



# Learning the flux and diffusion function for degenerate convection-diffusion equations using different types of observations

Qing Li<sup>1</sup> · Steinar Evje<sup>1</sup>

Received: 16 July 2023 / Accepted: 7 March 2024  
© The Author(s) 2024

## Abstract

In recent years, there has been an increasing interest in utilizing deep learning-based techniques to predict solutions to various partial differential equations. In this study, we investigate the identification of an unknown flux function and diffusion coefficient in a one-dimensional convection-diffusion equation. The diffusion function is allowed to vanish on intervals implying that solutions generally possess low regularity, i.e., are discontinuous. Therefore, solutions must be interpreted in the sense of entropy solutions which combine a weak formulation with an additional constraint (entropy condition). We explore a methodology that utilizes symbolic neural networks (S-Nets) in combination with an entropy-consistent discrete numerical scheme (ECDNS). Different types of observation data are explored. Extensive experiments in this paper demonstrate that the proposed method is a robust tool to identify the unknown flux and diffusion function. The flux and diffusion functions are restricted to be rational functions.

**Keywords** Scalar nonlinear conservation law · Degenerate convection-diffusion · Weak entropy solution · Symbolic multi-layer neural network · Entropy consistent discrete numerical scheme

**Mathematics Subject Classification** 35M12 · 35R30 · 65N21 · 68T07

---

Communicated by Nils Henrik Risebr.

---

✉ Steinar Evje  
steinar.evje@uis.no

Qing Li  
qing.li@uis.no

<sup>1</sup> Department of Energy and Petroleum, Group of Computational Engineering, University of Stavanger, Stavanger, Norway

# 1 Introduction

## 1.1 Background

Conservation laws, which are regarded as fundamental laws of nature, have wide-ranging applications in various fields such as physics, chemistry, biology, geology, and engineering. One of the significant branches in computational mathematics involves solving the differential equations associated with conservation laws, and several effective methods have been proposed in this regard, including the ones discussed in [31, 36, 37, 51, 52, 59, 81]. However, it is well known that solutions to conservation laws are prone to developing discontinuities in finite time, even for smooth initial data [19, 36, 49, 52]. Scalar nonlinear convection-diffusion partial differential equations (PDEs) of the form

$$\begin{aligned} u_t + f(u)_x &= \alpha A(u)_{xx}, & A(u) &= \int_0^u a(v)dv, & a(v) &\geq 0 \\ u|_{t=0} &= u_0(x) \\ u_x|_{x=0} &= u_x|_{x=L} = 0 \end{aligned} \quad (1.1)$$

where  $x \in [0, L]$  arise in different applications such as sedimentation of particles in liquid and various traffic flow type of problems. Here  $u = u(x, t)$  is the main variable which depends on the position  $x$  and time  $t$ . The flux function  $f(u)$  represents the convective transport whereas the diffusion function, denoted by  $A(u) = \int_0^u a(v)dv$ , is a function of  $u$  that describes the diffusive transport. For a typical situation we have a priori information about the magnitude of the scaling factor  $\alpha > 0$  but not precise information about the functional form of neither  $A(u)$  nor  $f(u)$ . Our goal is to determine analytical expressions for both  $f(u)$  and  $A(u)$  using observational data of different types.

## 1.2 Mathematical framework associated with (1.1)

Focusing on the Cauchy problem associated with (1.1), weak solutions must be considered in the sense of the integral equality

$$\int_0^T \int_{\mathbb{R}} \left( u\varphi_t + (f(u) - A(u)_x)\varphi_x \right) dx dt + \int_{\mathbb{R}} u_0(x)\varphi(x, 0) dx = 0 \quad (1.2)$$

for all test functions  $\varphi \in C_c^\infty(\mathbb{R} \times [0, T])$  where  $u \in C([0, T]; L^1(\mathbb{R})) \cap L^\infty(\mathbb{R} \times [0, T])$  and  $A(u)_x \in L^\infty(\mathbb{R} \times [0, T])$ . Herein, we have set  $\alpha = 1$  without loss of generality. If  $A(u)$  is constant on an interval then weak solutions are not uniquely determined from the initial data. An additional entropy condition must be imposed to determine the unique physically relevant solution. More precisely, this unique weak entropy solution  $u$  can be characterized by the following integral inequality when (1.1)

is considered on  $\mathbb{R}$  [10, 86]

$$\int_0^T \int_{\mathbb{R}} \left( |u - k| \varphi_t + \operatorname{sgn}(u - k) \left( f(u) - f(k) - A(u)_x \right) \varphi_x \right) dx dt + \int_{\mathbb{R}} |u_0(x) - k| \varphi(x, 0) dx \geq 0 \tag{1.3}$$

for all nonnegative  $\varphi \in C_c^\infty(\mathbb{R} \times [0, T))$  and any constant  $k \in \mathbb{R}$ . It is instructive to note that from (1.3) the following characterization of a jump  $(u_l, u_r)$  moving with a velocity  $s$  can be extracted [27]:

(i) The shock speed  $s$  is given by the Rankine-Hugoniot type of condition

$$s = \frac{f(u_r) - f(u_l) - \left( A(u)_x|_r - A(u)_x|_l \right)}{u_r - u_l} \tag{1.4}$$

where  $u_l$  and  $u_r$  are the left and right limits of  $u(x, t)$  of a discontinuity moving in the  $x - t$  space, whereas  $A(u)_x|_l$  and  $A(u)_x|_r$  are corresponding left and right limit values of  $A(u)_x$ .

(ii) For all  $v \in [\min\{u_l, u_r\}, \max\{u_l, u_r\}]$ , the following entropy condition holds

$$\frac{f(u_r) - f(v) - A(u)_x|_{u_r}}{u_r - u_l} \leq s \leq \frac{f(u_l) - f(v) - A(u)_x|_{u_l}}{u_r - u_l}. \tag{1.5}$$

For scalar conservation laws with  $a(u) = 0$ , the entropy framework (usually called entropy conditions) was introduced by Kruzkov [50] and Volpert [85], while for degenerate parabolic equations entropy solution were first considered by Volpert and Hudajev [86]. Uniqueness of entropy solutions was first proved by Carrillo [10]. Finite difference and finite volume schemes for degenerate equations were analysed by Evje and Karlsen [26–28] (using upwind difference schemes), Holden et al. [37] (using operator splitting methods), Kurganov and Tadmor [51] (central difference schemes), Bouchut et al. [6] (kinetic BGK schemes), Afif and Amaziane [1] and Ohlberger, Gallouet et al. [29, 70] (finite volume methods), and Cockburn and Shu [16] (discontinuous Galerkin methods). Rigorous estimates of the convergence rate of finite volume schemes for degenerate parabolic equations were proved in [45, 46]. More recently, convection-diffusion equations have also been explored where  $f(u)$  and  $A(u)$  as well as initial data are assumed to be known only in a stochastic sense. The problem of random initial data with  $a(u) = 0$  was considered in [66], and the existence and uniqueness of a random entropy solution was shown as well as convergence for Monte Carlo based finite volume discretizations. In [48] the mathematical framework was extended to include degenerate convection diffusion equations with random convective and diffusive flux functions with possibly correlated random perturbations.

### 1.3 The inverse problem of (1.1) when $a(u) = 0$

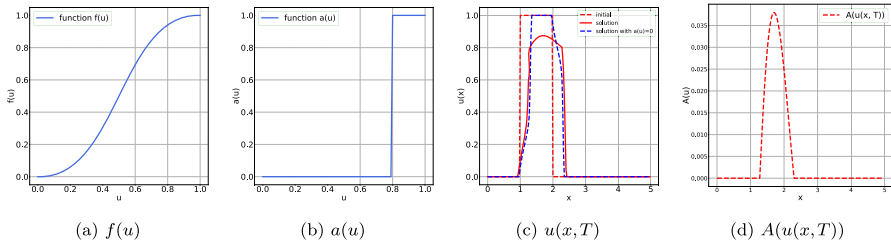
Previous research on inverse problems for nonlinear conservation laws where  $a(u) = 0$  includes a sophisticated methodology introduced by James and Sepúlveda [40, 41]. They formulated the identification of  $f$  as an optimization problem with a cost functional  $J$  that measures the distance between the observed solution and the PDE solution at a fixed time point. Under the assumption of a strictly convex flux  $f$  (and  $a(u) = 0$ ), Kang and Tanuma [44] show that  $f$  can be identified uniquely from initial data with compact support and observations of a single shock that is formed after a sufficiently large time.

Over the past decade, the rapid advancement in machine learning, data science, and computing power has led to the widespread adoption of modern machine learning techniques, particularly deep neural network models, in various fields of research and application. These techniques have proven particularly effective in the context of recovering ordinary differential equations (ODEs) [12, 13, 15, 21, 35, 76] and partial differential equations (PDEs) [7, 8, 30, 32, 33, 43, 53, 73, 89]. These developments have opened up new avenues for data-driven modeling and have enabled researchers to extract meaningful insights from complex, high-dimensional data. Raissi et al. [73] introduced physics informed neural network (PINN) for solving two main classes of problems: data-driven solution and data-driven discovery of partial differential equations. They suggested that if the considered PDE is well-posed and its solution is unique, then the PINN method is capable of achieving good predictive accuracy given a sufficiently expressive neural network architecture and a sufficient number of collocation points. The method was explored for Schrödinger equation, Allen-Cahn equation, and Korteweg-de Vries (KdV) in one dimension (1D) and Navier–Stokes in two dimensions (2D). However, the neural network methods are not straightforward to use in the context of nonlinear hyperbolic PDE that governs two-phase transport in porous media [30]. The authors experimentally indicate that this shortcoming of PINN for hyperbolic PDEs is related to the lack of regularity in the solution. For the same reason it is natural to assume that the fact that solutions of (1.1) cannot be understood in a classical sense but in the sense of (1.3), hampers use of various modern neural network-based methods.

### 1.4 Problem statement and purpose of this work

In this work focus is on learning the functional form of both the flux function  $f(u)$  and the diffusion function  $A(u)$  in the degenerate convection–diffusion model (1.1), where  $u$  is the primary variable. Main challenges associated with that problem is:

- (i) It is well known that the solutions of (1.1) typically develop discontinuities in accordance with the formulation (1.3) [10, 27, 86], as expressed by (1.4) and (1.5). This lack of regularity prevents direct use of various modern learning methods mentioned above.
- (ii) As jumps arise and disappear in the solution over the time period for which observation data is collected when  $a(u)$  vanishes over intervals in  $u$ , the data may lack information about  $f(u)$  in these intervals [53]. In the current problem, there is a



**Fig. 1** **a** Nonlinear flux function  $f(u) = \frac{u^2}{u^2+(1-u)^2}$ . **b** Diffusion function (1.6). **c** The solution  $u(x, T)$  of (1.1) at time  $T = 0.25$  is shown (red solid curve) together with its initial data  $u_0(x)$  (red dashed line). Corresponding solution of (1.1) with  $a(u) = 0$  is also shown (blue dashed line). **d** Illustration of  $A(u(x, T))$  with  $T = 0.25$  (color figure online)

blend of effects from  $f(u)$  and  $A(u)$  present in the observation data that might further complicate the identification of  $f$  and  $A$ . An illustration of this situation is given in Fig. 1. In panel (a) and (b), respectively, we plot the flux function  $f(u) = u^2/(u^2 + (1 - u)^2)$  and the diffusion function

$$a(u) = \begin{cases} 0.0, & \text{if } u \in [0, 0.8] \\ 1.0, & \text{otherwise.} \end{cases} \tag{1.6}$$

In panel (c) the entropy solution after a time  $T = 0.25$  is shown (red solid line). At the time  $t = 0$ , the initial data  $u_0(x)$  involves one jump at  $x = 1$  and another jump at  $x = 2$ . The initial jump at  $x = 1$  is instantly transformed into a solution that is a combination of a continuous wave solution and a discontinuous wave  $(u_l, u_r) \approx (0.4, 0.8)$ . Similarly, the initial jump at  $x = 2$  is transformed into a solution that is a combination of a continuous wave solution and a discontinuous wave  $(u_l, u_r) \approx (0.8, 0)$ . From this example, we see that we have no observation data that directly can reveal the shape of  $f(u)$  and  $a(u)$  in the interval  $u \in [0.4, 0.8]$  (approximately). In panel (c) we also include the solution with  $a(u) = 0$  (dashed blue line). In particular, we see that the evolving discontinuities are different as the discontinuities obeying the entropy condition (1.5) clearly are affected by the presence of the diffusive flux term  $A(u)$ , see a plot of it at time  $T = 0.25$  in panel (d).

(iii) A third issue is the role played by selecting different types of observation data. Are some observation data better than others to identify the unknown  $f(u)$  and  $A(u)$ ?

The methodology we rely on represents a generalization from the pure conservation law case with  $a(u) = 0$  explored in [53, 55] to the degenerate convection-diffusion case (1.1). The main contribution of this work includes:

(i) We introduce a novel approach for learning the functional forms of both the flux function  $f(u)$  and diffusion function  $A(u)$  in the degenerate convection-diffusion model (1.1), where  $u$  is the primary variable. Our approach is based on combining symbolic multilayer neural network functions to represent  $f(u)$  and  $A(u)$  with an entropy consistent discrete numerical scheme (ECDNS) for (1.1) which is known

to converge to the correct entropy solution [27]. Relying on observation data of the form  $u(x_j, t_i)$  at fixed positions  $(x_j, t_i)$ , this allows us to accurately learn both the unknown flux and diffusion functions using significantly less observational data compared with, for example, [88].

- (ii) Going beyond the reliance on the equation's solution in terms of  $u(x_j, t_i)$  as observational data, we investigate the same physical system from a distinct perspective. Specifically, we consider a scenario where the system is made up of particles that follow the flow field as described by  $u(x, t)$  governed by (1.1). That is, particle  $z_j(t)$  is governed by

$$\dot{z}_j(t) = w(u(z_j(t), t)), \quad z_j(t = 0) = x_j^0$$

for a given function  $w(v)$ . Using particle-based observational data, the proposed method generates quite remarkable outcomes despite the fact that the number of observation data points is much lower than in point (i).

- (iii) Representation of unknown functions can take many forms, and in this paper, we employ the neural network S-Net as our chosen method. In addition, we conduct a comparative experiment using the piecewise affine functions method described in [24]. The experimental results demonstrate the clear advantage of neural networks in terms of effective and robust identification of the unknown functions  $f$  and  $A$ .
- (iv) The study provides insight into the relative impact from the flux function  $f(u)$  and diffusion function  $A(u)$ . In particular, accurate learning of  $a(u)$  is difficult as the impact from  $a(u)$  goes through the integrated form  $A(u) = \int^u a(s) ds$ . More precisely, since the total flux  $f(u) - A(u)_x$  strongly impacts the evolution of (1.1), accurate identification of  $f(u)$  and  $A(u)$  is obtained, whereas the corresponding  $A'(u) = a(u)$  may differ from the ground truth  $a$  behind the observation data.

## 2 Related work

The inverse problem of flux identification, which involves minimizing a suitable cost function, was formulated by James and Sepúlveda [42]. They demonstrated, through the use of the viscous approximation, that the perturbed problem converges to the original hyperbolic problem by allowing the viscous term to vanish. Holden et al [38] employed the front-tracking algorithm to reconstruct the flux function from observed solutions with appropriate initial data. More recently, there have been several studies aimed at reconstructing the flux function for sedimentation problems, where a flocculated suspension separates into a concentrated sediment and a clear fluid [9, 23]. In [24], Diehl proposed a direct inversion method that utilizes linear combinations of finite element hat functions to represent unknown flux and diffusion functions. In our paper, we use this approach as a comparative method.

The last decade has seen remarkable progress in the field of machine learning which has enabled several notable efforts aimed at uncovering the governing equations for nonlinear dynamical systems. Among these, the SINDy (Sparse Identification of Nonlinear Dynamics) model [8] has emerged as a breakthrough tool for addressing this challenge. SINDy was developed to recursively identify the sparse solution from a pre-

defined basis function library using a sequential threshold ridge regression algorithm. SINDy quickly emerged as one of the leading methods in this field of study, igniting significant interest [11, 14, 20, 69, 79]. However, the success of this sparsity-promoting approach relies on a well-defined candidate function library, which requires prior knowledge of the system. It is also limited by the possibility that a linear combination of candidate functions may be insufficient to capture complex mathematical expressions. Furthermore, as the library size increases, it may fail to maintain the sparsity constraint, according to empirical observations.

A symbolic regressor offers great flexibility in model selection by allowing the free combination of mathematical operators and symbols to identify the governing equations that best describe the underlying physics. This approach to data-driven nonlinear dynamics discovery was first introduced in [5] and [80] that recast the problem as a symbolic regression task and used genetic programming (GP) to solve it [4]. This idea has inspired a series of subsequent endeavors [18, 34, 84]. More recently, Deep Symbolic Regression (DSR) [68, 71] was introduced as a reinforcement learning-based model that generally outperforms GP-based models. However, this approach is built upon ad-hoc steps and, to some extent lacks flexible automation in equation discovery.

Mesh-based simulations have made significant progress recently [33, 39, 72, 89], offering faster runtimes than principled solvers and greater adaptivity to the simulation domain compared to grid-based convolutional neural networks (CNNs) [83, 87]. However, the efficacy of this method was tested on one-dimensional conservation law problems by [56], revealing that when the parameters deviate significantly from the training parameters, the predictions have a large deviation.

Attempts have been made to tackle the nonlinear dynamics discovery problems by introducing neural networks with activation functions replaced by commonly seen mathematical operators [47, 62, 63, 65, 78]. Long et al [62, 63] proposed a combination of numerical approximation of differential operators by convolutions and a symbolic multi-layer neural network for model recovery. They used convolutions to approximate differential operators with properly constrained filters and to approximate the nonlinear response by deep neural networks. The intricate formulas are obtained via symbolic expansion of the well-trained network. This interpretation of physical laws results in larger candidate pools compared with the library-based representation of physics employed by SINDy. This type of approach has been demonstrated to be effective in tackling conservation law problems, as evidenced in previous research [53, 55] as well as ODEs [54].

Magiera et al. [64] investigated methods for building a reliable surrogate Riemann solver based on neural networks that incorporate conservation properties, such as mass or momentum conservation. In [3], Deniz et al. propose a data-driven physics-informed finite-volume scheme for the approximation of small-scale dependent shocks. A convolutional neural network combines local interpolation polynomials with a regularization term to form the adaptive nonlinear numerical flux in this numerical scheme. Reinforcement learning was utilized by Wang et al. in [88] to identify novel and potentially better data-driven solvers for conservation laws. A multilayer perceptron (MLP) is explored in [74]. It is trained offline using a supervised learning strategy and then employed as a universal troubled-cell indicator that can be used for generic con-

servation laws. In addition, a few recent studies have exploited machine learning to auto-discover conservation laws [60, 61, 67].

### 3 Method

In this section, we introduce our proposed method. Firstly, we present the two important components of the model: ECDNS and S-Net. Secondly, we illustrate how to combine ECDNS with S-Net in our model.

#### 3.1 Entropy consistent discrete numerical scheme (ECDNS)

We discretize the spatial domain  $[0, L]$  into  $N_x$  points  $\{x_i\}_{i=1}^{N_x}$ , where  $x_i = (1/2+i)\Delta x$  for  $i = 1, \dots, N_x$ , and  $\Delta x = L/N_x$ . Additionally, we consider a set of time steps  $\{t^n\}_{n=0}^{N_t}$  with  $N_t \Delta t = T$ . To discretize (1.1), we use the Rusanov scheme [52], which can be written as follows:

$$\begin{aligned} u_j^{n+1} &= u_j^n - \lambda(F_{j+1/2}^n - F_{j-1/2}^n) + \lambda(D_+ A_{j+1/2}^n - D_+ A_{j-1/2}^n), \quad \lambda = \frac{\Delta t}{\Delta x} \\ u_1^{n+1} &= u_2^{n+1}, \quad u_{N_x}^{n+1} = u_{N_x-1}^{n+1} \end{aligned} \tag{3.1}$$

with  $j = 2, \dots, N_x - 1$  and  $F_{j+1/2}^n$  and  $D_+ A_{j+1/2}^n$  take the forms

$$F_{j+1/2}^n = \frac{f(u_j^n) + f(u_{j+1}^n)}{2} - \frac{M_{j+1/2}^n}{2}(u_{j+1}^n - u_j^n), \tag{3.2}$$

and

$$D_+ A_{j+1/2}^n = \frac{A(u_{j+1}^n) - A(u_j^n)}{\Delta x} = \frac{\int_0^{u_{j+1}^n} a(z) dz - \int_0^{u_j^n} a(z) dz}{\Delta x}. \tag{3.3}$$

We adopt a local estimation by using  $M_{j+1/2}^n = \max\{|f'(u_j^n)|, |f'(u_{j+1}^n)|\}$ . The CFL condition determines the magnitude of  $\Delta t$  for a given  $\Delta x$ ,

$$\text{CFL} := \frac{\Delta t}{\Delta x} \left( M + \frac{2K}{\Delta x} \right) \leq 1, \quad M = \max_u |f'(u)|, \quad K = \max_u a(u).$$

We apply the CFL condition in Algorithm 1 to ensure numerical stability. To learn the solution  $U^n = \{u(x_j, t^n)\}_{j=1}^{N_x}$  of the discrete conservation law (3.1), we employ Algorithm 2.

#### 3.2 S-Net

To identify the unknown functions, we have opted for machine learning methods, which offer greater robustness and flexibility. In particular, we are using S-Nets, as employed in [62, 63, 65, 78]. Figure 2 depicts the building of the S-Net with three



**Algorithm 1 CFL**

**Input:**  $L$ : length of the spatial domain;  $N_x$ : the number of  $x$  spatial grid cells;  $f(u)$ : the nonlinear flux function;  $a(u)$ : the diffusion function;  $T$ : computational time period;

**Output:**  $\Delta t$ : local time interval

$$\Delta x = L/N_x$$

$$K = \max_u a(u)$$

$$M = \max_u |f'(u)|$$

$$dt = (\frac{3}{4}\Delta x)/(M + 0.0001 + 2K/\Delta x)$$

$$n\_time = \lfloor T/dt \rfloor$$

$$\Delta t = T/n\_time$$

**Algorithm 2 DataGenerator**

**Input:**  $T$ : computational time period;  $N_x$ : the number of  $x$  spatial grid cells;  $L$ : length of the spatial domain;  $u^0 = \{u^0(x_i, t = t_0)\}_{i=1}^{N_x}$ : initial state of dimension  $N_x$ ;  $f(u)$ : the flux function;  $a(u)$ : the diffusion function

**Output:**  $U = \{u^n\}$ : the solution based on initial state  $u^0$ ;

$$\Delta t = CFL(L, N_x, f(u), a(u), T)$$

$$\Delta x = L/N_x$$

$$U[0] = u^0$$

$$\tilde{u} = u^0$$

**for**  $i = 1, \dots, T/\Delta t$  **do**

**for**  $j = 1, \dots, N_x - 1$  **do**

$$F_{j+1/2} = \frac{1}{2} (f(\tilde{u}_j) + f(\tilde{u}_{j+1})) - \frac{\max\{|f'(\tilde{u}_j)|, |f'(\tilde{u}_{j+1})|\}}{2} (\tilde{u}_{j+1} - \tilde{u}_j)$$

$$D+A_{j+1/2} = \frac{\int_0^{\tilde{u}_{j+1}} a(z)dz - \int_0^{\tilde{u}_j} a(z)dz}{\Delta x}$$

**end**

**for**  $j = 2, \dots, N_x - 1$  **do**

$$u_j = \tilde{u}_j - \frac{\Delta t}{\Delta x} (F_{j+1/2} - F_{j-1/2}) + \frac{\Delta t}{\Delta x} (D+A_{j+1/2} - D+A_{j-1/2})$$

**end**

$$u_1 = u_2$$

$$u_{N_x} = u_{N_x-1}$$

$$\tilde{u} = u$$

$$U[i] = u$$

**end**

hidden layers. The linear combination map uses parameters  $\mathbf{w}_1$  and  $\mathbf{b}_1$  to choose two elements from  $u$  and are denoted by  $\alpha_1$  and  $\beta_1$

$$(\alpha_1, \beta_1)^T = \mathbf{w}_1 \cdot (u) + \mathbf{b}_1, \mathbf{w}_1 \in \mathbb{R}^{2 \times 1}, \mathbf{b}_1 \in \mathbb{R}^{2 \times 1}. \tag{3.4}$$

These two elements of  $\alpha_1$  and  $\beta_1$  are multiplied which give

$$f_1 = \alpha_1 \beta_1. \tag{3.5}$$

Apart from  $u$  gotten by the identity map,  $f_1$  also is input to the second hidden layer

$$(\alpha_2, \beta_2)^T = \mathbf{w}_2 \cdot (u, f_1)^T + \mathbf{b}_2, \mathbf{w}_2 \in \mathbb{R}^{2 \times 2}, \mathbf{b}_2 \in \mathbb{R}^{2 \times 1}. \tag{3.6}$$

Similarly, with the first hidden layer, we get another combination  $f_2$

$$f_2 = \alpha_2 \beta_2. \tag{3.7}$$

Then we obtain  $\alpha_3$  and  $\beta_3$  by means of  $\mathbf{w}_3$  and  $\mathbf{b}_3$  from  $u, f_1$  and  $f_2$

$$(\alpha_3, \beta_3)^T = \mathbf{w}_3 \cdot (u, f_1, f_2)^T + \mathbf{b}_3, \mathbf{w}_3 \in \mathbb{R}^{2 \times 3}, \mathbf{b}_3 \in \mathbb{R}^{3 \times 1}. \tag{3.8}$$

$f_3$ , which is the product of  $\alpha_3$  and  $\beta_3$  is put into the third hidden layer

$$f_3 = \alpha_3 \beta_3. \tag{3.9}$$

Finally, we arrive at the analytic expression of the function  $f$

$$f = \mathbf{w}_4 \cdot (u, f_1, f_2, f_3)^T + \mathbf{b}_4, \mathbf{w}_4 \in \mathbb{R}^{1 \times 4}, \mathbf{b}_4 \in \mathbb{R}. \tag{3.10}$$

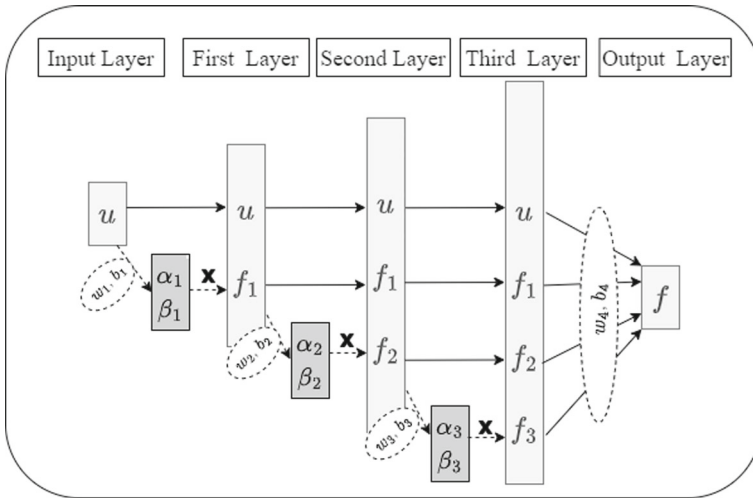
The weight and bias parameters of S-Net for this example are  $\theta = \{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4\}$ . The core of the method is the appropriate choice of  $\theta$ . If S-Net has  $n$  hidden layers, it is capable of learning a polynomial of order  $2^n$ . The complexity of S-Net is determined by the weights and biases across its structure. Specifically, the weights  $\mathbf{w}$  follow a pattern of  $1 \times 2 + 2 \times 2 + 2 \times 3 + \dots + 2 \times n + (n + 1)$ , while the biases  $\mathbf{b}$  are summed up as  $1 \times 2 + 2 + 3 + \dots + n + 1$ . This structure highlights the exponential increase in the order of the polynomial it can learn, directly correlating with the depth of the network through its hidden layers. In theory, increasing the number of hidden layers enhances the model’s representational flexibility. Yet, an excess of hidden layers complicates optimization. Thus, we aim for a minimal number of hidden layers while maintaining sufficient expressive capability. In this paper, focus is on studying flux functions that can be approximated well within the class of polynomial functions. However, to enhance its learning capabilities, we can readily incorporate additional operators, such as those for sine and cosine. This is among the reasons why we opted for S-Net in our study.

Besides S-Nets, other classical methods explicitly represent functions, such as the piecewise affine functions method described in [24]. In the piecewise affine functions method, divide the  $u$ -axis into  $n$  equidistant intervals: let  $u^{max}$  be the largest value of the data and set  $u_k = ku^{max}/n$  for  $k = -1, \dots, n + 1$ . Assume that

$$\hat{f}(u) = \sum_{k=0}^n f_k \psi_k(u) \tag{3.11}$$

where  $f_k$  is  $n + 1$  parameters to be determined, and the hat functions are defined by

$$\psi_k(u) = \begin{cases} \frac{u-u_{k-1}}{u_k-u_{k-1}}, & u_{k-1} < u \leq u_k \\ \frac{u_{k+1}-u}{u_{k+1}-u_k}, & u_k < u \leq u_{k+1} \\ 0, & \text{otherwise} \end{cases} \tag{3.12}$$



**Fig. 2** The structure of S-Net with three hidden layers.  $\alpha_i, \beta_i$  are terms with respect to  $u$ , obtained by the parameters  $w_i, b_i, i = 1, 2, 3$

where  $k = 0, \dots, n$ . In this work, we employ the piecewise affine functions method as a means of comparison.

### 3.3 The model

The model combines two classes of  $S - Nets$ ,  $S - Net_f$  and  $S - Net_a$ , and ECDNS to learn  $f(u)$  and  $a(u)$ . We employ a single S-Net, denoted as  $f_\theta(u)$ , to learn  $f(u)$  across the entire space  $u \in [0, 1]$ . However, we utilize  $N_a$  number of S-Nets distributed across different intervals  $\{[\frac{k}{N_a}, \frac{k+1}{N_a}] | k = 0, 1, \dots, N_a - 1\}$ , to learn  $a(u)$ , designated as  $a_\theta(u)$ ,

$$a_\theta(u) = \sum_{k=0}^{N_a-1} a_{\theta_k}(u) I_k(u), \tag{3.13}$$

where

$$I_k(u) = \begin{cases} 1, & \frac{k}{N_a} \leq u < \frac{k+1}{N_a} \\ 0, & \text{otherwise.} \end{cases} \tag{3.14}$$

In order to identify unknown equations, a commonly used approach is to rely on observed data in the form of solutions  $u(x, t)$  to the equations, as demonstrated in prior work [24, 53, 55, 62, 63]. In this work, we supplement this standard observational dataset with another set of observations obtained from particle trajectories.

- *Observation Data I (standard)*. We consider observation data in terms of  $x$ -dependent data at fixed times  $\{t_i^*\}_{i=1}^{N_{\text{obs}}}$  extracted from the solution  $U$  as follows:

$$U_{\text{sub}} = \{u(x_j, t_1^*), u(x_j, t_2^*), \dots, u(x_j, t_{N_{\text{obs}}}^*)\}, \quad j = 1, \dots, N_x. \tag{3.15}$$

(3.15) is utilized to select synthetic observation data denoted by  $U_{\text{sub}}$  as well as predictions based on the learned  $f_{\theta_f}(u)$  and  $a_{\theta_a}(u)$  written as  $\hat{U}_{\text{sub}}$ . We specify times for collecting the time-dependent data

$$T_{\text{obs}} = \{t_i^* = i \Delta t^{\text{obs}} : i = 1, \dots, N_{\text{obs}}\}. \tag{3.16}$$

Typically, we set  $N_x = 200$  and  $N_{\text{obs}} = 9$  with  $\Delta t^{\text{obs}} = 0.1$ .

– *Observation Data II (particle trajectories).*

In addition to studying the conservation law (1.1) from the standard perspective, we also investigate the same physical system from a different vantage point. Specifically, we consider a scenario where the system consists of particles that move with the flow field described by (1.1). Instead of examining the solution  $u(x, t)$  as observation data, we focus on the trajectories of these particles as it has been suggested in the context of traffic flow for a scalar nonlinear conservation law [17, 58, 75]. The classical traffic flow model takes the following form

$$u_t + (uw(u))_x = 0. \tag{3.17}$$

The flux function in this case is  $f(u) = uw(u)$ . The work [17] suggests to study an ODE of the form  $\dot{z}(t) = w(u(z(t), t))$  for a suitable known speed function  $w$  and a prescribed initial position to represent the path  $z(t)$  of one driver that travels along the road modelled by (3.17). The authors demonstrate the existence and stability of particle trajectories in relation to their initial positions. Additionally, [25] establishes several novel findings regarding the stability of particle trajectories when subjected to variations in the initial field.

We may apply the same approach based on the model (1.1). Let us suppose a single particle  $j$  begins at time  $t_0$  from a location  $x_j^0$ , and let  $z_j(t)$  denote its position at time  $t$ . The velocity field of the flow is represented by a function  $w$  whose functional form is assumed known. Without loss of generality, we will use  $w(u) = u$  in the following. Then  $z_j(t)$  satisfies the following equation

$$\dot{z}_j(t) = w(u(z_j(t), t)), \quad j = 1, \dots, N_{\text{par}} \tag{3.18}$$

where  $u(x, t)$  is the entropy solution of (1.1) as captured by the entropy satisfying scheme (3.1)–(3.3). Equation (3.18) can be interpreted as an ordinary differential equation (ODE) for  $z_j(t)$ . If we take the initial position  $x_j^0$  of particle  $j$  into account then we have the following Cauchy initial data to (3.18):

$$z_j^0(t = t_0) = x_j^0, \quad j = 1, \dots, N_{\text{par}}. \tag{3.19}$$

To discretize (3.18), we use the Euler method [2], which can be written as follows:

$$z_j^{n+1} = z_j^n + w(u(z_j^n, t^n)) \Delta t^{\text{ode}} \tag{3.20}$$

where  $\Delta t^{\text{ode}}$  refers to the time stepping used to solve the ODE. Our observational data is the trajectories of  $N_{\text{par}}$  particles. Specifically, we will denote the initial

positions of these  $N_{par}$  particles at time  $t_0$  by

$$Z^0 = \{x_1^0, x_2^0, \dots, x_{N_{par}}^0\} = \{z_1^0, z_2^0, \dots, z_{N_{par}}^0\}. \tag{3.21}$$

Based on the solution  $U$  derived from Algorithm 2, we interpolate position  $Z^0$  to obtain the corresponding solution  $U^{Z^0}$  at the time  $t_0$ . According to (3.20), we obtain the particle positions at the subsequent time point  $t_0 + \Delta t^{ode}$ ,

$$Z^1 = Z^0 + w(U^{Z^0})\Delta t^{ode}. \tag{3.22}$$

Additionally, by interpolating position  $Z^1$ , we derive the corresponding solution  $U^{Z^1}$  at the time  $t_0 + \Delta t^{ode}$ . Hence, we can get the particle positions at the subsequent time point  $t_0 + 2\Delta t^{ode}$

$$Z^2 = Z^1 + w(U^{Z^1})\Delta t^{ode}. \tag{3.23}$$

It should be noted that we set  $\Delta t^{ode} = \Delta t$  gotten from Algorithm 2, i.e., we use the same time step as for the evolution of the discrete scheme that solves (1.1). We also set  $t_0 = 0$ . This iterative process proceeds, enabling us to derive the trajectories of all particles

$$Z = \{Z^i = \{z_1^i, z_2^i, \dots, z_{N_{par}}^i\} | i = t_0 + \hat{i}\Delta t^{ode}, \hat{i} = 0, 1, 2, \dots\}. \tag{3.24}$$

For the evaluation of  $U^{Z^n}$  associated with time  $t^n = n\Delta t$  based on the discrete solution  $\{u_j^n\}$  generated by Algorithm 2, we use linear interpolation. We consider observation data at fixed times  $\{t_i^*\}_{i=1}^{N_{obs}}$  as given by (3.16) extracted from  $Z$  as follows:

$$Z_{sub} = \{Z^{t_1^*}, Z^{t_2^*}, \dots, Z^{t_{N_{obs}}^*}\}. \tag{3.25}$$

Equation (3.25) is utilized to select synthetic observation data denoted by  $Z_{sub}$  as well as predictions based on the learned  $f_{\theta_f}(u)$  and  $a_{\theta_a}(u)$  written as  $\hat{Z}_{sub}$ .

Standard Observation (Data I) and Particle Trajectories (Data II) represent two data sets that study conservation laws from different perspectives. Furthermore, Particle Trajectories utilize a significantly smaller amount of data for model training compared to Standard Observation data. Specifically, while Standard Observation data employs  $N_x = 200$  grid points when the number of observation time points is the same, Particle Trajectories only utilize  $N_{par} = 10$  particle trajectories.

During model training,  $f_{\theta_f}(u)$  and  $a_{\theta_a}(u)$ , in place of  $f(u)$  and  $a(u)$ , are fed into Algorithm 2 to obtain the predicted solution  $\hat{U}$ . Similar to  $U_{sub}$  and  $Z_{sub}$ , we select predicted solutions  $\hat{U}_{sub}$  and  $\hat{Z}_{sub}$  according to (3.15) and (3.25). The difference

between  $U_{sub} (Z_{sub})$  and  $\hat{U}_{sub} (\hat{Z}_{sub})$ , denoted as  $L^{data}$ , serves as the loss function,

$$L^{data} = \frac{1}{K_{init} N_x N_{obs}} \sum_{k=1}^{K_{ini}} \sum_{j=1}^{N_x} \sum_{i=1}^{N_{obs}} \left( U_{sub,k}(x_j, t_i^*) - \hat{U}_{sub,k}(x_j, t_i^*) \right)^2, \quad (3.26)$$

or

$$L^{data} = \frac{1}{K_{init} N_{par} N_{obs}} \sum_{k=1}^{K_{ini}} \sum_{j=1}^{N_{par}} \sum_{i=1}^{N_{obs}} \left( Z_{sub,k}(x_j, t_i^*) - \hat{Z}_{sub,k}(x_j, t_i^*) \right)^2. \quad (3.27)$$

To update the parameter vectors  $\theta_f$  and  $\theta_a$ , we employ the second-order quasi-Newton method, L-BFGS-B [77, 90]. Our implementation leverages the PyTorch machine learning framework, simplifying the process to merely invoking the  $L B F G S()$  function provided within PyTorch. By inputting our loss function into  $L B F G S()$ , we efficiently optimize the model’s parameters. The training process of this model based on standard observation data is depicted in Algorithm 3.

---

**Algorithm 3** ConsLaw-Net

---

**Input:**  $T$ : computational time period;  $N_x$ : the number of  $x$  spatial grid cells;  $L$ : length of the spatial domain;  $u^0$ : initial state vector of dimension  $N_x$ ;  $f(u)$ : the true flux function;  $a(u)$ : the diffusion function;  $\theta_f^0$ : initial parameters of S-Net for  $f_\theta(u)$ ;  $\theta_{a\theta_k}^0$ : initial parameters of S-Net for  $a_{\theta_k}$  where  $k = 0, 1, 2, \dots, N_a - 1$ ;  $T_{obs}$ : observation time points (3.16);  $Loss$ : loss function;  $epoch$ : the number of epoch; **DataGenerator**: Algorithm 2;  $I_k(u)$ : as defined in (3.14).

**Output:**  $\theta^*_f; \theta^*_a = \{\theta^*_{a\theta_k} | k = 0, 1, \dots, N_a - 1\}$ ;

$U = \text{DataGenerator}(T, N_x, L, u^0, f(u), a(u))$

$U_{sub} = \{u \in U | t \in T_{obs}\}$

$\theta_f = \theta_f^0, \theta_a = \{\theta_{a\theta_k}^0 | k = 0, 1, \dots, N_a - 1\}$

**for**  $i = 1, \dots, epoch$  **do**

$a_\theta(u) = \sum_{k=0}^{N_a-1} a_{\theta_k}(u) I_k(u)$

$\hat{U} = \text{DataGenerator}(T, N_x, L, u^0, f_{\theta_f}(u), a_{\theta_a}(u))$

$\hat{U}_{sub} = \{u \in \hat{U} | t \in T_{obs}\}$

$loss = Loss(U_{sub}, \hat{U}_{sub})$

Updating  $\theta_f, \theta_a$  and loss by optimizer L-BFGS-B;

**end**

$\theta^*_f = \theta_f, \theta^*_a = \{\theta_{a\theta_k} | k = 0, 1, \dots, N_a - 1\}$

---

## 4 Experiment 1

This section explores a group of nonlinear conservation laws that arises naturally from the vertical displacement of one fluid by another [82]. In this case the the ground truth

**Table 1** The initial states

$u_1^0 = \begin{cases} 1.0 & \text{if } x \in [4, 6] \\ 0 & \text{otherwise} \end{cases}$	$u_2^0 = \begin{cases} 0.9 & \text{if } x \in [4, 6] \\ 0 & \text{otherwise} \end{cases}$	$u_3^0 = \begin{cases} 0.8 & \text{if } x \in [3.5, 5] \\ 0 & \text{otherwise} \end{cases}$
$u_4^0 = \begin{cases} 0.7 & \text{if } x \in [4.5, 6.5] \\ 0 & \text{otherwise} \end{cases}$	$u_5^0 = \begin{cases} 0.85 & \text{if } x \in [0, 2] \\ 0 & \text{otherwise} \end{cases}$	$u_6^0 = \begin{cases} 0.75 & \text{if } x \in [0, 2] \\ 0 & \text{otherwise} \end{cases}$

flux function  $f(u)$  is given by the following flux function [53]

$$f(u) = \frac{1}{2}u(3 - u^2) + \frac{300}{12}u^2 \left( \frac{3}{4} - 2u + \frac{3}{2}u^2 - \frac{1}{4}u^4 \right). \tag{4.1}$$

Our focus is on examining identification of two distinct diffusion functions, namely  $a_1(u)$  given by

$$a_1(u) = \begin{cases} 0, & \text{if } u \in [0, 0.5] \\ 1.0, & \text{otherwise,} \end{cases} \tag{4.2}$$

and  $a_2(u)$  given by

$$a_2(u) = \begin{cases} 0, & \text{if } u \in [0, 0.5] \\ (u - 0.5)u, & \text{otherwise.} \end{cases} \tag{4.3}$$

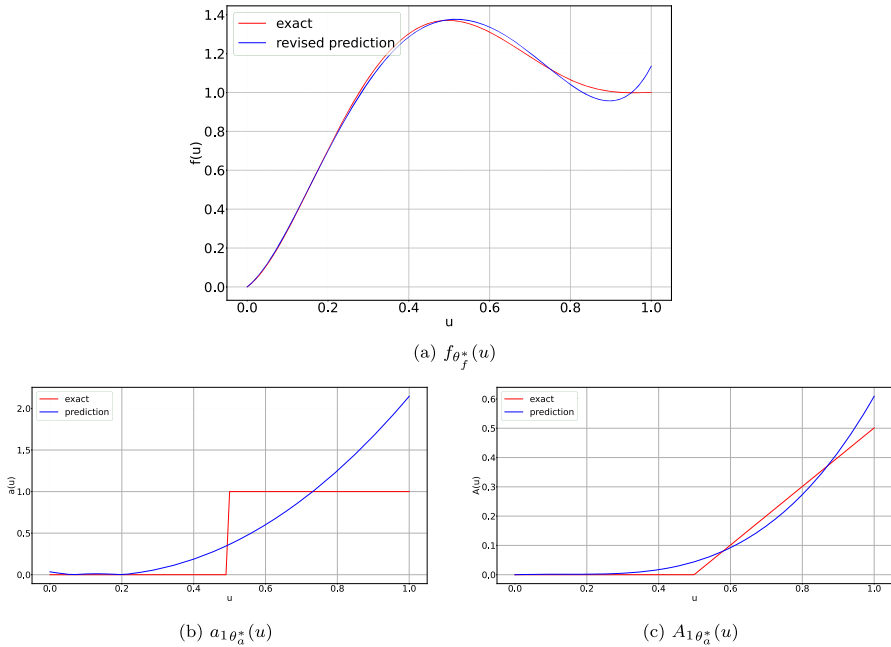
with  $\alpha = 0.5$ . In [22], R. Diaz-Adame et al. presented a discussion on fast and optimal WENO schemes for degenerate parabolic conservation laws with similar diffusion functions.

We examine a one-dimensional spatial domain of length  $L = 10$ , where  $x \in [0, 10]$ , and a time interval  $[0, T]$  with  $T = 2$ . We gather observation data in the form of (3.15) and (3.25), with  $N_{\text{obs}} = 9$ ,  $\Delta t^{\text{obs}} = 0.1$ , and a numerical grid consisting of  $N_x = 200$  grid cells. The goal is to identify the unknown flux function  $f(u)$  and diffusion function  $a(u)$  for  $u \in [0, 1]$ . To learn  $f(u)$  and  $a(u)$ , we consider a set of initial states  $\{u_k^0\}_{k=1}^{K_{\text{init}}}$  such that  $0 \leq u_k^0(x) \leq 1$ . Table 1 displays the initial states utilized in the following experiments.

To represent unknown functions using S-Nets, we employ an S-Net to learn the flux function  $f(u)$ . We evaluate the effectiveness of representing  $a(u)$  using either one or more S-Nets or piecewise affine functions. When using multiple S-Nets, we partition the  $u$ -axis into four equidistant intervals:  $u \in [0, 0.25)$ ,  $u \in [0.25, 0.5)$ ,  $u \in [0.5, 0.75)$ , and  $u \in [0.75, 1.0)$ , and train four distinct S-Nets on each interval. The initial values of the neural networks are randomly generated. In addition, we set  $n = 10$  in (3.11) for the piecewise affine function approach.

### 4.1 Identification of $f(u)$ and $a_1(u)$ using standard observations (Data I) with a single S-Net for representing $a_1(u)$

Figure 3a and b depict the learned  $f_{\theta_f^*}(u)$  and  $a_{1\theta_a^*}(u)$ , respectively. Since the derivative of  $f_{\theta_f^*}(u)$  is what matters for  $f(u)$ , we plot the translated function  $f_{\theta_f^*}(u) - f(0)$ . The



**Fig. 3** The graphical result of identification of flux function  $f_{\theta_f^*}(u)$ , diffusion function  $a_{1\theta_a^*}(u)$  and  $A_{1\theta_a^*}(u)$ . The red line represents the true function, while the blue line illustrates the learned function (color figure online)

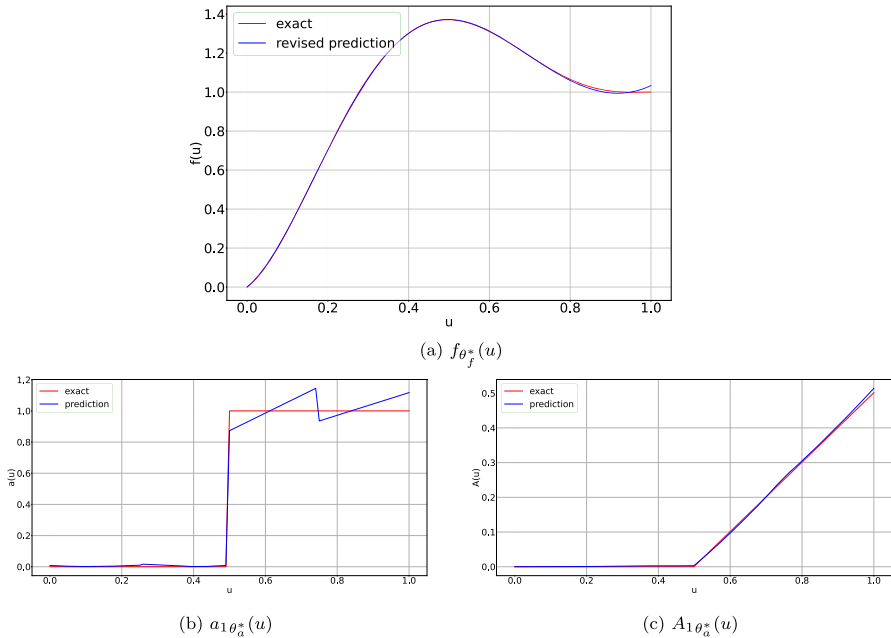
learned  $A_{1\theta_a^*}(u)$  based on  $a_{1\theta_a^*}(u)$  is presented in Fig. 3c. Notably, the predicted  $f(u)$  deviates significantly from the true value at  $u \in [0.6, 1.0]$ , and there is a significant difference between the actual and learned functions for both  $a_1(u)$  and  $A_1(u)$ . Therefore, the piecewise nature of  $a_1(u)$  makes it unsuitable for an S-Net representation as S-Net learns only smooth functions.

### 4.2 Identification of $f(u)$ and $a_1(u)$ using standard observations (Data I) with four S-Nets for representing $a_1(u)$

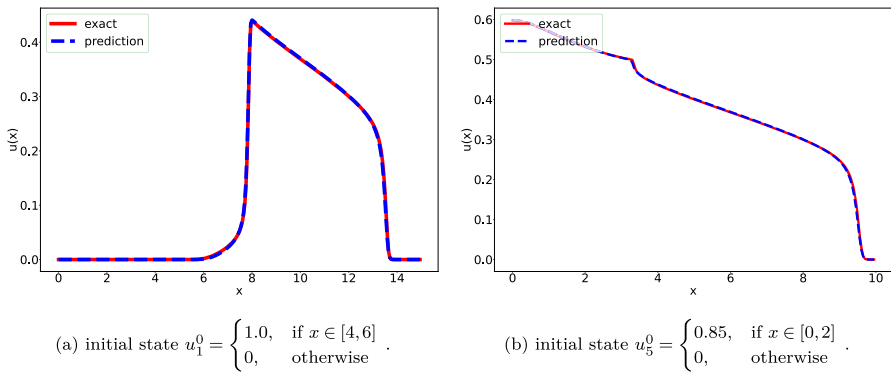
To overcome the limitations of using a single S-Net to represent  $a_1(u)$ , we employed multiple S-Nets to learn  $a_1(u)$ . Specifically, we divided the  $u$ -axis into four intervals, namely  $u \in [0, 0.25]$ ,  $u \in [0.25, 0.5]$ ,  $u \in [0.5, 0.75]$ , and  $u \in [0.75, 1.0]$ , and allocated one S-Net with a single hidden layer for each interval. In contrast, the S-Net used to represent  $f(u)$  consisted of six hidden layers.

Figure 4a–c display the learned functions for  $f_{\theta_f^*}(u)$ ,  $a_{1\theta_a^*}(u)$  and  $A_{1\theta_a^*}(u)$ , respectively. The learned  $f_{\theta_f^*}(u)$  approximates the true function  $f(u)$  closely, but for the diffusion function  $a_1(u)$ , noticeable discrepancies can be observed between the learned and the true functions on  $u \in [0.5, 0.75]$  and  $u \in [0.75, 1.0]$ . However, since the primitive form  $A_1(u)$  is employed in the numerical scheme, we examine the learned function  $A_{1\theta_a^*}(u)$  based on  $a_{1\theta_a^*}(u)$  in Fig. 4c, which remarkably fits the true function



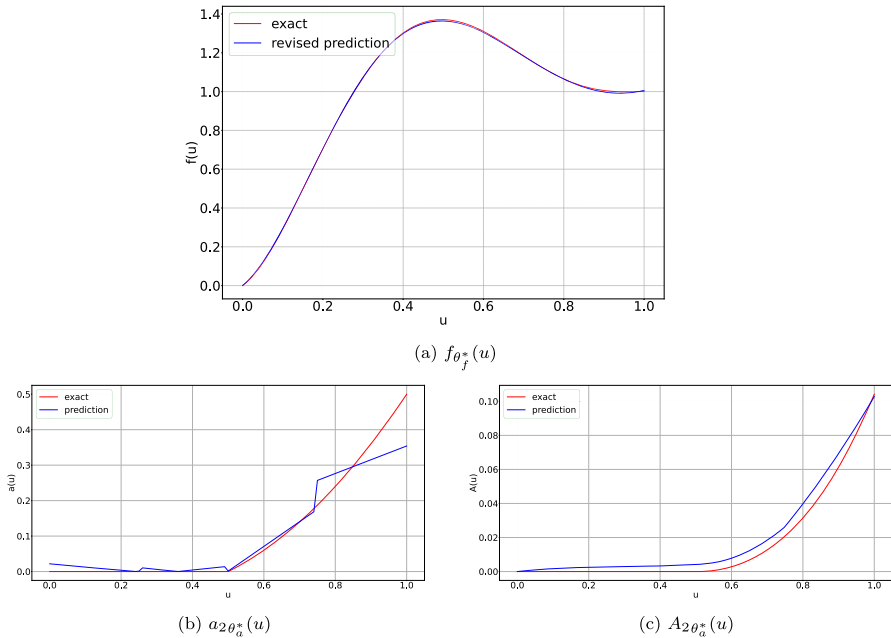


**Fig. 4** The graphical result of identification of flux function  $f_{\theta_f^*}(u)$ , diffusion function  $a_{1\theta_a^*}(u)$  and  $A_{1\theta_a^*}(u)$ . The red line represents the true function, while the blue line illustrates the learned function (color figure online)



**Fig. 5**  $u(x, t)$  at  $t = 2.0$  based on the initial states of  $u_1^0$  and  $u_5^0$  in Table 1. The left and right subplots are based on  $u_1^0$  and  $u_5^0$ , respectively

$A_1(u)$ . Clearly, utilizing multiple S-Nets to learn  $a_1(u)$  can improve overall model performance. In addition, we utilized the learned functions to solve the solutions of (1.1), and the comparison of the true and predicted solutions based on the initial states  $u_1^0$  and  $u_5^0$  in Table 1 is presented in Fig. 5. These results demonstrate the ability of the proposed method to accurately identify unknown functions based on standard observation data.



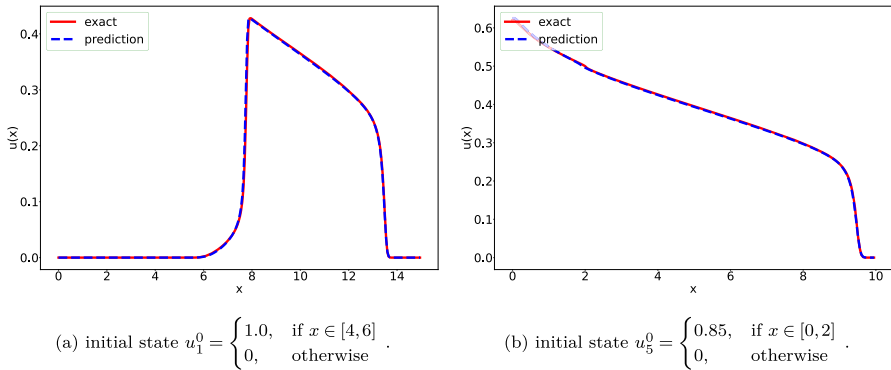
**Fig. 6** The graphical result of identification of flux function  $f_{\theta_f}^*(u)$ , diffusion function  $a_{2\theta_a}^*(u)$  and  $A_{2\theta_a}^*(u)$ . The red line represents the true function, while the blue line illustrates the learned function (color figure online)

### 4.3 Identification of $f(u)$ and $a_2(u)$ using standard observations (Data I) with four S-Nets for representing $a_2(u)$

We learn  $f(u)$  and  $a_2(u)$  using the same experimental setting as in Sect. 4.2. Figure 6a–c show the learned functions  $f_{\theta_f}^*(u)$ ,  $a_{2\theta_a}^*(u)$  and  $A_{2\theta_a}^*(u)$ . While  $f_{\theta_f}^*(u)$  is very similar to  $f(u)$ , there are significant differences between  $a_{2\theta_a}^*(u)$  and  $a_2(u)$  for  $u \in (0.75, 1.0)$ . These differences also affects the learning of  $A_{2\theta_a}^*(u)$  as compared to  $A_{1\theta_a}^*(u)$  in Sect. 4.2. Even with an increase in observation data, we are unable to achieve a good result for  $a_2(u)$ . However, the error associated with  $a_{2\theta_a}^*(u)$  or  $A_{2\theta_a}^*(u)$  does not appear to impact much the predicted solutions, as demonstrated in Fig. 7. This explains why there is room for this error in the identification of  $A_2(u)$ . In Sect. 4.6 we will discuss further why this happens.

### 4.4 Identification of $f(u)$ and $a_1(u)$ using standard observations (Data I) with the piecewise affine functions for representing $f(u)$ and $a_1(u)$

In this section, we replicate the experimental settings of Sect. 4.1 with the exception of the representation of unknown functions which is now based on affine functions as given by (3.11) and (3.12). We set  $n = 10$  in (3.11). We present the learned functions



**Fig. 7**  $u(x, t)$  at  $t = 2.0$  based on the initial states of  $u_1^0$  and  $u_5^0$  in Table 1. The left and right subplots are based on  $u_1^0$  and  $u_5^0$ , respectively

of  $\hat{f}(u)$ ,  $\hat{a}_1(u)$ , and  $\hat{A}_1(u)$  in Fig. 8a–c, respectively. Additionally, Fig. 9 displays the real and predicted solutions based on the initial states  $u_1^0$  and  $u_5^0$  in Table 1.

While  $\hat{f}(u)$  generally fits  $f(u)$  well, there is a significant discrepancy for  $u \in (0.7, 1.0)$ . In addition, since  $f(u)$  is learned by piecewise linear functions,  $\hat{f}(u)$  involve some non-smooth points. These non-smooth points have a large impact on the solutions, causing the volatility observed in Fig. 9a and b.  $\hat{a}_1(u)$  differs significantly from  $a_1(u)$ , especially for  $u \in (0.5, 1.0)$ , see Fig. 8b. As  $A_1(u)$  is an integral function of  $a_1(u)$ , the difference between  $A_1(u)$  and  $\hat{A}_1(u)$  is less apparent than that between  $a(u)$  and  $\hat{a}(u)$ . When comparing the optimization times between the method utilizing piecewise affine functions and the S-Net method, the former required 1879s to complete, whereas the latter took 4901s. This underscores that achieving good results often necessitates more time.

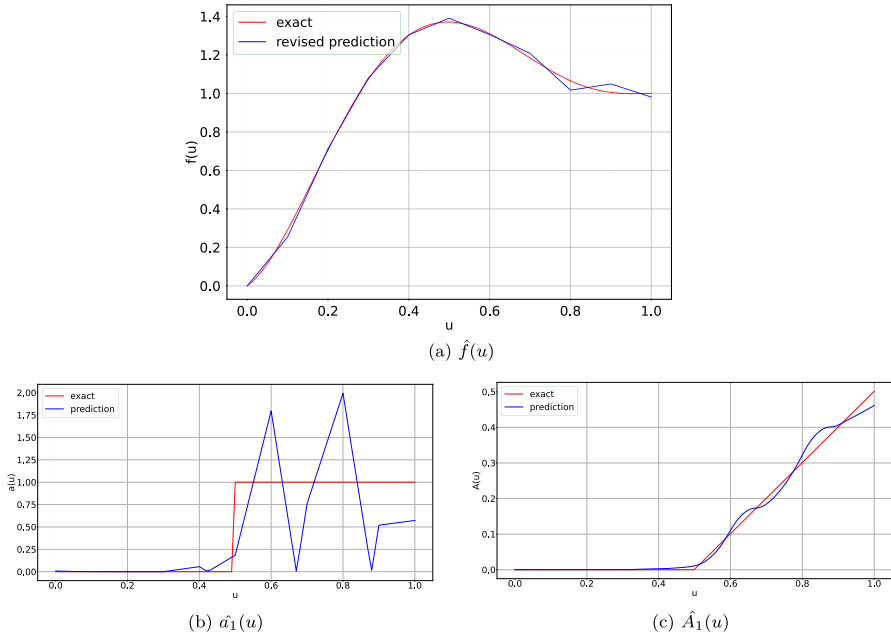
#### 4.5 Identification of $f(u)$ and $a_1(u)$ using particle trajectories (Data II) with four S-Nets for representing $a_1(u)$

In this experiment, we use a six-layer S-Net to represent  $f(u)$ , while the S-Nets used for  $a(u)$  have only one hidden layer. We select ten particles and use their trajectories as observation data. The initial positions of these particles at  $t = 0$  are chosen as follows

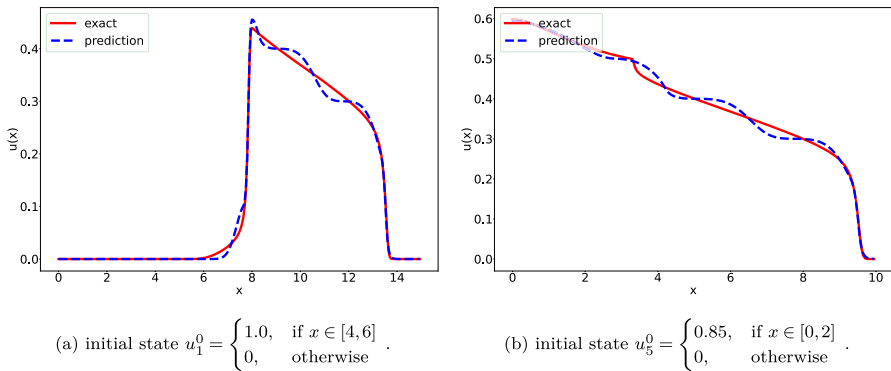
$$Z^0 = \{0.025, 1.025, 2.025, 3.025, 4.025, 5.025, 6.025, 7.025, 8.025, 9.025\}. \tag{4.4}$$

Figure 10 shows the trajectories of different particles in the observation period under different initial states. The three sub-pictures on the left, middle, and right show the real and predicted trajectories of the particles with initial positions of 7.025, 4.025, and 2.025 under the conditions of initial states of  $u_1^0$ ,  $u_2^0$  and  $u_6^0$  respectively.

Figures 11a–c display the learned functions  $f_{\theta_f^*}(u)$ ,  $a_{1\theta_a^*}(u)$  and  $A_{1\theta_a^*}(u)$ . While there is a slight difference between  $f_{\theta_f^*}(u)$  and  $f(u)$  for  $u \in [0.8, 1.0]$ , the approximation is generally close. For  $a_1(u)$ , noticeable discrepancies can be observed between

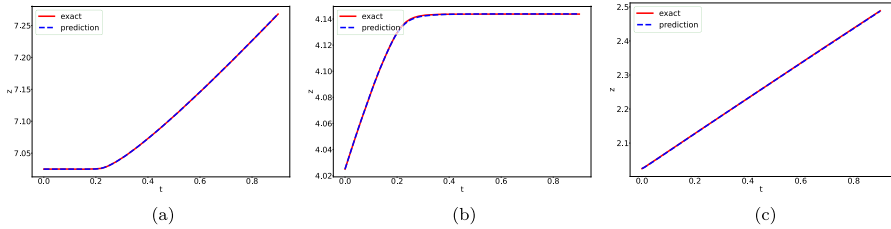


**Fig. 8** The graphical result of identification of flux function  $\hat{f}(u)$ , diffusion function  $\hat{a}(u)$  and  $\hat{A}(u)$ . The red line represents the true function, while the blue line illustrates the learned function (color figure online)

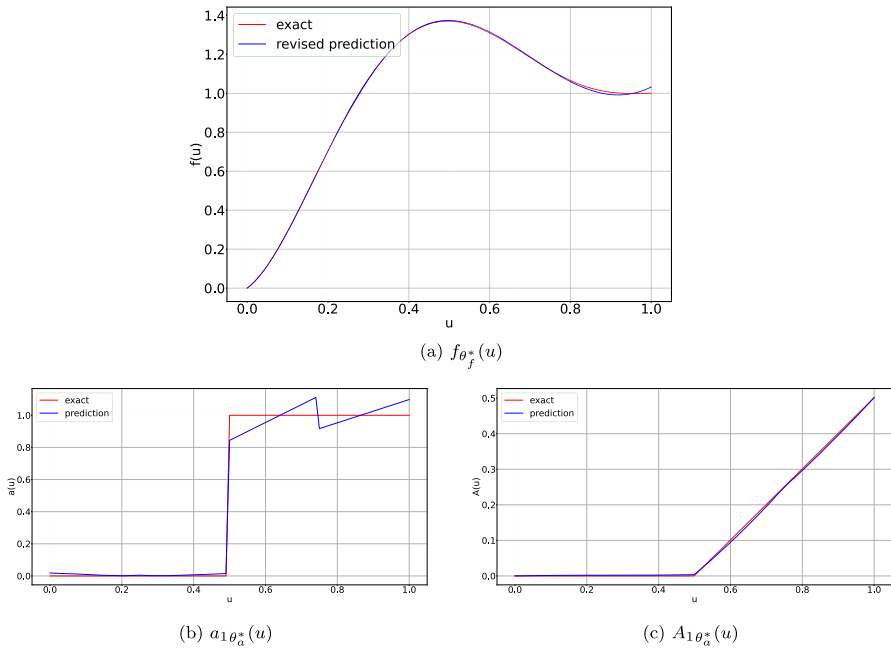


**Fig. 9**  $u(x, t)$  at  $t = 2.0$  based on the initial states of  $u_1^0$  and  $u_5^0$  in Table 1. The left and right subplots are based on  $u_1^0$  and  $u_5^0$ , respectively

the learned and true functions for  $u \in [0.5, 0.75]$  and  $u \in [0.75, 1.0]$ . However,  $A_{1\theta_a^*}(u)$  based on  $a_{1\theta_a^*}(u)$  generally fits the true function  $A_1(u)$  well, except for some smaller fluctuations in  $u \in (0.7, 0.8) \cup (0.9, 1.0)$ . The predicted solutions in Fig. 12 demonstrate that these small deviations in  $f_{\theta_f^*}(u)$ ,  $a_{1\theta_a^*}(u)$  and  $A_{1\theta_a^*}(u)$  have little effect on the predicted solutions. In this section, we use only 5% of the data used in the previous section, but we still get the same good results as seen in Sect. 4.2.



**Fig. 10** The trajectories of different particles in the observation period under different initial states. In each subgraph, the red solid line represents the real trajectory, and the blue dashed line represents the learned trajectory (color figure online)



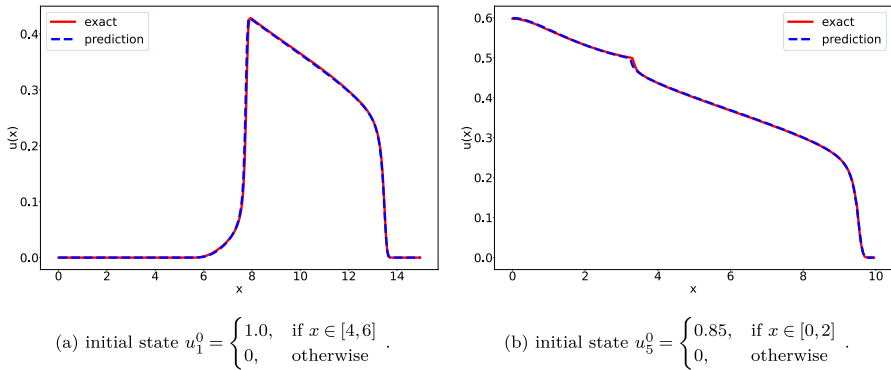
**Fig. 11** The graphical result of identification of flux function  $f_{\theta_f^*}(u)$ , diffusion function  $a_{1\theta_a^*}(u)$  and  $A_{1\theta_a^*}(u)$ . The red line represents the true function, while the blue line illustrates the learned function (color figure online)

**4.6 Discussion**

**4.6.1 Analyzing mean and maximum errors in  $f(u)$ ,  $A(u)$ , and  $u(x, T)$**

We discretize  $u \in [0, 1]$  into  $N_u + 1$  points  $\{u_i\}_{i=0}^{N_u}$ , where  $u_i = i \Delta u$  for  $i = 0, \dots, N_u$ , and  $\Delta u = 1/N_u$ . Let  $N_u = 400$ . We then proceed to quantitatively evaluate the error between the learned and true functions using two specific metrics:

$$Mean = \frac{1}{N_u} \sum_{i=0}^{N_u+1} |f(u_i) - f_{\theta_f^*}(u_i)| \tag{4.5}$$



**Fig. 12**  $u(x, t)$  at  $t = 2.0$  based on the initial states of  $u_1^0$  and  $u_5^0$  in Table 1. The left and right subplots are based on  $u_1^0$  and  $u_5^0$ , respectively

$$Max = \max\{|f(u_i) - f_{\theta^*}(u_i)|, i = 0, 1, 2, \dots\} \tag{4.6}$$

Similarly to  $f(u)$ , we also compute the mean and maximum values of the error about the function  $A(u)$ . Additionally, we compute the mean and maximum values of the absolute difference between the true and predicted solutions at the final time point  $T$ :

$$Mean = \frac{1}{N_x} \sum_{i=1}^{N_x} |u(x_i, T) - \hat{u}(x_i, T)| \tag{4.7}$$

$$Max = \max\{|u(x_i, T) - \hat{u}(x_i, T)|, i = 0, 1, 2, \dots\}, \tag{4.8}$$

where  $x_i$  refers to the  $N_x$  number of grid cells and  $u$  and  $\hat{u}$  are true and predicted solution, respectively. In the Table 3, we analysis Mean and Maximum Errors in  $f(u)$ ,  $A(u)$ , and  $u(x, T)$  learned in Sects. 4.1, 4.2, 4.3, 4.4 and 4.5.

### 4.6.2 Robustness analysis of the model

The flux and diffusion functions are indeed independent of the initial data. In our study, we consistently apply the same set of initial states (referenced as Table 1) across various sub-experiments. The reason is the observation that the data derived from this initial state encompasses a broader spectrum of information pertinent to the unidentified functions. If an alternative set of initial states and their corresponding observations equally encapsulate the essence of these unknown functions, they too could serve as viable starting points for our model training, promising similarly satisfactory outcomes. The distribution of observations is often used to judge whether the observations reflect enough information about the unknown functions [57]. To demonstrate this point, we use a set of other initial states as shown in Table 2. The row labeled “learned functions in Sect. 4.2” in Table 3 (see 2nd row) showcases the outcomes from models trained using observations derived from Table 1. When comparing these results to those obtained from initial states in Table 2 (see 6th row), the

**Table 2** The initial states

$u_1^0 = \begin{cases} 0.95 & \text{if } x \in [4, 6] \\ 0 & \text{otherwise} \end{cases}$	$u_2^0 = \begin{cases} 0.7 & \text{if } x \in [4, 6] \\ 0 & \text{otherwise} \end{cases}$	$u_3^0 = \begin{cases} 0.6 & \text{if } x \in [3.5, 5] \\ 0 & \text{otherwise} \end{cases}$
$u_4^0 = \begin{cases} 0.85 & \text{if } x \in [4.5, 6.5] \\ 0 & \text{otherwise} \end{cases}$	$u_5^0 = \begin{cases} 0.55 & \text{if } x \in [0, 2] \\ 0 & \text{otherwise} \end{cases}$	$u_6^0 = \begin{cases} 0.65 & \text{if } x \in [0, 2] \\ 0 & \text{otherwise} \end{cases}$

error margins for both sets in terms of functions  $f(u)$ ,  $A(u)$ , or solutions  $u(x, T)$  are comparable and fall within an acceptable range.

We adjust the parameters from  $L = 10, T = 2.0$  to  $L = 20, T = 5.0$ , while maintaining the same settings as described in Sect. 4.2. The outcomes of these adjustments are detailed in the row marked “Learned functions with changed  $L$  and  $T$ ” in Table 3 (see 7th row). The performance of  $f(u)$  surpasses that achieved in Sect. 4.2, although it does not excel for  $A(u)$ . Nevertheless,  $f(u)$  demonstrates superior effectiveness in the solutions compared to  $A(u)$ . The errors associated with  $u(x, T)$  in both scenarios are comparable, indicating the algorithm’s resilience to changes in  $T$  and  $L$ .

We tried to randomly add 3% noise to the observation data generated by initial states on Table 1, and then train the model with the noisy data. The result is shown in the row labeled “Learned functions with adding 3% noise to observations” in Table 3.  $f(u)$  and  $A(u)$  are slightly affected by the noise, but the effect on  $u(x, T)$  is ultimately small and within acceptable limits of the error perturbation. If the noise is large, it has a relatively large impact on the results of the model, as shown in the row labeled “Learned functions with adding 9% noise to observations”. In this case, we can minimize the effect of noise on the model by increasing the number of observations [57].

Selecting the optimal number of layers involves a hyperparameter tuning process, guided by the principle of model simplicity. We incrementally adjust the layer count from low to high until the loss function’s final value ceases to improve. It’s important to note that excessively increasing the model’s layers can deteriorate training outcomes due to the complexities introduced in the optimization direction during parameter updates. In Sect. 4.2, the S-Net for representing  $f(u)$  featured six hidden layers. We now employ seven hidden layers to learn  $f(u)$ , keeping all other settings consistent with Sect. 4.2. The outcomes are detailed in the row labeled “Learned functions with seven hidden layers”. Further increasing the number of layers is unlikely to benefit the model significantly.

In Sect. 4.5, the function  $w(u) = u$  was utilized. In this instance, we evaluate  $w(u) = u^2$ , maintaining all other experimental parameters as described in Sect. 4.5. The outcomes are presented in the row titled “Learned functions with  $w(u) = u^2$ .” The performance of these models is nearly identical to those discussed in Sect. 4.5.

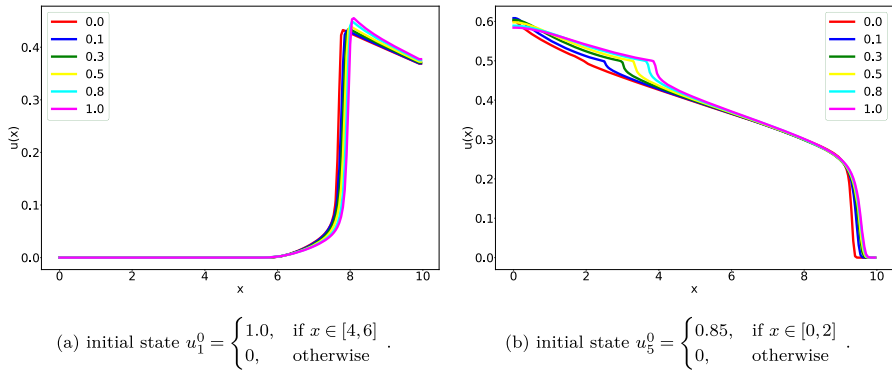
### 4.6.3 The impact of the diffusion functions on solutions

Through the above experiments, we have observed that the learning effect of  $f(u)$  is better than that of  $a(u)$  or  $A(u)$ . While a poorer learning of  $a(u)$  seems to have little impact on the predicted solution, as seen in the experiment in Sect. 4.3, poor learning for  $f(u)$  can cause significant fluctuations in the solution, as shown in the experiment

**Table 3** Analyzing mean and maximum errors in  $f(u)$ ,  $A(u)$ , and  $u(x, T)$

Learned functions	$f(u)$		$A(u)$		$u(x, T)$	
	Mean	Max	Mean	Max	Mean	Max
Learned functions in Sect. 4.1	0.01893	0.13549	0.01934	0.10797	0.00471	0.07192
Learned functions in Sect. 4.2	0.00366	0.03367	0.00279	0.01270	0.00112	0.02127
Learned functions in Sect. 4.3	0.00358	0.00817	0.00421	0.00915	0.00140	0.05186
Learned functions in Sect. 4.4	0.01251	0.04900	0.00895	0.04055	0.00627	0.03917
Learned functions in Sect. 4.5	0.00321	0.03089	0.00213	0.01755	0.00084	0.02372
Learned functions with initial states Table 2	0.00366	0.01190	0.00402	0.01720	0.00107	0.01700
Learned functions with changed $L$ and $T$	0.00356	0.02395	0.00326	0.01842	0.00135	0.01665
Learned functions with adding 3% noise to observations	0.00478	0.02031	0.00831	0.02720	0.00137	0.02109
Learned functions with adding 9% noise to observations	0.01046	0.02563	0.03001	0.06090	0.00423	0.08165
Learned functions with seven hidden layers	0.00850	0.02141	0.01133	0.03025	0.00349	0.03025
Learned functions with $w(u) = u^2$	0.00308	0.03478	0.00202	0.02725	0.00078	0.02451



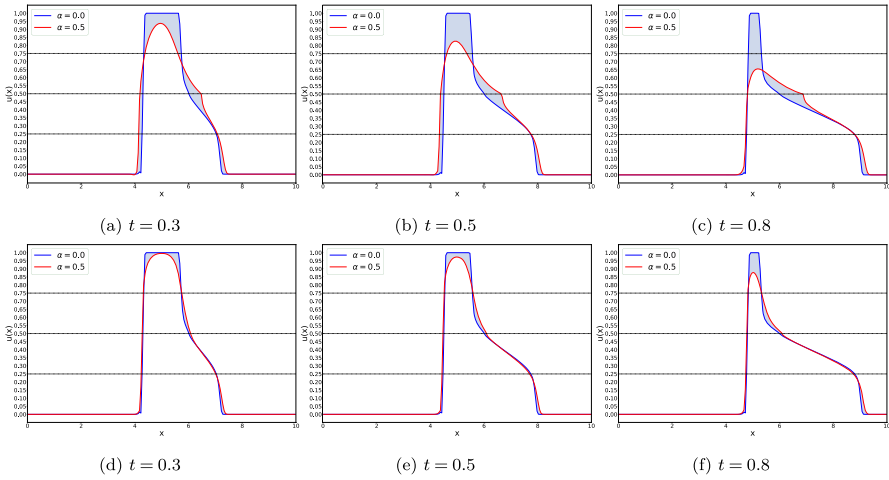


**Fig. 13**  $u(x, t)$  at  $t = 2.0$  based on the initial states  $u_1^0$  and  $u_5^0$  in Table 1 with different values of  $\alpha$ . The left and right subplots are based on  $u_1^0$  and  $u_5^0$ , respectively

in Sect. 4.4. Additionally, the learning effect of  $a_2$  ( $A_2$ ) differs significantly from that of  $a_1$  ( $A_1$ ), even under the same experimental settings, as seen in the experiments in Sects. 4.2 and 4.3. These experimental phenomena have motivated us to investigate further the relative impact of  $f(u)$  and  $a(u)$  on the solution of (1.1).

To demonstrate the impact of the diffusion function on the solution, we vary the value of  $\alpha$  in  $A_1(u) = \alpha \int_0^u a_1(v)dv$ . Setting  $\alpha = 0.0$  corresponds to omitting the diffusion function in (1.1). The solution of (1.1) at  $t = 2.0$  based on the initial states  $u_1^0$  and  $u_5^0$  with different  $\alpha$  is displayed in Fig. 13. The red line shows the solution when there is no diffusion function  $a(u)$ . Increasing  $\alpha$  gradually enhances the influence of  $A_1(u)$  on the solutions. However, despite increasing  $\alpha$  to 1.0, the solutions of the equation remain largely unaffected. Taking Fig. 13a as an example, the diffusion function weakly affects the solutions only in the range of  $x \in (7.0, 10.0)$ . This may indicate why the learning effect of  $f(u)$  is superior to that of  $a(u)$  or  $A(u)$ . Even if the learning effect of  $a(u)$  or  $A(u)$  is poor, it does not have a substantial influence on the solutions.

To investigate why the learning effect of  $a_2(u)$  or  $A_2(u)$  is inferior to that of  $a_1(u)$  or  $A_1(u)$ , we plotted the data for several observation time points ( $t = 0.3, 0.5, 0.8$ ) in Fig. 14. In each graph, the red and blue lines denote the inclusion and exclusion of the diffusion function ( $a_1(u)$  or  $a_2(u)$ ) in (1.1), respectively, and the shaded area illustrates the difference between them. The observation data in the top row is utilized to learn  $a_1(u)$ , whereas the observation data in the bottom row is employed to learn  $a_2(u)$ . The shaded area reflects that  $a_1(u)$  has a greater effect (see upper row) than  $a_2(u)$  (see lower row) on the solutions.



**Fig. 14** Visualization of observed data at different time points. The upper and lower rows represent the observations generated by  $a_1(u)$  and  $a_2(u)$ , respectively

### 5 Experiment 2

In this section, we explore (1.1) and identification of the flux function  $f(u)$  given by

$$f(u) = \frac{u^2}{u^2 + 0.5(1 - u)^4} \left( 1 - 5(1 - u)^4 \right) \tag{5.1}$$

and the diffusion function  $a(u)$

$$a(u) = \begin{cases} 0, & \text{if } u \in [0, 0.5] \\ (u - 0.5)u, & \text{otherwise} \end{cases} \tag{5.2}$$

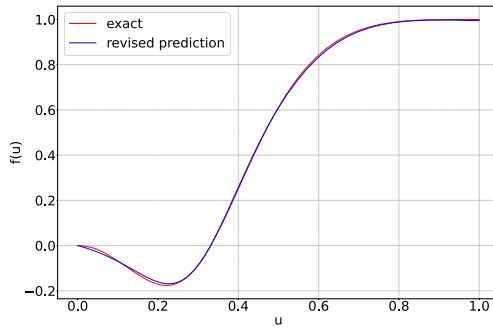
with  $\alpha = 1$ .

We investigate a one-dimensional spatial domain with  $x \in [0, 10]$  and a time interval  $[0, T]$  with  $T = 2$ . Observations are collected in the form of (3.15) and (3.25), with a numerical grid of  $N_x = 200$  grid cells,  $N_{\text{obs}} = 9$ , and  $\Delta t^{\text{obs}} = 0.1$ . Our objective is to identify the unknown flux function  $f(u)$  and diffusion function  $a(u)$  for  $u \in [0, 1]$ . We use a set of initial states  $\{u_k^0\}_{k=1}^{K_{\text{init}}}$  such that  $0 \leq u_k^0(x) \leq 1$  to learn  $f(u)$  and  $a(u)$ . Table 4 shows the initial states used in the experiments that follow.

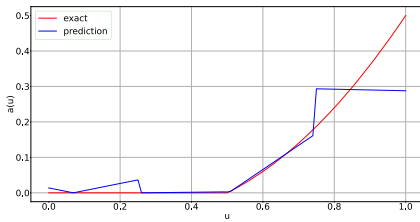
As in Sect. 4, we employ one S-Net to learn  $f(u)$ , and four separate S-Nets corresponding to  $u \in [0, 0.25)$ ,  $u \in [0.25, 0.5)$ ,  $u \in [0.5, 0.75)$ , and  $u \in [0.75, 1.0)$  to learn  $a(u)$ . The initial values of the neural networks are randomly generated. Additionally, for the piecewise affine functions method that represents unknown functions, we set  $n = 10$  in (3.11).

**Table 4** The initial states

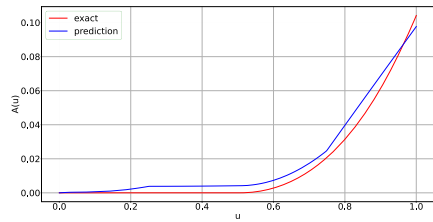
$u_1^0 = \begin{cases} 0.8 & \text{if } x \in [4, 6] \\ 0 & \text{otherwise} \end{cases}$	$u_2^0 = \begin{cases} 1.0 & \text{if } x \in [4, 6] \\ 0 & \text{otherwise} \end{cases}$	$u_3^0 = \begin{cases} 0.9 & \text{if } x \in [3, 5] \\ 0 & \text{otherwise} \end{cases}$
$u_4^0 = \begin{cases} 0.95 & \text{if } x \in [2.5, 4.5] \\ 0.3 & \text{if } x \in [0, 2.5) \\ 0.3 & \text{if } x \in (4.5, 10] \end{cases}$	$u_5^0 = \begin{cases} 0.85 & \text{if } x \in [2.5, 4.5] \\ 0.3 & \text{if } x \in [0, 2.5) \\ 0.3 & \text{if } x \in (4.5, 10] \end{cases}$	$u_6^0 = \begin{cases} 1.0 & \text{if } x \in [2.5, 4.5] \\ 0.05 & \text{if } x \in [0, 2.5) \\ 0.05 & \text{if } x \in (4.5, 10] \end{cases}$
$u_7^0 = \begin{cases} 1.0 & \text{if } x \in [2.5, 4.5] \\ 0.1 & \text{if } x \in [0, 2.5) \\ 0.1 & \text{if } x \in (4.5, 10] \end{cases}$		



(a)  $f_{\theta_f^*}(u)$



(b)  $a_{\theta_a^*}(u)$

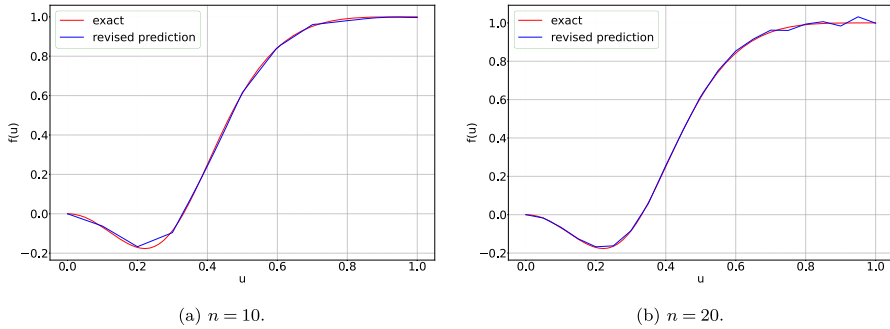


(c)  $A_{\theta_a^*}(u)$

**Fig. 15** The graphical result of identification of flux function  $f_{\theta_f^*}(u)$ , diffusion function  $a_{\theta_a^*}(u)$  and  $A_{\theta_a^*}(u)$ . The red line represents the true function, while the blue line illustrates the learned function (color figure online)

### 5.1 Identification of $f(u)$ and $a(u)$ using standard observations (Data I) with four S-Nets for representing $a(u)$

The S-Net used to represent  $f(u)$  consists of six hidden layers, while the S-Nets used to represent  $a(u)$  have only one hidden layer. The learned functions of  $f_{\theta_f^*}(u)$ ,  $a_{\theta_a^*}(u)$  and  $A_{\theta_a^*}(u)$  are shown in Fig. 15a–c, respectively. The learned function  $f_{\theta_f^*}(u)$  accurately approximates the true function  $f(u)$ , but for the diffusion function  $a(u)$ , noticeable discrepancies are observed for  $u \in [0.8, 1.0)$ . Consequently, some error exists between  $A_{\theta_a^*}(u)$  and  $A(u)$ .



**Fig. 16** Learned  $\hat{f}(u)$  based on the piecewise affine functions with  $n = 10$  and  $n = 20$ , respectively

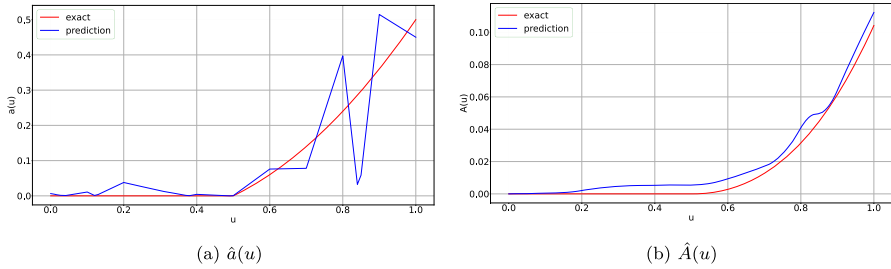
## 5.2 Identification of $f(u)$ and $a(u)$ using standard observations (Data I) with the piecewise affine functions for representing $a(u)$

In this section, the experimental settings mirror those in Sect. 5.1, except for the representation of the unknown function. The learned functions of  $\hat{f}(u)$ ,  $\hat{a}(u)$ , and  $\hat{A}(u)$  are presented in Fig. 16a, 17a, and 17b, respectively. Aside from the interval  $u \in (0.2, 0.35)$ ,  $\hat{f}(u)$  is generally close to  $f(u)$ . However, due to the numerous piecewise affine functions used in learning  $\hat{f}(u)$ , it contains multiple non-smooth points. This non-smoothness results in considerable fluctuations in the predicted solutions, as depicted in Fig. 18. The learning performance of  $\hat{a}(u)$  is suboptimal, especially for  $u \in [0.75, 1.0]$ . However,  $\hat{A}(u)$  can capture the overall trend of  $A(u)$ . Figure 18 demonstrates that regardless of the chosen method for representing  $f(u)$ , maintaining its smoothness is essential.

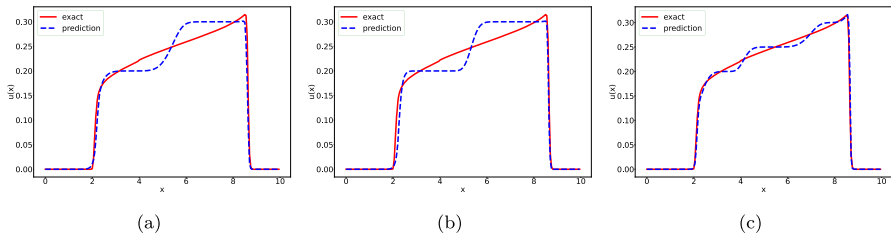
Is it so that the lack of accuracy in the predicted solutions comes essentially from a lack of precision in  $f(u)$ , and not  $a(u)$ ? To prove it, we use the predicted  $\hat{f}(u)$  and true  $a(u)$  to solve the solutions. Figure 18b illustrates that the accuracy of solutions comes from  $f(u)$ . Increasing the number of piecewise affine functions improves the model's performance to a certain extent. Figure 16b demonstrates the learning effect of  $\hat{f}(u)$  based on  $n = 20$ , which is double the value used in Fig. 16a. However, due to computing resource limitations,  $n$  cannot be increased indefinitely. The non-smooth nature of piecewise affine functions leads to fluctuations in the predicted solution, as illustrated in Fig. 18c.

## 5.3 Identification of $f(u)$ and $a(u)$ using particle trajectories (Data II) with four S-Nets for representing $a(u)$

In this experiment, we use a six-layer S-Net to represent  $f(u)$ , while the S-Nets used for  $a(u)$  have only one hidden layer. We select ten particles and use their trajectories as observation data. The initial positions of these particles at  $t = 0$  are denoted by (5.3)



**Fig. 17** The graphical result of identification of diffusion function  $\hat{a}(u)$  and  $\hat{A}(u)$ . The red line represents the true function, while the blue line illustrates the learned function (color figure online)



**Fig. 18**  $u(x, t)$  at  $t = 2.0$  based on the initial states of  $u_1^0$  in Table 4. The left subplot displays the solution generated by  $\hat{f}(u)$  and  $\hat{a}(u)$ , both with  $n = 10$ . The middle subplot presents the solution derived from  $\hat{f}(u)$  with  $n = 10$  and the true  $a(u)$ . The right subplot features the solution obtained from  $\hat{f}(u)$  with  $n = 20$  and  $\hat{a}(u)$  with  $n = 10$

$$X^0 = \left\{ 0.025, 1.025, 2.025, 3.025, 4.025, 5.025, 6.025, 7.025, 8.025, 9.025 \right\}. \tag{5.3}$$

The learned functions  $f_{\theta_f^*}(u)$ ,  $a_{1\theta_a^*}(u)$ , and  $A_{1\theta_{a^*}}(u)$  are displayed in Fig. 19a–c, respectively. There is a slight difference between  $f_{\theta_f^*}(u)$  to  $f(u)$ . However, given the small amount of data, such an error seems acceptable.

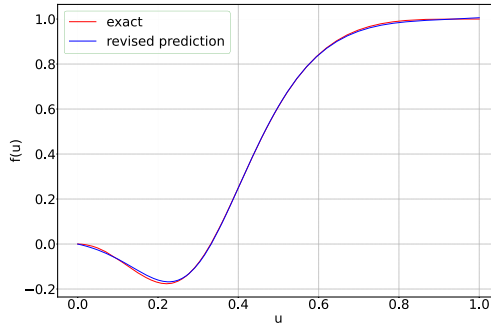
### 5.4 Discussion

#### 5.4.1 Analyzing mean and maximum errors in $f(u)$ , $A(u)$ , and $u(x, T)$

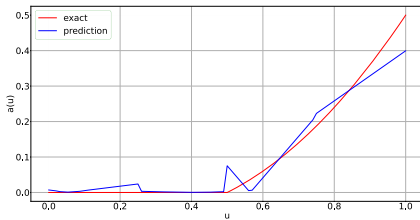
We analyze Mean and Maximum Errors in  $f(u)$ ,  $A(u)$ , and  $u(x, T)$  learned in different situations. The results are shown in Table 5. The main observation is that the findings from Experiment 1 largely carry over to the case studied in Experiment 2 with the rational flux function (5.1) and diffusion function (5.2) as the unknown functions.

## 6 Conclusion

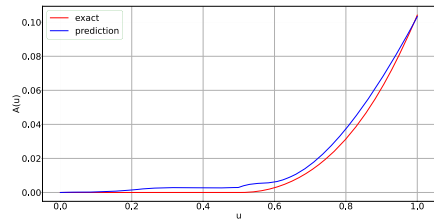
In this paper we have explored how to learn both the flux function  $f(u)$  and the diffusive flux  $A(u)$  for a degenerate convection-diffusion model (1.1). Since  $a(u) = A'(u)$



(a)  $f_{\theta_f^*}(u)$



(b)  $a_{\theta_a^*}(u)$



(c)  $A_{\theta_a^*}(u)$

**Fig. 19** The graphical result of identification of flux function  $f_{\theta_f^*}(u)$ , diffusion function  $a_{\theta_a^*}(u)$  and  $A_{\theta_a^*}(u)$ . The red line represents the true function, while the blue line illustrates the learned function (color figure online)

**Table 5** Analyzing mean and maximum errors in  $f(u)$ ,  $A(u)$ , and  $u(x, T)$

Model	$f(u)$		$A(u)$		$u(x, T)$	
	Mean	Max	Mean	Max	Mean	Max
Learned functions in Sect. 5.1	0.00481	0.01135	0.00399	0.00935	0.00392	0.09399
Learned functions in Sect. 5.2	0.00862	0.03343	0.00446	0.01049	0.01065	0.15263
Learned functions in Sect. 5.3	0.00438	0.01047	0.00401	0.00957	0.00382	0.08179

may vanish on intervals, solutions evolve into discontinuous solutions which must be interpreted in the sense of entropy solutions (1.3). Building on experience with how to learn  $f(u)$  for the case when  $a(u) = 0$  is not present [53], we here extend the methodology to also account for the unknown  $a(u)$ . The essential ingredients for successful identification of  $f(u)$  and  $a(u)$  is a proper combination of acquiring observation data based on several randomly selected initial data and a certain regularity imposed on the candidate functions for  $f$  and  $a$  by using the symbolic neural network structure. We conducted experiments to evaluate the effectiveness of the proposed approach, which showed promising results. To provide a comprehensive assessment, we explored both standard observational datasets and particle trajectory-based observational datasets. Main findings are:

- The method shows a strong ability to learn the correct flux function  $f(u)$  and  $A(u)$ , however, there is more uncertainty related to the identification of the correct  $a(u)$  as the impact from this term is typically much weaker.
- The use of trajectory data  $z_j(t)$  generated by solving the ODE (3.18) which depends on the solution  $u(x, t)$  of (1.1), gives the interesting finding that the quality of the identification of  $f$  and  $A$  is as good as the one achieved by using observations directly of  $u(x, t)$ . However, in the first case we only need a small fraction of the number of data points used in the latter case. This suggests that there is room for improved learning by using clever choices of observation data.
- The symbolic neural network based functions impose regularity when we search for candidate functions  $f(u)$  and  $A(u)$  which appears to be a robust and convenient approach when we work with hidden PDEs where solutions are discontinuous.

In conclusion, the proposed method offers a reliable and robust method to recover hidden degenerate convection-diffusion equations of the form (1.1) from observation data that have been collected, for instance, from an experimental setup.

**Funding** Open access funding provided by University of Stavanger & Stavanger University Hospital.

## Declarations

**Conflict of interest** The authors declare no Conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Afif, M., Amaziane, B.: Convergence of finite volume schemes for a degenerate convection-diffusion equation arising in flow in porous media. *Comput Methods Appl Mech Engrg* **191**(46), 5265–5285 (2002)
2. Atkinson, K.: *An Introduction to Numerical Analysis*. Wiley, New York (1991)
3. Bezgin, D.A., Schmidt, S.J., Adams, N.A.: A data-driven physics-informed finite-volume scheme for nonclassical undercompressive shocks. *J. Comput. Phys.* **437**, 110324 (2021)
4. Billard, L., Diday, E.: From the statistics of data to the statistics of knowledge: symbolic data analysis. *J. Am. Stat. Assoc.* **98**(462), 470–487 (2003)
5. Bongard, J., Lipson, H.: Automated reverse engineering of nonlinear dynamical systems. *Proc. Natl. Acad. Sci.* **104**(24), 9943–9948 (2007)
6. Bouchut, F., Guarguaglini, F.R., Natalini, R.: Diffusive bgk approximations for nonlinear multidimensional parabolic equations. *Indiana Univ. Math. J.* **49**(2), 749–282 (2000)
7. Brandstetter, J., Worrall, D., Welling, M.: Message passing neural PDE solvers. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25–29, 2022*. OpenReview.net (2022)
8. Brunton, S.: Discovering governing equations from data by sparse identification of nonlinear dynamics. In: *APS March Meeting Abstracts*, volume 2017, pages X49–004 (2017)

9. Bustos, M.C., Concha, F., Bürger, R., Tory, E.M.: Sedimentation and Thickening - Phenomenological Foundation and Mathematical Theory. Kluwer Academic Publishers, Cambridge (1999)
10. Carillo, J.: Entropy solutions for nonlinear degenerate problems. *Arch. Ration. Mech. Anal.* **147**(4), 269–361 (1999)
11. Champion, K., Lusch, B., Kutz, J.N., Brunton, S.L.: Data-driven discovery of coordinates and governing equations. *Proc. Natl. Acad. Sci.* **116**(45), 22445–22451 (2019)
12. Chang, B., Chen, M., Haber, E., Chi, E.H.: Antisymmetricrnn: A dynamical system view on recurrent neural networks. In: International Conference on Learning Representations (2018)
13. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. *Adv. Neural Inf. Process. Systems*, **31**, (2018)
14. Chen, Z., Liu, Y., Sun, H.: Physics-informed learning of governing equations from scarce data. *Nat. Commun.* **12**(1), 6136 (2021)
15. Chen, Z., Zhang, J., Arjovsky, M., Bottou, L.: Symplectic recurrent neural networks. In: International Conference on Learning Representations (2019)
16. 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)
17. Colombo, R.M., Marson, A.: A hölder continuous ode related to traffic flow. *Proc. R. Soc. Edinb. Sect. A Math.* **133**(4), 759–772 (2003)
18. Cornforth, T., Lipson, H.: Symbolic regression of multiple-time-scale dynamical systems. In: Proceedings of the 14th annual conference on Genetic and evolutionary computation, pages 735–742 (2012)
19. Dafermos, C.M., Dafermos, C.M.: *Hyperbolic Conservation Laws in Continuum Physics*, vol. 3. Springer, Berlin (2005)
20. Dam, M., Brøns, M., Juul Rasmussen, J., Naulin, V., Hesthaven, J.S.: Sparse identification of a predator-prey system from simulation data of a convection model. *Phys. Plasmas* **24**(2), 022310 (2017)
21. DeBrouwer, E., Simm, J., Arany, A., Moreau, Y.: Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *Adv. Neural Inf. Process. Syst.* **32**, (2019)
22. Díaz-Adame, R., Jerez, S., Carrillo, H.: Fast and optimal weno schemes for degenerate parabolic conservation laws. *J. Sci. Comput.* **90**(1), 22 (2022)
23. Diehl, S.: Estimation of the batch-settling flux function for an ideal suspension from only two experiments. *Chem. Eng. Sci.* **62**, 4589–4601 (2007)
24. Diehl, S.: Numerical identification of constitutive functions in scalar nonlinear convection-diffusion equations with application to batch sedimentation. *Appl. Numer. Math.* **95**, 154–172 (2015)
25. Duong, D.L.: Inverse problems for hyperbolic conservation laws: a Bayesian approach. PhD thesis, University of Sussex (2021)
26. Evje, S., Karlsen, K.H.: Viscous splitting approximation of mixed hyperbolic-parabolic convection-diffusion equations. *Numer. Math.* **83**(1), 107–137 (1999)
27. Evje, S., Karlsen, K.H.: Monotone difference approximations of BV solutions to degenerate convection-diffusion equations. *SIAM J. Numer. Anal.* **37**(6), 1838–1860 (2000)
28. Evje, S., Karlsen, K.H.: An error estimate for viscous approximate solutions of degenerate parabolic equations. *J. Nonlinear Math. Phys.* **9**(3), 262–281 (2002)
29. Eymard, R., Gallouet, T., Herbin, R.: Convergence of a finite volume scheme for nonlinear degenerate parabolic equations. *Numer. Math.* **92**, 41–82 (2002)
30. Fuks, O., Tchelepi, H.A.: Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *J. Mach. Learn. Model. Comput.*, **1**(1) (2020)
31. Galiano, S.J., Zapata, M.U.: A new tvd flux-limiter method for solving nonlinear hyperbolic equations. *J. Comput. Appl. Math.* **234**(5), 1395–1403 (2010)
32. Gao, H., Sun, L., Wang, J.X.: Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *J. Comput. Phys.* **428**, 110079 (2021)
33. Gao, H., Zahr, M.J., Wang, J.X.: Physics-informed graph neural galerkin networks: A unified framework for solving pde-governed forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **390**, 114502 (2022)
34. Gaucel, S., Keijzer, M., Lutton, E., Tonda, A.: Learning dynamical systems using standard symbolic regression. In: Genetic Programming: 17th European Conference, EuroGP 2014, Granada, Spain, April 23–25, 2014, Revised Selected Papers 17, pp. 25–36. Springer (2014)



35. Herrera, C., Krach, F., Teichmann, J.: Neural jump ordinary differential equations: Consistent continuous-time prediction and filtering. In: International Conference on Learning Representations (2020)
36. Hesthaven, J.S.: Numerical methods for conservation laws: From analysis to algorithms. *SIAM. Comput. Sci. Eng.* (2017)
37. Holden, H., Karlsen, K.H., Lie, K.A., Risebro, N.H.: Splitting methods for partial differential equations with rough solutions. *Eur. Math. Soc.* (2010)
38. Holden, H., Priuli, F.S., Risebro, N.H.: On an inverse problem for scalar conservation laws. *Inverse Prob.* **30**, 035015 (2014)
39. Iakovlev, V., Heinonen, M., Lähdesmäki, H.: Learning continuous-time pdes from sparse data with graph neural networks. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. [OpenReview.net](https://openreview.net) (2021)
40. James, F., Sepúlveda, M.: Parameter identification for a model of chromatographic column. *Inverse Prob.* **10**(6), 1299 (1994)
41. James, F., Sepúlveda, M.: Convergence results for the flux identification in a scalar conservation law. *SIAM J. Control. Optim.* **37**(3), 869–891 (1999)
42. James, F., Sepúlveda, M.: Convergence results for the flux identification in a scalar conservation law. *SIAM J. Control. Optim.* **37**(3), 869–891 (1999)
43. Kaheman, K., Kutz, J.N., Brunton, S.L.: Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proc. R. Soc. A* **476**(2242), 20200279 (2020)
44. Kang, H., Tanuma, K.: Inverse problems for scalar conservation laws. *Inverse Prob.* **21**(3), 1047 (2005)
45. Karlsen, K.H., Risebro, N.H., Storrøsten, E.B.:  $L^1$  error estimates for difference approximations of degenerate convection-diffusion equations. *Math. Comp.* **83**(290), 2717–2762 (2014)
46. Karlsen, K.H., Risebro, N.H., Storrøsten, E.B.: On the convergence rate of finite difference methods for degenerate convection-diffusion equations in several space dimensions. *ESAIM Math. Modell. Numer. Anal.* **50**(2), 499–539 (2016)
47. Kim, S., Lu, P.Y., Mukherjee, S., Gilbert, M., Jing, L., Čeperić, V., Soljačić, M.: Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Trans. Neural Netw. Learn. Syst.* **32**(9), 4166–4177 (2020)
48. Koley, U., Risebro, N.H., Schwab, C., Weber, F.: A multilevel Monte Carlo finite difference method for random scalar degenerate convection diffusion equations. *J. Hyperbolic Differ. Equ.* **14**(3), 415–445 (2017)
49. Kröner, D.: Numerical schemes for conservation laws. Wiley-Teubner Series Advances in Numerical Mathematics (1997)
50. Kruzkov, S.N.: First order quasilinear equations with several independent variables. *Mat. Sb. (N.S.)* **81**(123), 228–255 (1970)
51. Kurganov, A., Tadmor, E.: New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations. *J. Comput. Phys.* **160**(1), 241–282 (2000)
52. LeVeque, R.J.: Finite volume methods for hyperbolic problems. Cambridge Texts in Applied Mathematics (2007)
53. Li, Q., Evje, S.: Learning the nonlinear flux function of a hidden scalar conservation law from data. *Network Heterogeneous Media*, **18**, (2023)
54. Li, Q., Evje, S., Geng, J.: Learning parameterized odes from data. *IEEE Access* **11**, 54897–54909 (2023)
55. Li, Q., Geng, J., Evje, S.: Identification of the flux function of nonlinear conservation laws with variable parameters. *Physica D* **451**, 133773 (2023)
56. Li, Q., Geng, J., Evje, S., Rong, C.: Solving nonlinear conservation laws of partial differential equations using graph neural networks. *Proc. Northern Lights Deep Learn. Workshop* **2023**, 4 (2023)
57. Li, Q., Evje, S.: Learning the nonlinear flux function of a hidden scalar conservation law from data. *Netw. Heterogen. Media* **18**(1), 48–79 (2023)
58. Lighthill, M.J., Whitham, G.B.: On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proc. R. Soc. Lond. A* **229**(1178), 317–345 (1955)
59. Liu, X.D., Oshery, S., Chanz, T.: Weighted essentially non-oscillatory schemes. *J. Comput. Phys.* **115**(1), 200–212 (1994)
60. Liu, Z., Madhavan, V., Tegmark, M.: Machine learning conservation laws from differential equations. *Phys. Rev. E* **106**(4), 045307 (2022)

61. Liu, Z., Tegmark, M.: Machine learning conservation laws from trajectories. *Phys. Rev. Lett.* **126**(18), 180604 (2021)
62. Long, Z., Lu, Y., Dong, B.: PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *J. Comput. Phys.* **399**, 108925 (2019)
63. Long, Z., Lu, Y., Ma, X., Dong, B.: Pde-net: Learning PDEs from data. In: International Conference on Machine Learning, pages 3208–3216. PMLR (2018)
64. Magiera, J., Ray, D., Hesthaven, J.S., Rohde, C.: Constraint-aware neural networks for riemann problems. *J. Comput. Phys.* **409**, 109345 (2020)
65. Martius, G., Lampert, C.H.: Extrapolation and learning equations. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Workshop Track Proceedings. OpenReview.net (2017)
66. Mishra, S., Schwab, C.: Sparse tensor multi-level monte carlo finite volume methods for hyperbolic conservation laws with random initial data. *Math. Comput.* **81**(280), 1979–2018 (2012)
67. Mototake, Y.I.: Interpretable conservation law estimation by deriving the symmetries of dynamics from trained deep neural networks. *Phys. Rev. E* **103**(3), 033303 (2021)
68. Mundhenk, T.N., Landajuela, M., Glatt, R., Santiago, C.P., Faissol, D.M., Petersen, B.K.: Symbolic regression via neural-guided genetic programming population seeding. arXiv preprint [arXiv:2111.00053](https://arxiv.org/abs/2111.00053) (2021)
69. Narasingam, A., Kwon, J.S.I.: Data-driven identification of interpretable reduced-order models using sparse regression. *Comput. Chem. Eng.* **119**, 101–111 (2018)
70. Ohlberger, M.: A posteriori error estimates for vertex centered finite volume approximations of convection-diffusion-reaction equations. *M2AN Math. Model Numer. Anal.* **35**(2), 355–387 (2002)
71. Petersen, B.K., Landajuela, M., Mundhenk, T.N., Santiago, C.P., Kim, S.K., Kim, J.T.: Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. arXiv preprint [arXiv:1912.04871](https://arxiv.org/abs/1912.04871) (2019)
72. Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., Battaglia, P.W.: Learning mesh-based simulation with graph networks. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3–7, 2021. OpenReview.net (2021)
73. 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)
74. Ray, D., Hesthaven, J.S.: An artificial neural network as a troubled-cell indicator. *J. Comput. Phys.* **367**, 166–191 (2018)
75. Richards, P.I.: Shock waves on the highway. *Oper. Res.* **4**(1), 42–51 (1956)
76. Rubanova, Y., Chen, R.T., Duvenaud, D.K.: Latent ordinary differential equations for irregularly-sampled time series. *Adv. Neural Inf. Process. Syst.* **32**, (2019)
77. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint [arXiv:1609.04747](https://arxiv.org/abs/1609.04747) (2016)
78. Sahoo, S., Lampert, C., Martius, G.: Learning equations for extrapolation and control. In: International Conference on Machine Learning, pp. 4442–4450. PMLR (2018)
79. Schaeffer, H.: Learning partial differential equations via data discovery and sparse optimization. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **473**(2197), 20160446 (2017)
80. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. *Science* **324**(5923), 81–85 (2009)
81. Shu, C.W.: Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. *Adv. Numerical Approximation of Nonlinear Hyperbolic Equ.*, pp. 325–432 (1998)
82. Skadsem, H.J., Kragset, S.: A numerical study of density-unstable reverse circulation displacement for primary cementing. *J. Energy Res. Technol.* **144**, 123008 (2022)
83. Thurey, N., Weißenow, K., Prantl, L., Hu, X.: Deep learning methods for reynolds-averaged navier-stokes simulations of airfoil flows. *AIAA J.* **58**(1), 25–36 (2020)
84. Vaddireddy, H., Rasheed, A., Staples, A.E., San, O.: Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data. *Phys. Fluids* **32**(1), 015113 (2020)
85. Volpert, A.I.: Generalized solutions of degenerate second-order quasilinear parabolic and elliptic equations. *Adv. Differ. Equ.* **5**(10–12), 1493–1518 (2000)
86. Volpert, A.I., Hudjaev, S.I.: The cauchy problem for second order quasilinear degenerate parabolic equations. *Mat. Sb. (N.S.)* **78**(120), 374–396 (1969)

87. Wandel, N., Weinmann, M., Klein, R.: Learning incompressible fluid dynamics from scratch - towards fast, differentiable fluid models that generalize. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021)
88. Wang, Y., Shen, Z., Long, Z., Dong, B.: Learning to discretize: solving 1d scalar conservation laws via deep reinforcement learning. arXiv preprint [arXiv:1905.11079](https://arxiv.org/abs/1905.11079) (2019)
89. Zhao, Q., Lindell, D.B., Wetzstein, G.: Learning to solve pde-constrained inverse problems with graph networks. In: Kamalika, C., Stefanie, J., Le, S., Csaba, S., Gang, N., Sivan, S. (eds.), International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, volume 162 of Proceedings of Machine Learning Research, pp. 26895–26910. PMLR (2022)
90. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw. (TOMS)* **23**(4), 550–560 (1997)

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