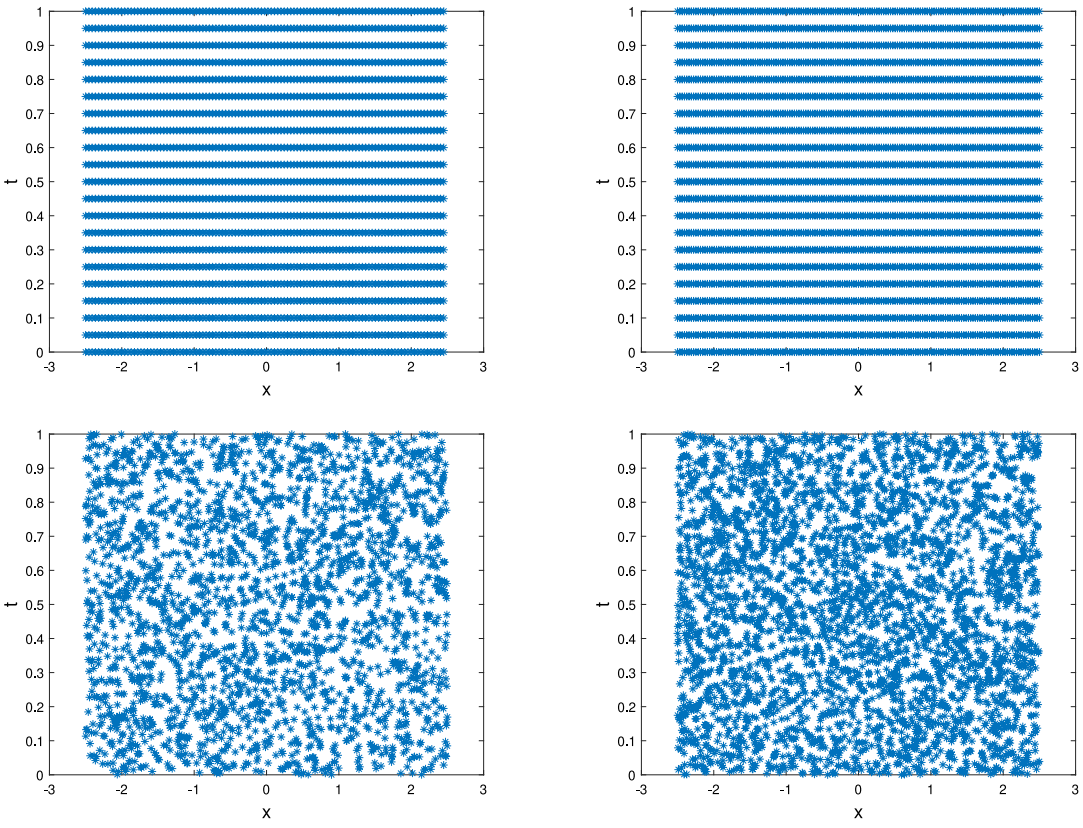

Fig. 14. Distribution of the training points for the inverse problem of Example 3, i.e., the inverse Sod problem: Upper left: uniformly distributed data for the density gradient; Upper right: uniformly distributed points for F, i.e., the Euler equations; Lower left: randomly distributed data for the density gradient; Lower right: randomly distributed points for F, i.e., the Euler equations.

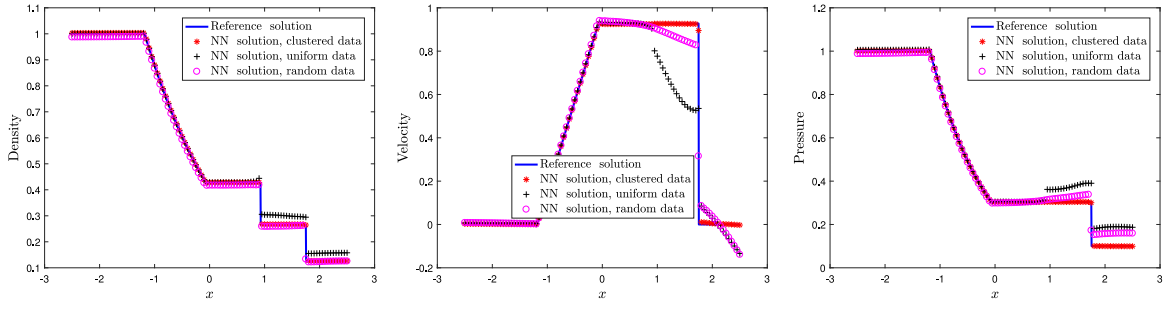

Fig. 15. Effect of the distribution of the data for the inverse problem of Example 3, i.e., the inverse Sod problem: Comparison of the PINN solutions of the density (left), velocity (middle) and pressure (right) with different distributions of data by using the loss function (4.3). Given also is information from a pressure probe located at $x^* = 0.6$.

points with the ones obtained by using uniformly (see upper plots of Fig. 14) and randomly (see lower plots of Fig. 14) distributed training points.

In this test, we use the networks with the same architecture as the one considered in the last subsection, which has 4 hidden layers with 120 neurons in each layer for these three types of data and set the numbers of training points: $N_F = 3340$, $N_{\nabla\rho} = 2330$, $N_p = 200$, $N_m = 10$. The comparison of the results of the density, velocity and pressure at time $t = 1.0$ by using different types of training data is shown in Fig. 15. We can see that the result obtained by using the clustered mesh exhibits *superior* performance overall.

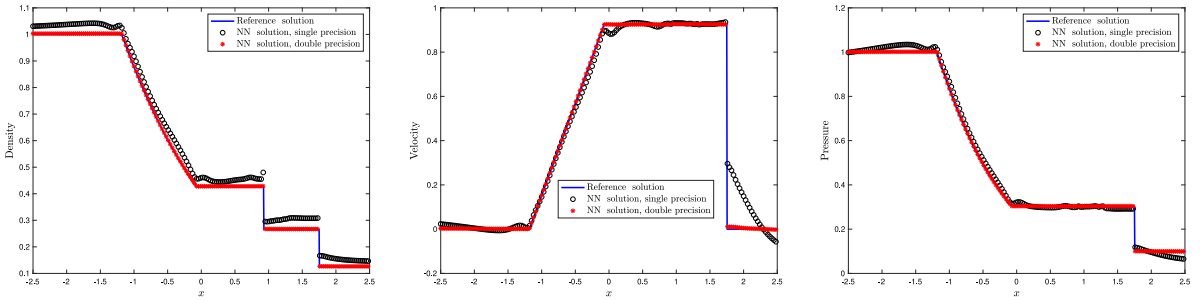

Fig. 16. Effect of arithmetic precision for the inverse problem of Example 3, i.e., the inverse Sod problem: Comparison of PINN solutions of the density (left), velocity (middle) and pressure (right) obtained with single precision against the double precision by using the loss function (4.3). Given also is information from a pressure probe located at $x^* = 0.6$.

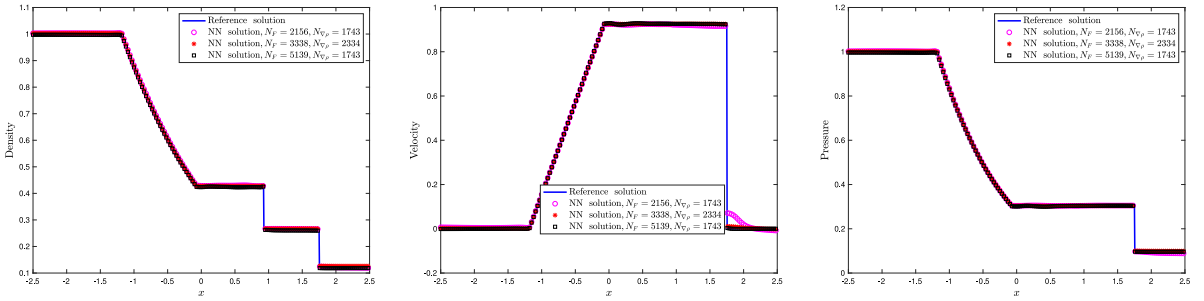

Fig. 17. Effect of the number of training points for the inverse problem of Example 3, i.e., the inverse Sod problem: Comparison of PINN solutions of the density (left), velocity (middle) and pressure (right) obtained with loss function (4.3) and different number of training points. Given also is information from a pressure probe located at $x^* = 0.6$.

4.1.4. Influence of the arithmetic precision

For the problem with smooth solutions, we solve the inverse problem with single precision and obtain solutions with satisfactory accuracy; see the result in Appendix B. However, for the inverse Lax problem, we use the double precision in all previous tests. In this test, we study the influence of the arithmetic precision, especially for the inverse Lax problem. In particular, we compare the results obtained by using single precision with the ones obtained by using double precision. The results are shown in Fig. 16.

Observe that the results using double precision are much better than the ones using single precision for the inverse Lax problem.

4.1.5. Using different numbers of training and residual points in the smooth region

Firstly, similarly as we did in Example 1 for the forward problem, for the inverse Sod problem, we keep the dense points around the discontinuous region and use much less points in the smooth region ($N_F = 2156,\ N_{\nabla\rho} = 1743$), which is about 2/3 number of the training and residual points as used previously ($N_F = 3338,\ N_{\nabla\rho} = 2334$). The results are shown in Fig. 17 (magenta circle and red star). Observe that we obtain a good prediction except of a small flaw for the velocity. However, this can be resolved by using more residual points, i.e., sample points for the equation ($N_F = 5139$). See the result (black square) in Fig. 17. In fact, we can sample as many points as desired for the equation without increasing substantially the overall cost by using the mini-batch procedure.

4.1.6. Using the Euler equations in characteristic form

Instead of using the Euler equations in conservative form, i.e., Eq. (1.1), we now employ the Euler equations in characteristic form. In particular, Eq. (1.1) can be rewritten as [50]

$$U_t + \boldsymbol{A} U_x = 0, \tag{4.5}$$

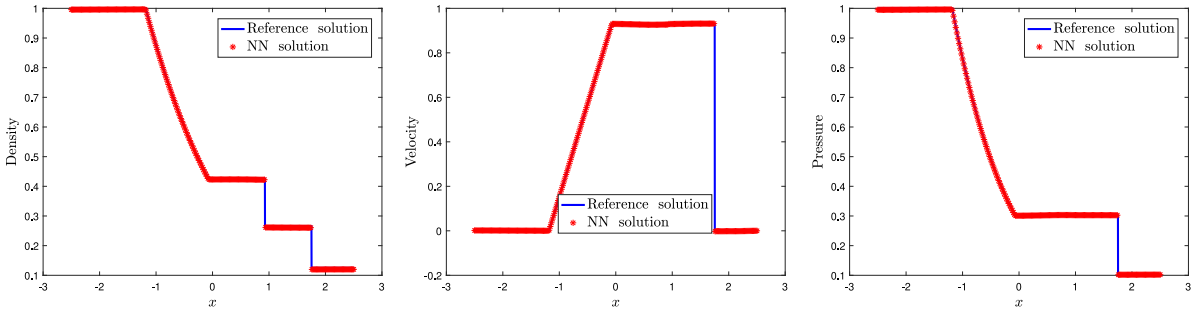

Fig. 18. Effect of the characteristic form for the inverse problem of Example 3, i.e., the inverse Sod problem: PINN solutions of the density (left), velocity (middle) and pressure (right) by using the loss function (4.3) and the characteristic equation (4.6). Given also is information from a pressure probe located at $x^* = 0.6$.

where $\boldsymbol{A}(U) = \frac{\partial f}{\partial U}$. Furthermore, it can be written as

$$\boldsymbol{L}U_t + \boldsymbol{D}\boldsymbol{L}U_x = 0, \tag{4.6}$$

where $\boldsymbol{D}$ and $\boldsymbol{L}$ are the diagonal matrix of the eigenvalues and the left eigenvectors of $\boldsymbol{A}$, respectively, given by

$$\boldsymbol{D} = \begin{pmatrix} u-c & & \\ & u & \\ & & u+c \end{pmatrix}, \quad \boldsymbol{L} = \begin{pmatrix} \frac{u^2\beta c/2+u}{2c} & -\frac{1+u\beta c/2}{2c} & \frac{\beta}{2} \\ 1-\frac{1}{2}\beta u^2 & \beta u & -\beta \\ \frac{u^2\beta c/2-u}{2c} & \frac{1-u\beta c/2}{2c} & \frac{\beta}{2} \end{pmatrix}$$

with $c = \sqrt{\gamma p/\rho}$ being the speed of the sound and $\beta = \frac{\gamma-1}{c^2}$.

We now use the characteristic equation (4.6) to replace Eq. (1.1), namely, for the function F, instead of using (2.1), we use

$$F = \boldsymbol{L}U_t + \boldsymbol{D}\boldsymbol{L}U_x. \tag{4.7}$$

In the PINN, to ensure the positivity of the density and the pressure, we let $\rho = e^{O_1}$, $u = O_2$, $P = e^{O_3}$, where O_j, $j = 1, 2, 3$ are the three outputs of the PINN.

For the training details, we use the clustered training points, and a neural network that has 3 hidden layers with 120 neurons at each layer, and we train all models with the Adam optimizer with an initial learning rate of 0.0005 for 12 000 steps followed by a L-BFGS-B optimizer with 200 000 steps. We set the weights $\omega_{\nabla\rho} = \omega_F = \omega_p = \omega_m = \omega_M = 1.0$. The numerical results for the density, velocity and pressure are shown in Fig. 18. Observe that we obtain, as the case of using the conservative form, very good results for all states. The results for both (conservative and characteristic) forms are similar, however, as shown in the next subsection of the inverse Lax problem, the result obtained by using the characteristic form is better than the one obtained by using the conservative form.

4.1.7. Lax problem

In this subsection, we shall solve the inverse problem for the Lax problem using the same type of data, i.e., we use the data on density gradient $\nabla\rho(x, t)$, the pressure $p(x^*, t)$ as well as the global conservation of mass and momentum. In particular, again, we use the loss function (4.3) and we let $dx = 0.02$.

Example 4. We consider the Riemann problem subject to the following initial condition

$$U(x, 0) = U_0(x) = \begin{cases} U_L, & x < 0, \\ U_R, & x > 0 \end{cases}$$

with

$$U_L = (\rho_L, v_L, p_L)^T = (0.445, 0.698, 3.528)^T, \quad U_R = (\rho_R, v_R, p_R)^T = (0.5, 0, 0.571)^T. \tag{4.8}$$

The computational domain and the time interval are set to $(x, t) \in [-2.5, 2.5] \times [0, 0.7]$.

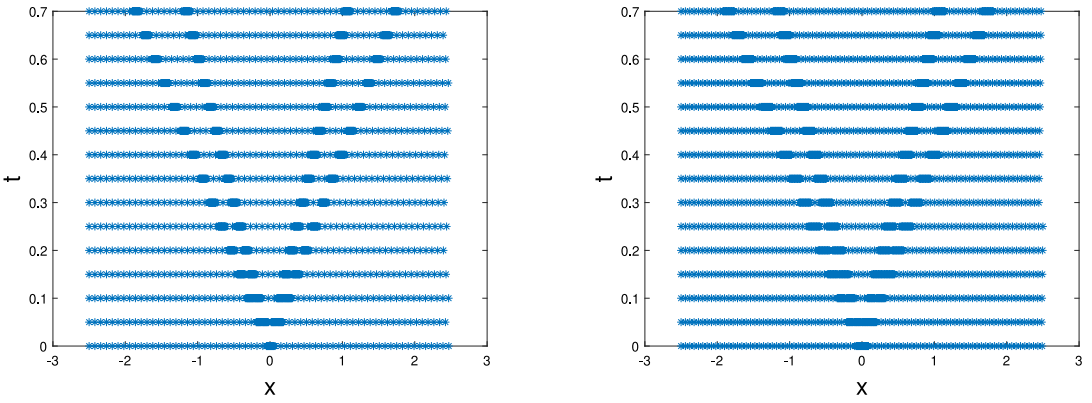

Fig. 19. Inverse problem of Example 4, i.e., the inverse Lax problem: Distribution of the clustered training points. We use more points around the discontinuous regions while use less point in the smooth regions. Left: distribution of training data for the density gradient, right: Distribution of training points for F, i.e., the Euler equations.

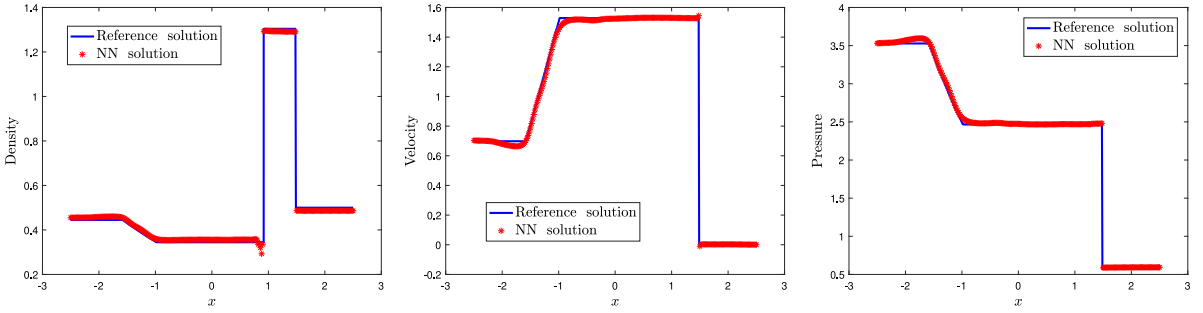

Fig. 20. Effect of the conservative form for the inverse problem of Example 4, i.e., the inverse Lax problem: PINN solutions of the density (left), velocity (middle) and pressure (right) obtained by using the loss function (4.3) and the Euler equations in the *conservative* form. Given also is information from a pressure probe located at $x^* = 0.6$.

In this example, we set the weights of the loss function used in this example all to be 1.0, i.e., $\omega_{\nabla\rho} = \omega_F = \omega_p = \omega_m = \omega_M = 1.0$. For the distribution of the data, we use the clustered data shown in Fig. 19.

We first consider the case of using the Euler equations in conservative form, i.e., Eq. (1.1). In this case, we use a neural network having 3 hidden layers with 200 neurons at each layer and, train all models with the Adam optimizer with an initial learning rate of 0.0005 with 30 000 steps followed by a L-BFGS-B optimizer with 300 000 steps. The results for the density, velocity and pressure at time $T = 0.7$ are shown in Fig. 20. Observe that we infer all the states and capture the shock. However, the accuracy is not so good. Then, to obtain more accurate solutions, we next employ the equation in characteristic form in the next test.

We now consider to use the *characteristic* equation, i.e., Eq. (4.6). In this case, we use a neural network having 3 hidden layers with 160 neurons at each layer and, train all models with the Adam optimizer with an initial learning rate of 0.0005 with 12 000 steps followed by a L-BFGS-B optimizer with 300 000 steps. The results for the density, velocity and pressure at time $T = 0.7$ are shown in Fig. 21. We can see that unlike the inverse Sod problem, we obtain higher accuracy solutions than that by using the conservative form.

4.1.8. Riemann problem with high pressure ratio

Example 5. In this example we consider the inverse Riemann problem with high pressure ratio, for instance, $p_L/p_R = 1000$. In particular, we consider the Riemann problem subject to the following initial condition

$$U(x,0) = U_0(x) = \begin{cases} U_L, & x < 0.5, \\ U_R, & x > 0.5 \end{cases}$$

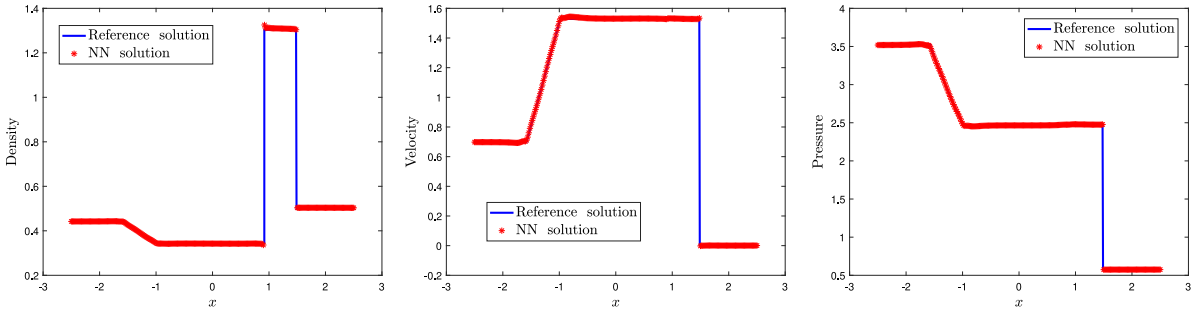

Fig. 21. Effect of the characteristic form for the inverse problem of Example 4, i.e., the inverse Lax problem: PINN solutions of the density (left), velocity (middle) and pressure (right) obtained by using the loss function (4.3) and the Euler equations in the *characteristic* form. Given also is information from a pressure probe located at $x^* = 0.6$.

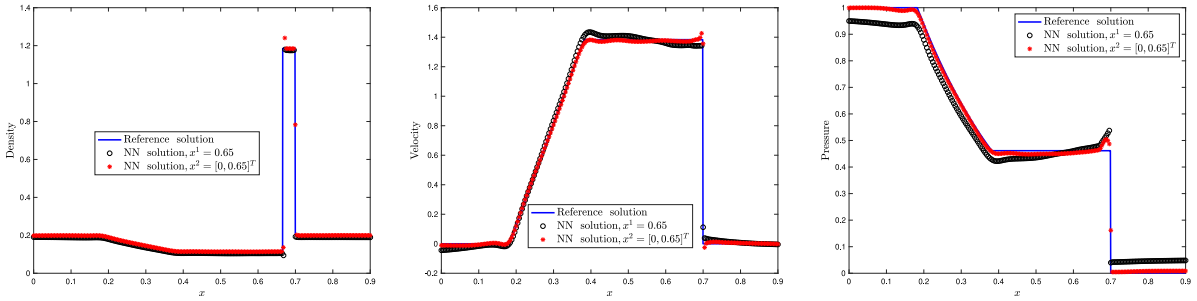

Fig. 22. Inverse problem for Example 5, i.e., the inverse Riemann problem with high pressure ratio: Comparison of PINN solutions of the density (left), velocity (middle) and pressure (right) obtained by using the loss function (4.3) and the Euler equations in the *characteristic* form with two different sets of pressure information. $x^1 = 0.65$ means that we use the information from a pressure probe located at $x = 0.65$ while $x^2 = [0, 0.65]^T$ means that we use the information from two pressure probes located at $x = 0$ and 0.65.

with

$$U_L = (\rho_L, v_L, p_L)^T = (0.2, 0, 1.0)^T, \quad U_R = (\rho_R, v_R, p_R)^T = (0.2, 0, 0.001)^T. \tag{4.9}$$

The computational domain and the time interval are set to $(x, t) \in [0, 0.9] \times [0, 0.18]$.

Again, we use the data on density gradient $\nabla\rho(x, t)$ as well as the global conservation of mass and momentum. We point out here that in this case we use the pressure values located at *two* points, one is the left boundary while another is $x = 0.65$. We use the equation in the characteristic form and the loss function (4.3) and we let $dx = 0.02$. Furthermore, we set $\omega_{\nabla\rho} = \omega_F = \omega_p = \omega_m = \omega_M = 1.0$. For the distribution of the data, we use the clustered data.

We train the neural network which has 3 hidden layers with 200 neurons at each layer, with the Adam optimizer with an initial learning rate of 0.0005 with 15 000 steps followed by a L-BFGS-B optimizer with 250 000 steps. The results of the comparison for the density, velocity and pressure at time $T = 0.12$ with the pressure values located at $x = 0.65$ and $x = [0, 0.65]^T$ are shown in Fig. 22. Here $x^1 = 0.65$ means that we use the information from a pressure probe located at $x = 0.65$ and $x^2 = [0, 0.65]^T$ means that we use the information from two pressure probes located at $x = 0$ and 0.65. Observe that we can obtain good prediction by using more pressure values.

4.2. Two-dimensional problem: learning the EOS from data

For the equation of state, i.e., Eq. (1.2), we set the adiabatic index $\gamma = 1.4$ for the polytropic gas. However, from the physical point of view, the values of γ may vary depending on the type of the gas. Therefore, it is interesting to see whether we can learn the value of the parameter γ from given data of the states as well as the governing equation. Thus, in this subsection, we would like to solve another type of inverse problem for the two-dimensional oblique shock wave problem using PINN, i.e., we shall learn the value of the parameter γ from given data.

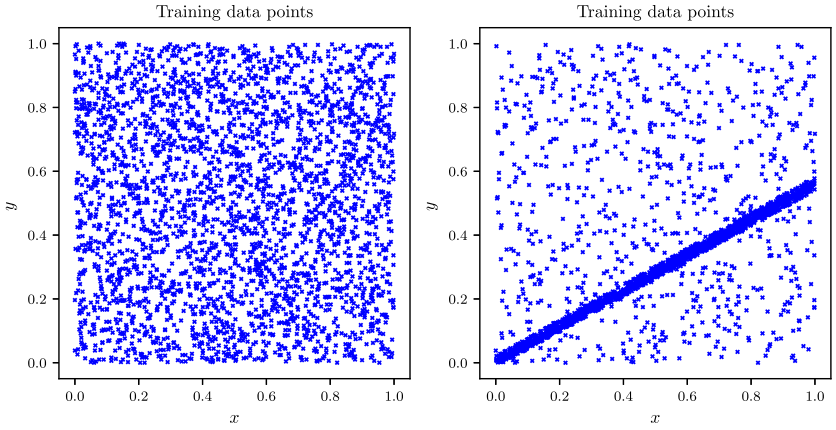

Fig. 23. Inverse oblique shock wave problem: Distribution of the training points. Left: randomly distributed points (4600 points); Right: clustered points (7000 points).

Table 3
Inverse problem of Example 2: Identification of γ using clean data, 1% and 2% noise data using random and clustered training points.

True γ	Identified γ (clean data)		Identified γ (1% noise)		Identified γ (2% noise)	
	Random	Clustered	Random	Clustered	Random	Clustered
1.4	1.4618	1.3966	1.3192	1.4089	1.2975	1.4098

Assuming we have data for the density, velocity and pressure $\rho^D := \rho(x_j^D, y_j^D)$, $\boldsymbol{u}^D := \boldsymbol{u}(x_j^D, y_j^D)$, $p^D := p(x_j^D, y_j^D)$, $j = 1, \ldots, N_D$, where $\{x_j^D, y_j^D\}_{j=1}^{N_D}$ are the training points. We then use the following loss function

$$Loss = MSE_{BC} + MSE_D + MSE_F, \tag{4.10}$$

where MSE_{BC} and MSE_F are given by Eqs. (3.5) and (2.3), respectively, and

$$MSE_D = \frac{1}{N_D}\sum_{i=1}^{N_D}\left(|\rho^D - \rho_{NN}(x_j^D, y_j^D)|^2 + |\boldsymbol{u}^D - \boldsymbol{u}_{NN}(x_j^D, y_j^D)|^2 + |p^D - p_{NN}(x_j^D, y_j^D)|^2\right).$$

Note that in this case, the parameter γ is unknown and to be discovered.

In this example, we test two sets of data for both MSE_D and MSE_F, namely, the randomly distributed data and the clustered data, see the distribution of data in Fig. 23; For the BCs, we use the randomly distributed data with $N_{BC} = 300$. The numerical results of the comparison of the all states with the exact solution at different x locations corresponding to these two sets of (clean) data are shown in Figs. 24 and 25, respectively.

Furthermore, we test two more sets of noisy data (with 1% and 2% noise) for each set of (random or clustered) data to identify the value of the parameter γ. The summary of the inferred values of the parameter γ for the two types of data is shown in Table 3. Observe that we obtain good identification of the value of the parameter γ by using the clustered points, even for the case of noisy data.

The convergence of the loss with respect to the epochs is presented in Fig. 26 showing that the accuracy of PINN solution using clustered training points, even with much less points, is higher than that with random training points.

5. Conclusion and discussion

In this work we employ the physics-informed neural networks (PINNs) to solve the forward and inverse problems for the one-dimensional and two-dimensional Euler equations that model high-speed aerodynamic flows. By using the Euler equations and the initial/boundary conditions (IC/BCs), we construct the loss function for the forward problem and consequently solve the one-dimensional Euler equations with smooth solutions or with solutions that

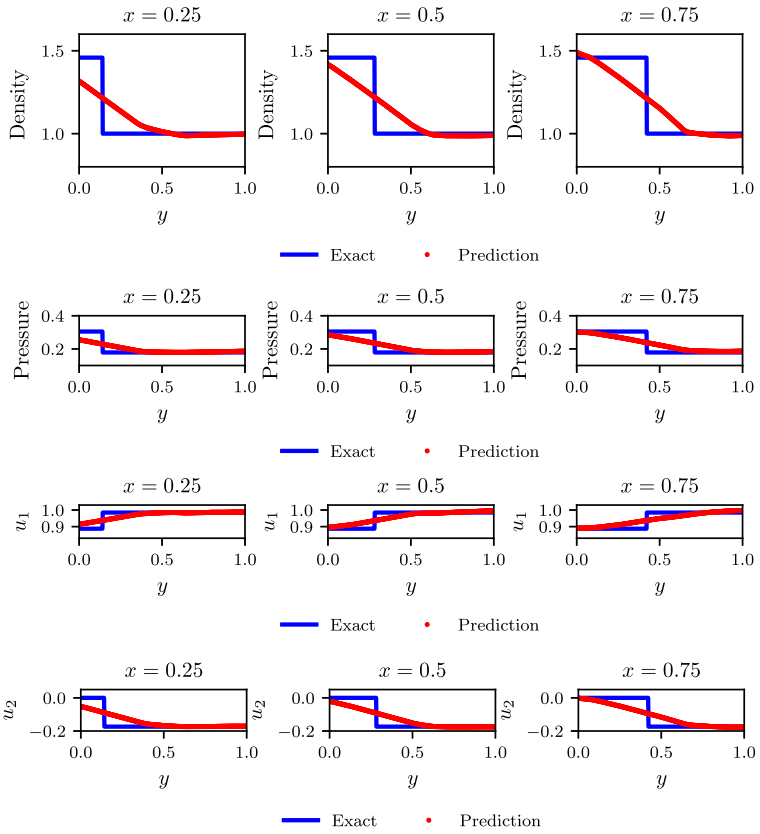

Fig. 24. Inverse oblique shock wave problem: Comparison of PINN solutions using uniform training points with the exact solution at various x locations.

have a contact discontinuity as well as a two-dimensional oblique shock wave problem, showing that the inference of smooth solutions is better than that of discontinuous solutions. Also, the inference of the solutions using clustered training points, i.e., more data around the discontinuous region, captures the discontinuity well. By mimicking the Schlieren photography experimental technique used traditionally in high-speed aerodynamics, we have data on density gradient $\nabla\rho(x,t)$ and hence we can use it to infer the density, velocity and pressure fields. *Without* using the IC/BCs, we investigate what is the minimum number of additional measurements required, i.e., on the pressure $p(x^*,t)$. Furthermore, we incorporate other invariants known *a priori*, e.g., global conservation of mass and momentum. We present illustrative benchmark examples for both the problem with smooth solutions and Sod and Lax problems with PINNs, showing that all inferred states are in good agreement with the reference solutions. We also show that the choice of the position of the point x^* plays an important role in order to infer accurately the velocity and pressure, especially for Sod and Lax problems. In particular, for the problem with smooth solutions we can randomly choose the position of the point x^* from the computational domain, while for the Sod or Lax problem, we choose the position of the point x^* from the domain between the initial discontinuous point and the shock position at the final time. In particular, for the Riemann problem with high pressure ratio, we require pressure values at more locations. Furthermore, we solve the inverse problem by combining the aforementioned data and the Euler equations in characteristic form, showing that the result obtained by using the Euler equations in characteristic form is better than that obtained by using the Euler equations in conservative form. Finally, we solve an inverse problem by employing PINNs to learn the value of the parameter γ in the equation of state for the parameterized two-dimensional oblique wave problem with measurements of the density, pressure and velocity fields, and we obtain good identifications of the parameter γ using the clustered data.

We have shown that we can obtain more accurate solution with the clustered (but random) training points, i.e., more data around the discontinuous region, compared with that with random or uniform training points.

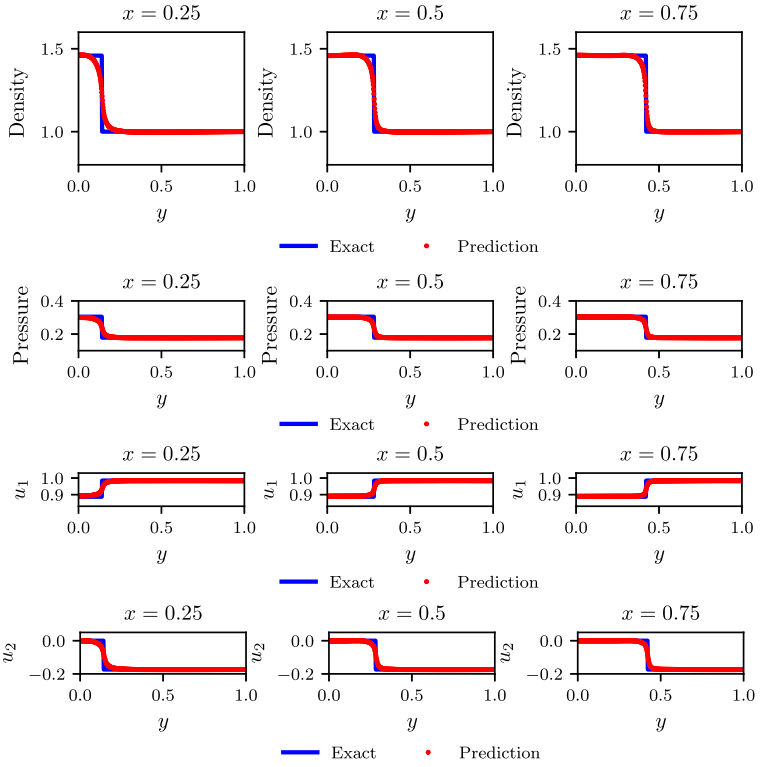

Fig. 25. Inverse oblique shock wave problem: Comparison of PINN solutions using clustered training points with the exact solution at various x locations.

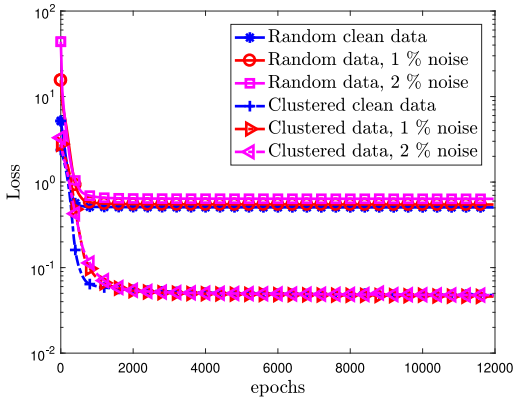

Fig. 26. Inverse problem of Example 2: Loss against the epochs using randomly distributed and clustered training points. Here "epochs" refers to the number of steps that the Adam optimizer performs.

However, it is hard to sample the training points without *a priori* knowledge of the solution. In future work, we are going to develop an adaptive algorithm, with which we can adaptively sample the training points to obtain high accuracy solution for sharp interface or discontinuous problems. There are two possibilities to develop the adaptive sampling strategy, the first one is based on the PDE residual; in particular, we can initially sample some training points and then adaptively add training points where the residual values are relatively big. Another strategy is to use the gradient of the predicated solutions, namely, we can add the points where the solution has large gradient. The main point here is that even if we start with totally random points we can locate the shock and subsequently we can add points adaptively based on the values of our residual evaluate around the shock or steep gradients. In example

Example 1, we tested two different architectures and obtained good predictions for both cases. For the inverse Sod or Lax problem, our simulation experience tells us that it is much harder to choose a good network architecture, and here one may employ the meta-learning strategies, which is currently a hot topic in machine learning, for instance, see [42–45]. To put things in perspective, it took conventional methods more than 50 years to mature and we still do not have effective techniques for good meshes for multi-physics problems. Similarly, it will take some time to accumulate some experience on how to design a good NN architecture for high-speed aerodynamic flows but the new thrust for efficient search for best architectures using meta-learning looks very promising.

Acknowledgments

This work was supported by the AFOSR grant FA9550-17-1-0013. The last author (GEK) would like to thank the Alexander von Humboldt foundation that enabled him to collaborate with Prof Adams of TUM and initiate this investigation. In addition, we would like to thank Prof. Nikolaus A. Adams for helpful discussions.

Appendix A. Forward problem for the one-dimensional Euler equations with smooth solutions

In this Appendix, we consider the Euler equations that have smooth solutions. In particular, we consider the following example:

Example 6. Consider the Euler equation (1.1) in the one-dimensional case with the periodic boundary conditions

$$U(a,t) = U(b,t), \quad \nabla U(a,t) = \nabla U(b,t)$$

and initial conditions

$$U_0 = (\rho_0, \quad u_0, \quad p_0) = (1.0 + 0.2\sin(\pi x), \quad 1.0, \quad 1.0)\,,$$

in which case we have the exact solutions

$$(\rho, \quad u, \quad p) = (1.0 + 0.2\sin(\pi(x-t)), \quad 1.0, \quad 1.0)\,,$$

where $x \in (-1, 1)$.

We first consider the forward problem; in this case, we use the loss function (3.1), where the mean square errors MSE_{IC} and MSE_{BC} corresponding to the IC/BCs are given by

$$MSE_{IC} = \frac{1}{N_i}\sum_{j=1}^{N_i} |U_0(x_j^{IC}) - U_{NN}(x_j^{IC}, 0)|^2,$$

$$MSE_{BC} = \frac{1}{N_b}\sum_{j=1}^{N_b} \left(|U_{NN}(-1, t_j^{BC}) - U_{NN}(1, t_j^{BC})|^2 + |\nabla U_{NN}(-1, t_j^{BC}) - \nabla U_{NN}(1, t_j^{BC})|^2 \right),$$

where $\{x_j^{IC}\}_{j=1}^{N_{IC}}$ and $\{t_j^{BC}\}_{j=1}^{N_{BC}}$ are the training points for the IC/BCs, respectively, N_{IC} and N_{BC} are the numbers of points associated with the IC/BCs.

In the test, we use randomly distributed training points and the number of training points is $N_{BC} = 50$, $N_{IC} = 50$, $N_F = 2000$. We employ a neural network that has 7 hidden layers with 20 neurons in each layer and train the model using the Adam optimizer for 5000 steps with an initial learning rate of 0.001 followed by a L-BFGS-B optimizer with 2000 steps. The neural network approximations of the density, velocity and the pressure at time $t = 1.0$ are shown in Fig. A.27.

Observe that in this case, the neural network solutions agree well with the exact solutions. We also show the loss, the residual of the equation and the L_2 relative error of the density at time $t = 2$ in Fig. A.28. We observe that the L_2 relative error of the density has the similar decay rate as the one of the loss while the residual of the equation decays fast at first and then increases and then decays again. Moreover, we calculate the L_2 relative errors for the density, velocity and pressure shown in Table A.4 showing that the accuracy of the results is of the order $O(10^{-5})$.

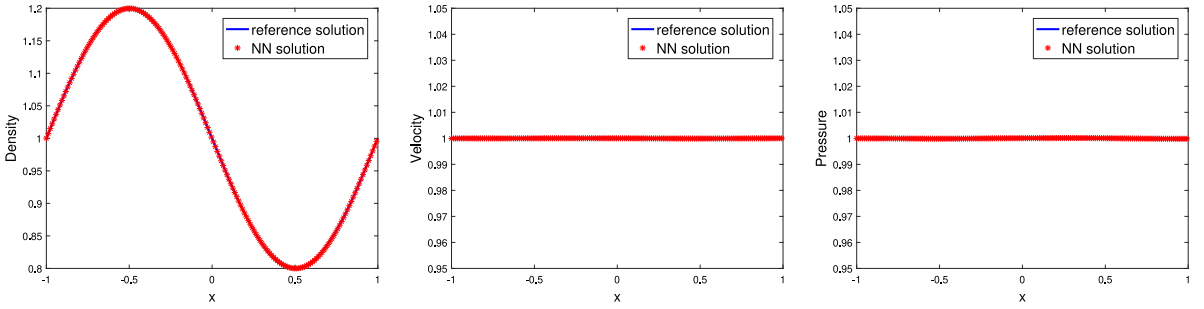

Fig. A.27. Forward problem of Example 6: PINN solutions of the density (left), velocity (middle) and pressure (right) obtained by using the loss function (3.1).

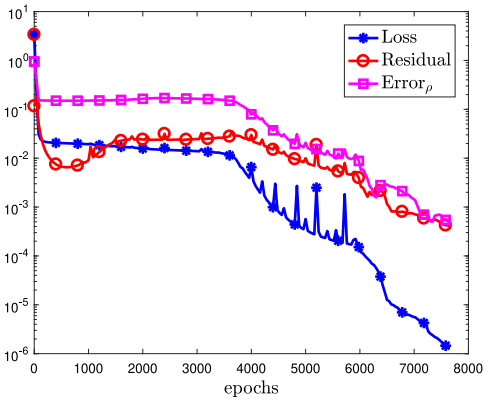

Fig. A.28. Forward problem of Example 6: Loss, residual of the equation and the relative L_2 error for the density against the epochs. Here "epochs" refers to the number of steps that the Adam and the L-BFGS-B optimizers perform.

Table A.4
Forward problem of Example 6: Relative L_2 error for the density, velocity and pressure.

State variable	Density	Velocity	Pressure
Relative L_2 error	3.031362e−04	7.098853e−05	8.844720e−05

Appendix B. Inverse problem for the one-dimensional Euler equations with smooth solutions

We now turn to the first type of the inverse problems for the Euler equations with smooth solutions, namely, we are going to use data on the density gradient, i.e., $\nabla\rho(x,t)$, and investigate what is the minimum number of additional measurements required, i.e., on the pressure $p(x^*,t)$ as well as global conservation of mass for Example 6. To this end, let us start from the data of the gradient of density $\nabla\rho(x,t)$ and the pressure $p(x^*,t)$. In particular, we employ the following loss function

$$Loss = Loss_{Data} + Loss_F = MSE_{\nabla\rho} + MSE_{p^*} + MSE_F \tag{B.1}$$

in the learning process, where MSE_F is given by (2.3) and

$$\begin{aligned} MSE_{\nabla\rho} &= \frac{1}{N_{\nabla\rho}}\sum_{j=1}^{N_{\nabla\rho}} |\nabla\rho(x_j^{\nabla\rho}, t_j^{\nabla\rho}) - \nabla\rho_{NN}(x_j^{\nabla\rho}, t_j^{\nabla\rho})|^2, \\ MSE_{p^*} &= \frac{1}{N_p}\sum_{j=1}^{N_p} |p(x^*, t_j^p) - p_{NN}(x^*, t_j^p)|^2, \end{aligned} \tag{B.2}$$

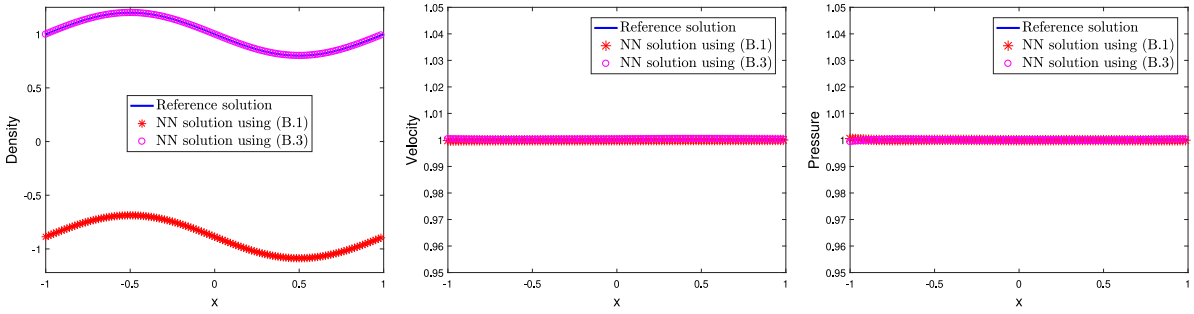

Fig. B.29. Inverse problem of Example 6. PINN solutions of the density (left), velocity (middle) and pressure (right). The results corresponding to red star points are obtained by using the loss function (B.1) while the results corresponding to magenta circle points are obtained by using the loss function (B.3).

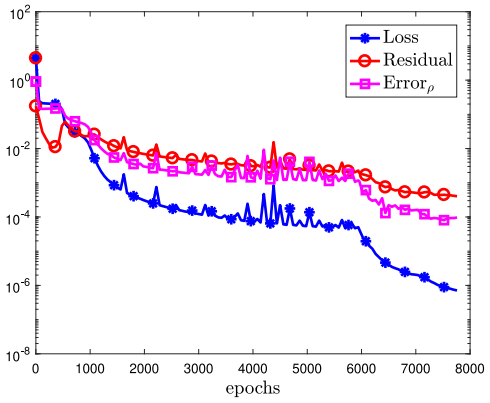

Fig. B.30. Inverse problem for Example 6. Loss, residual and the error for density against the epochs. Here "epochs" refers to the number of steps that the Adam and the L-BFGS-B optimizers perform.

Table B.5
Inverse problem for Example 6. Relative L_2 error for the density, velocity and pressure.

State variable	Density	Velocity	Pressure
Relative L_2 error	2.672760e−04	2.915027e−04	3.050062e−04

where $\{x_j^{\nabla\rho}, t_j^{\nabla\rho}\}_{j=1}^{N_{\nabla\rho}}$ are the training points for $\nabla\rho$ with $N_{\nabla\rho}$ being the number of points while $\{t_j^p\}_{j=1}^{N_p}$ are the training points for p with N_p being the number of points. x^* is a given point. We point out here that the choice of the point x^* plays an important role for the inference. For the case considered in this subsection, we *randomly* choose the point x^* from the training domain $[a, b]$. However, as shown in Section 4.1.2, we cannot randomly choose the point from the training domain, we should choose the point from a subset of the training domain.

In this test, we use a neural network that has 7 hidden layers with 20 neurons at each layer and randomly distributed data with $N_{\nabla\rho} = 640$, $N_p = 50$ and $N_F = 2000$, and train the model with the Adam optimizer with an initial learning rate of 0.001 for 5000 steps followed by a L-BFGS-B optimizer with 3000 steps. Here and after, we would like to use the single precision. The results are shown in Fig. B.29, (see the results corresponding to red star points).

Observe that the predictions for the velocity and the pressure are in good agreement with the exact solutions. However, we cannot have good prediction of the density. Therefore, we should have more information to obtain a good inference of the density.

In the present work, we adopt the conservation law of mass. Actually, we only require the equivalence of the predicted and true mass at one time step, here we use the initial mass. To this end, the loss function is given by

$$Loss = MSE_{\nabla\rho} + MSE_{p^*} + MSE_{Mass_0} + MSE_F, \tag{B.3}$$

where $MSE_{\nabla\rho}$ and MSE_{p^*} are given by (B.2), and

$$MSE_{Mass_0} = \left(\int_\Omega \rho_{NN}(x,0)dx - \int_\Omega \rho_0(x)dx \right)^2. \tag{B.4}$$

To compute the above integral, we use a trapezoidal rule with a uniform grid containing 1000 points.

By using the network that has the same architecture as the above, we obtain the results for the density, velocity and pressure shown in Fig. B.29, (see the results corresponding to magenta circle points). We see that we infer the density, velocity and pressure pretty well. This means that in this case, by using the data of $\nabla\rho(x,t)$, $p(x^*,t)$ and the initial mass, we can predict all states well with the neural network.

Same as the case for the forward problem, we show the plot of the loss, the residual of the equation and the L_2 relative error of the density at time $t = 2$ against the epochs in Fig. B.30. We observe similar convergence as the case of the forward problem, namely, the loss and the L_2 relative error have similar convergence with respect to the epoch, and the residual of the equation converges fast at first and then diverges and then converges again. We present the L_2 relative errors for the density, velocity and pressure in Table B.5 showing that solutions with the relative L^2 error of the order $O(10^{-4})$.

References

[1] R. Courant, K.O. Friedrichs, Supersonic Flow and Shock Waves, Vol. 21, Springer Science & Business Media, 1999.
[2] H.W. Liepmann, A. Roshko, Elements of Gasdynamics, Courier Corporation, 2001.
[3] R.D. Zucker, O. Biblarz, Fundamentals of Gas Dynamics, John Wiley & Sons, 2002.
[4] C.M. Dafermos, Hyperbolic Conservation Laws in Continuum Physics, fourth ed., in: Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 325, Springer-Verlag, Berlin, 2016.
[5] R.J. LeVeque, Finite Volume Methods for Hyperbolic Problems, in: Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge, 2002.
[6] I. Lomtev, G. Karniadakis, Discontinuous Galerkin methods in CFD, in: APS Division of Fluid Dynamics Meeting Abstracts, 1998.
[7] I. Lomtev, G.E. Karniadakis, A discontinuous Galerkin method for the Navier-Stokes equations, Int. J. Numer. Methods Fluids 29 (5) (1999) 587–603.
[8] J.S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Springer Science & Business Media, 2007.
[9] B. Cockburn, G.E. Karniadakis, C.-W. Shu, Discontinuous Galerkin Methods: Theory, Computation and Applications, Vol. 11, Springer Science & Business Media, 2012.
[10] C.-W. Shu, A brief survey on discontinuous Galerkin methods in computational fluid dynamics, Adv. Mech. 43 (6) (2013) 541–553.
[11] A. Harten, B. Engquist, S. Osher, S. Chakravarthy, Uniformaly high order essentially non-oscillatory schemes III, Math. Comput. 193 (2004) 563–594.
[12] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1) (1996) 202–228.
[13] Z. Wang, High-order methods for the Euler and Navier–Stokes equations on unstructured grids, Prog. Aerosp. Sci. 43 (1–3) (2007) 1–41.
[14] S. Pirozzoli, Numerical methods for high-speed flows, in: Annual Review of Fluid Mechanics, Volume 43, in: Annu. Rev. Fluid Mech., vol. 43, Annual Reviews, Palo Alto, CA, 2011, pp. 163–194.
[15] E. Johnsen, J. Larsson, A.V. Bhagatwala, W.H. Cabot, P. Moin, B.J. Olson, P.S. Rawat, S.K. Shankar, B. Sjögreen, H.C. Yee, X. Zhong, S.K. Lele, Assessment of high-resolution methods for numerical simulations of compressible turbulence with shock waves, J. Comput. Phys. 229 (4) (2010) 1213–1237.
[16] M. Raissi, G.E. Karniadakis, Hidden physics models: machine learning of nonlinear partial differential equations, J. Comput. Phys. 357 (2018) 125–141.
[17] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[18] G. Pang, L. Yang, G.E. Karniadakis, Neural-net-induced Gaussian process regression for function approximation and PDE solution, J. Comput. Phys. 384 (2019) 270–288.
[19] K.O. Lye, S. Mishra, D. Ray, Deep learning observables in computational fluid dynamics, arXiv preprint arXiv:1903.03040.
[20] J. Magiera, D. Ray, J.S. Hesthaven, C. Rohde, Constraint-aware neural networks for Riemann problems, arXiv preprint arXiv:1904.12794.
[21] G. Pang, L. Lu, G.E. Karniadakis, FPINNs: fractional physics-informed neural networks, SIAM J. Sci. Comput. 41 (4) (2019) A2603–A2626.
[22] L. Yang, D. Zhang, G.E. Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, arXiv preprint arXiv:1811.02033.
[23] X. Meng, G.E. Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems, J. Comput. Phys. 401 (2020) 109020.

[24] J.S. Wong, D.L. Darmofal, J. Peraire, The solution of the compressible Euler equations at low Mach numbers using a stabilized finite element algorithm, Comput. Methods Appl. Mech. Engrg. 190 (43–44) (2001) 5719–5737.
[25] M. Nazarov, A. Larcher, Numerical investigation of a viscous regularization of the Euler equations by entropy viscosity, Comput. Methods Appl. Mech. Engrg. 317 (2017) 128–152.
[26] L. Monthe, F. Benkhaldoun, I. Elmahi, Positivity preserving finite volume Roe schemes for transport-diffusion equations, Comput. Methods Appl. Mech. Engrg. 178 (3–4) (1999) 215–232.
[27] J.-L. Guermond, M. Nazarov, A maximum-principle preserving C^0 finite element method for scalar conservation equations, Comput. Methods Appl. Mech. Engrg. 272 (2014) 198–213.
[28] C. Michoski, M. Milosavljevic, T. Oliver, D. Hatch, Solving irregular and data-enriched differential equations using deep neural networks, arXiv preprint arXiv:1905.04351.
[29] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data, arXiv preprint arXiv:1808.04327.
[30] J.D. Anderson, Computational Fluid Dynamics: The Basics with Applications, International ed., McGraw-Hill, 1995.
[31] A. Bayliss, E. Turkel, Far field boundary conditions for compressible flows, J. Comput. Phys. 48 (2) (1982) 182–199.
[32] C. Coclici, W.L. Wendland, Domain decomposition methods and far-field boundary conditions for 2D compressible viscous flows, in: Recent Advances in Numerical Methods and Applications, II (Sofia, 1998), World Sci. Publ., River Edge, NJ, 1999, pp. 429–437.
[33] R. Sanders, A. Weiser, A high order staggered grid method for hyperbolic systems of conservation laws in one space dimension, Comput. Methods Appl. Mech. Engrg. 75 (1–3) (1989) 91–107.
[34] Y.-H. Tang, S.-T. Lee, J.-Y. Yang, A high-order pathline Godunov scheme for unsteady one-dimensional equilibrium flows, Comput. Methods Appl. Mech. Engrg. 161 (3–4) (1998) 257–288.
[35] M. Alves, P. Cruz, A. Mendes, F. Magalhaes, F. Pinho, P. Oliveira, Adaptive multiresolution approach for solution of hyperbolic PDEs, Comput. Methods Appl. Mech. Engrg. 191 (36) (2002) 3909–3928.
[36] J.W. Banks, J.A.F. Hittinger, J.M. Connors, C.S. Woodward, Numerical error estimation for nonlinear hyperbolic PDEs via nonlinear error transport, Comput. Methods Appl. Mech. Eng. 213/216 (2012) 1–15.
[37] L.D. Gryngarten, S. Menon, A generalized approach for sub-and super-critical flows using the Local Discontinuous Galerkin method, Comput. Methods Appl. Mech. Engrg. 253 (2013) 169–185.
[38] J.-L. Guermond, B. Popov, V. Tomov, Entropy-viscosity method for the single material Euler equations in Lagrangian frame, Comput. Methods Appl. Mech. Engrg. 300 (2016) 402–426.
[39] X. Nogueira, L. Ramírez, S. Clain, R. Loubère, L. Cueto-Felgueroso, I. Colominas, High-accurate SPH method with multidimensional optimal order detection limiting, Comput. Methods Appl. Mech. Engrg. 310 (2016) 134–155.
[40] Z. Ji, L. Fu, X.Y. Hu, N.A. Adams, A new multi-resolution parallel framework for SPH, Comput. Methods Appl. Mech. Engrg. 346 (2019) 1156–1178.
[41] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018) 1–43.
[42] J.S. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: Advances in Neural Information Processing Systems, 2011, pp. 2546–2554.
[43] J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms, in: Advances in Neural Information Processing Systems, 2012, pp. 2951–2959.
[44] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, R. Adams, Scalable bayesian optimization using deep neural networks, in: International Conference on Machine Learning, 2015, pp. 2171–2180.
[45] X. Chen, J. Duan, G.E. Karniadakis, Learning and meta-learning of stochastic advection-diffusion-reaction systems from sparse measurements, arXiv preprint arXiv:1910.09098.
[46] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 109136 (2019) http://dx.doi.org/10.1016/j.jcp.2019.109136.
[47] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Locally adaptive activation functions with slope recovery term for deep and physics-informed neural networks, arXiv preprint arXiv:1909.12228.
[48] G.A. Sod, A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, J. Comput. Phys. 27 (1) (1978) 1–31.
[49] E.F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction, Springer Science & Business Media, 2013.
[50] G.E. Karniadakis, S.J. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics, second ed., Oxford University Press, 2013.