Journal of Computational Physics 443 (2021) 110514

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation ☆

Zhiqiang Cai [a,*], Jingshuang Chen [a], Min Liu [b]

[a] Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067, United States of America
[b] School of Mechanical Engineering, Purdue University, 585 Purdue Mall, West Lafayette, IN 47907-2088, United States of America

ARTICLE INFO

Article history:
Available online 16 June 2021

Keywords:
Least-squares method
ReLU neural network
Linear advection-reaction equation

ABSTRACT

This paper studies least-squares ReLU neural network method for solving the linear advection-reaction problem with discontinuous solution. The method is a discretization of an equivalent least-squares formulation in the set of neural network functions with the ReLU activation function. The method is capable of approximating the discontinuous interface of the underlying problem automatically through the *free* hyper-planes of the ReLU neural network and, hence, outperforms mesh-based numerical methods in terms of the number of degrees of freedom. Numerical results of some benchmark test problems show that the method can not only approximate the solution with the least number of parameters, but also avoid the common Gibbs phenomena along the discontinuous interface. Moreover, a three-layer ReLU neural network is necessary and sufficient in order to well approximate a discontinuous solution with an interface in $\mathbb{R}^2$ that is not a straight line.

1. Introduction

During the past several decades, numerical methods for linear advection-reaction equations have been intensively studied by many researchers and many numerical schemes have been developed. When inflow boundary data is discontinuous, so is the solution. It is well-known that traditional mesh-based numerical methods often exhibit oscillations near a discontinuity (called the Gibbs phenomena). Such spurious oscillations are unacceptable for many applications (see, e.g, [16]). To eliminate or reduce the Gibbs phenomena, finite difference and finite volume methods often use numerical techniques such as limiters, filters, ENO/WENO, etc. [13,15,16,20]; and finite element methods usually employ discontinuous finite elements [4,9,12] and/or adaptive mesh refinement (AMR) to generate locally refined elements along discontinuous interfaces (see, e.g., [5,17,18]).

Recently, there has been increasing interests in using deep neural networks (DNNs) to solve partial differential equations (see, e.g., [7,25,28]). DNNs produce a large class of functions through compositions of linear transformations and activation functions. One of the striking features of DNNs is that this class of functions is not subject to a hand-crafted geometric mesh or point cloud as are the traditional, well-studied finite difference, finite volume, and finite element methods. The physical partition of the domain Ω, formed by free hyper-planes, can automatically adapt to the target function. This is much better than the AMR generated mesh because AMR is based on a geometric mesh and subject to mesh conformity; moreover, it is

☆ This work was supported in part by the National Science Foundation under grant DMS-2110571.
* Corresponding author.
E-mail addresses: caiz@purdue.edu (Z. Cai), chen2042@purdue.edu (J. Chen), liu66@purdue.edu (M. Liu).

https://doi.org/10.1016/j.jcp.2021.110514
0021-9991/

not easy to remove unnecessary elements or points. This paper will make use of this powerful approximation property of DNNs for solving linear advection-reaction problem with discontinuous solution.

DNN functions are nonlinear functions of the parameters. Hence, the advection-reaction equation will be discretized through least-squares principles. In the context of finite element approximations, several least-squares methods have been studied (see, e.g., [1–3,8,10,11,22]). Basically, there are two least-squares formulations which are equivalent to the original differential equation. One is a direct application of least-squares principle (see, e.g., [1,10]) with a weighted L^2 norm for the inflow boundary condition, where the weight is the magnitude of the normal component of the advection velocity field. The other is to apply the least-squares principle to an equivalent system of the underlying problem by introducing an additional flux variable (see [11,22]). Some numerical techniques such as feedback least-squares finite element method [2], adaptive local mesh refinement with proper finite elements [22], etc. were introduced in order to reduce the Gibbs phenomena for problems with discontinuous solutions.

The purpose of this paper is to study the least-squares neural network (LSNN) method for solving the linear advection-reaction problem with discontinuous solution. The LSNN method is based on the least-squares formulation studied in ([1, 10]), i.e., a direct application of the lease-squares principle to the underlying problem, and on the ReLU neural network as the class of approximating functions. The class of neural network functions enables the LSNN method to automatically approximate the discontinuous solution without using *a priori* knowledge of the location of the discontinuities. Compared to various AMR methods that locate the discontinuous interface through local mesh refinement, the LSNN method is much more effective in terms of the number of the degrees of freedom (see, e.g., Fig. 1(c) and 2(c)).

Theoretically, it is proved in [10] that the homogeneous least-squares functional is equivalent to a natural norm in the solution space $V_{\boldsymbol{\beta}}$ consisting of all square-integrable functions whose directional derivative along $\boldsymbol{\beta}$ is also square-integrable (see section 2). This equivalence enables us to prove Ceá's lemma for the LSNN approximation, i.e., the error of the LSNN approximation is bounded by the approximation error of the set of ReLU neural network functions. This result is extended to the LSNN method with numerical integration as well. Even though approximation theory of the ReLU neural network has been intensively studied by many researchers (see, e.g., [24] for work before 2000 and [26,27]), we are not able to find a result which is applicable to the discontinuous solution of the advection-reaction problem.

To explore how well the ReLU neural network approximates the discontinuous solution, we consider two-dimensional transport problem, i.e., (2.4) with $\hat{\gamma} = 0$. When the boundary data g is discontinuous at point $\mathbf{x}_0 \in \Gamma_-$, the solution of the transport problem is discontinuous across an interface: the streamline of the advection velocity field starting at $\mathbf{x}_0$. The solution of this problem can be decomposed as the sum of a piece-wise constant function and a continuous piece-wise smooth function (see, e.g., (3.7)). We show that the piece-wise constant function can be approximated well without the Gibbs phenomena by either a two- or a three-layer ReLU neural network with the minimal number of neurons depending on the shape of the interface (see Lemmas 3.1 and 5.2). Together with the universal approximation property, this implies that a two- or three-layer ReLU neural network is sufficient to well approximate the solution of the linear transport problem without oscillation. These theoretical results are confirmed by numerical results.

The procedure for determining the values of the parameters of the network is now a problem in nonlinear optimization even though the underlying PDE is linear. This high dimensional, nonlinear optimization problem usually has many solutions. In order to obtain the desired one, we need to start from a close enough first approximation, and a common way to do so is by the method of continuation. In this paper, we propose the method of model continuation through approximating the advection velocity field by a family of piece-wise constant vector fields. Numerical results for a test problem with variable velocity field show that this method is able to reduce the total number of the parameters significantly.

The paper is organized as follows. Section 2 introduces the advection-reaction problem, its least-squares formulation, and preliminaries. The ReLU neural network and the least-squares neural network are described and analyzed in section 3. Initialization for the two-layer neural network and the method of model continuation for initialization are presented in sections 4 and 6, respectively. Finally, numerical results for various benchmark test problems are given in section 5.

Standard notations and definitions are used for the Sobolev space $H^s(\Omega)^d$ and $H^s(\Gamma_-)^d$ when $s \geq 0$. The associated norms with these two spaces are denoted by $\|\cdot\|_{s,\Omega}$ and $\|\cdot\|_{s,\Gamma_-}$, and their respective inner products are denoted as $(\cdot,\cdot)_{s,\Omega}$ and $(\cdot,\cdot)_{s,\Gamma_-}$. For $s = 0$ case, $H^s(\Omega)^d$ is the same as $L^2(\Omega)^d$, then the norm and inner product are simply denoted as $\|\cdot\|$ and $(\cdot,\cdot)$, respectively. The subscripts Ω in the designation of norms will be suppressed when there is no ambiguity.

2. Problem formulation

Let Ω be a bounded domain in $\mathbb{R}^d$ with Lipschitz boundary, and denote the advective velocity field by $\boldsymbol{\beta}(\mathbf{x}) = (\beta_1, \cdots, \beta_d)^T \in C^1(\bar{\Omega})^d$. Define the inflow and outflow parts of the boundary $\Gamma = \partial\Omega$ by

$$\Gamma_- = \{\mathbf{x} \in \Gamma : \boldsymbol{\beta}(\mathbf{x}) \cdot \boldsymbol{n}(\mathbf{x}) < 0\} \quad \text{and} \quad \Gamma_+ = \{\mathbf{x} \in \Gamma : \boldsymbol{\beta}(\mathbf{x}) \cdot \boldsymbol{n}(\mathbf{x}) > 0\}, \tag{2.1}$$

respectively, where $\boldsymbol{n}(\mathbf{x})$ is the unit outward normal vector to Γ at $\mathbf{x} \in \Gamma$.

As a model hyperbolic boundary value problem, we consider the linear advection-reaction equation

$$\begin{cases} \nabla \cdot (\boldsymbol{\beta} u) + \gamma u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_-, \end{cases} \tag{2.2}$$

where $\gamma \in C(\bar{\Omega})$, $f \in L^2(\Omega)$, and $g \in L^2(\Gamma_-)$ are given scalar-valued functions. We assume that there exist a positive constant γ_0 such that

$$\gamma(\mathbf{x}) + \frac{1}{2}\nabla \cdot \boldsymbol{\beta}(\mathbf{x}) \geq \gamma_0 > 0 \quad \text{for all } \mathbf{x} \in \Omega. \tag{2.3}$$

For simplicity of presentation, we also assume that g is bounded so that streamline functions from Γ_- to Γ_+ is not needed (see [10]).

Denote by $v_{\boldsymbol{\beta}} = \boldsymbol{\beta} \cdot \nabla v$ the directional derivative along the advective velocity field $\boldsymbol{\beta}$, then (2.2) may be rewritten as follows

$$\begin{cases} u_{\boldsymbol{\beta}} + \hat{\gamma}\, u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_-, \end{cases} \tag{2.4}$$

where $\hat{\gamma} = \gamma + \nabla \cdot \boldsymbol{\beta}$. The solution space of (2.2) is given by

$$V_{\boldsymbol{\beta}} = \{v \in L^2(\Omega) : v_{\boldsymbol{\beta}} \in L^2(\Omega)\},$$

which is equipped with the norm as

$$|||v|||_{\boldsymbol{\beta}} = \left(\|v\|_{0,\Omega}^2 + \|v_{\boldsymbol{\beta}}\|_{0,\Omega}^2\right)^{1/2}.$$

Denote the weighted $L^2(\Gamma_-)$ norm over the inflow boundary by

$$\|v\|_{-\boldsymbol{\beta}} = \langle v, v\rangle_{-\boldsymbol{\beta}}^{1/2} = \left(\int_{\Gamma_-} |\boldsymbol{\beta} \cdot \boldsymbol{n}|\, v^2\, ds\right)^{1/2}.$$

The following trace and Poincaré inequalities are proved in [10] (see also [2]) that there exist positive constants C_t and C_p such that

$$\|v\|_{-\boldsymbol{\beta}} \leq C_t\, |||v|||_{\boldsymbol{\beta}}, \quad \forall\, v \in V_{\boldsymbol{\beta}} \tag{2.5}$$

and

$$\|v\|_{0,\Omega} \leq C_p \left(\|v\|_{-\boldsymbol{\beta}} + D\, \|v_{\boldsymbol{\beta}}\|_{0,\Omega}\right), \quad \forall\, v \in V_{\boldsymbol{\beta}}, \tag{2.6}$$

respectively, where $D = \mathrm{diam}(\Omega)$ is the diameter of the domain Ω.

Remark 2.1. Let $\mathcal{C}$ be the streamline of the advection velocity field $\boldsymbol{\beta}$ starting at $\mathbf{x}_0 \in \Gamma_-$ in two dimensions. Assume that the inflow boundary condition g is discontinuous at $\mathbf{x}_0$. Then it is easy to see that the solution of (2.2) is also discontinuous across $\mathcal{C}$ because the restriction of the solution on $\mathcal{C}$ satisfies the same differential equation but different initial condition. Moreover, if $\hat{\gamma} = 0$, then the jump of the solution along $\mathcal{C}$ is a constant $|g(\mathbf{x}_0^+) - g(\mathbf{x}_0^-)|$, where $g(\mathbf{x}_0^+)$ and $g(\mathbf{x}_0^-)$ are the values of g at $\mathbf{x}_0$ from different sides. The streamline $\mathcal{C}$ is referred to be the discontinuous interface.

In the remainder of this section, we describe the least-squares (LS) formulation following [2,10]. To this end, define the LS functional

$$\mathcal{L}(v; \mathbf{f}) = \|v_{\boldsymbol{\beta}} + \hat{\gamma}\, v - f\|_{0,\Omega}^2 + \|v - g\|_{-\boldsymbol{\beta}}^2 \tag{2.7}$$

for all $v \in V_{\boldsymbol{\beta}}$, where $\mathbf{f} = (f, g)$. Now, the corresponding least-squares formulation is to seek $u \in V_{\boldsymbol{\beta}}$ such that

$$\mathcal{L}(u; \mathbf{f}) = \min_{v \in V_{\boldsymbol{\beta}}} \mathcal{L}(v; \mathbf{f}). \tag{2.8}$$

It follows from the trace, triangle, and Poincaré inequalities and assumptions on $\boldsymbol{\beta}$ and γ that the homogeneous LS functional $\mathcal{L}(v; \mathbf{0})$ is equivalent to the norm $|||v|||_{\boldsymbol{\beta}}^2$, i.e., there exist positive constants α and M such that

$$\alpha\, |||v|||_{\boldsymbol{\beta}}^2 \leq \mathcal{L}(v; \mathbf{0}) \leq M\, |||v|||_{\boldsymbol{\beta}}^2. \tag{2.9}$$

Furthermore, problem (2.8) has a unique solution $u \in V_{\boldsymbol{\beta}}$ satisfying the following *a priori* estimate

$$|||u|||_{\boldsymbol{\beta}} \leq C \left(\|f\|_{0,\Omega} + \|g\|_{-\boldsymbol{\beta}}\right). \tag{2.10}$$

Denote the bilinear and linear forms by

$$a(u,v) = (u_\beta + \hat{\gamma}\, u, v_\beta + \hat{\gamma}\, v) + \langle u, v\rangle_{-\beta} \quad \text{and} \quad f(v) = (f, v_\beta + \hat{\gamma}\, v) + \langle g, v\rangle_{-\beta},$$

respectively. Then the minimization problem in (2.8) is to find $u \in V_\beta$ such that

$$a(u,v) = f(v), \quad \forall\, v \in V_\beta. \tag{2.11}$$

3. Least-squares neural network method

This section describes deep neural networks and the corresponding least-squares method for linear transport equations. We consider a deep neural network (DNN) with a scalar-valued output as

$$\mathcal{N} : \mathbf{x} \in \mathbb{R}^d \longrightarrow \mathcal{N}(\mathbf{x}) \in \mathbb{R}.$$

The DNN function $\mathcal{N}(\mathbf{x})$ is typically represented as compositions of many layers of functions:

$$\mathcal{N}(\mathbf{x}) = N^{(L)} \circ \cdots N^{(2)} \circ N^{(1)}(\mathbf{x}), \tag{3.1}$$

where the symbol $\circ$ denotes the composition of functions, and L is the depth of the network. In this case, $N^{(l)}$ is called the l^{th} layer of the network. All layers except the last one $N^{(L)}$ are called hidden layers. A layer $N^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ is defined as a composition of a linear transformation $T^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ as follows:

$$N^{(l)}(\mathbf{x}^{(l-1)}) = \sigma\left(T^{(l)}(\mathbf{x}^{(l-1)})\right) = \sigma(\boldsymbol{\omega}^{(l)}\mathbf{x}^{(l-1)} - \mathbf{b}^{(l)}) \quad \text{for } \mathbf{x}^{(l-1)} \in \mathbb{R}^{n_{l-1}}, \tag{3.2}$$

where $\boldsymbol{\omega}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$, $\mathbf{x}^{(0)} = \mathbf{x}$, and application of σ to a vector is defined component-wise. There is typically no activation function in the output layer. Components of $\boldsymbol{\omega}^{(l)}$ and $\mathbf{b}^{(l)}$ are called weights and bias, respectively, and are parameters to be determined (trained). Each component of the vector-valued function $N^{(l)}$ is interpreted as a neuron and the dimension n_l defines the width or the number of neurons of the l^{th} layer in a network. This paper will use the popular rectified linear unit (ReLU) activation function defined by

$$\sigma(t) = \max\{0, t\} = \begin{cases} 0, & \text{if } t \le 0, \\ t, & \text{if } t > 0. \end{cases} \tag{3.3}$$

For given integers $\{n_l\}_{l=1}^{L}$, denote the set of DNN functions by

$$\mathcal{M}(\boldsymbol{\theta}, L) = \left\{\mathcal{N}(\mathbf{x}) = N^{(L)} \circ \cdots \circ N^{(1)}(\mathbf{x}) : \boldsymbol{\omega}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}},\ \mathbf{b}^{(l)} \in \mathbb{R}^{n_l} \text{ for } l = 1, ..., L\right\},$$

where $N^{(l)}(\mathbf{x}^{(l-1)})$ is defined in (3.2) and $\boldsymbol{\theta}$ denotes all parameters: $\boldsymbol{\omega}^{(l)}$ and $\mathbf{b}^{(l)}$ for $l = 1, ..., L$. It is easy to see that $\mathcal{M}(\boldsymbol{\theta}, L)$ is a subset of V_β, but not a linear subspace. The least-squares approximation is to find $u_N(\mathbf{x}; \boldsymbol{\theta}^*) \in \mathcal{M}(\boldsymbol{\theta}, L)$ such that

$$\mathcal{L}\big(u_N(\mathbf{x}; \boldsymbol{\theta}^*);\, \mathbf{f}\big) = \min_{v \in \mathcal{M}(\boldsymbol{\theta}, L)} \mathcal{L}\big(v(\mathbf{x}; \boldsymbol{\theta});\, \mathbf{f}\big) = \min_{\boldsymbol{\theta} \in \mathbb{R}^N} \mathcal{L}\big(v(\mathbf{x}; \boldsymbol{\theta});\, \mathbf{f}\big), \tag{3.4}$$

where N is the total number of parameters in $\mathcal{M}(\boldsymbol{\theta}, L)$ given by

$$N = M_d(L) = \sum_{l=1}^{L} n_l \times (n_{l-1} + 1).$$

Lemma 3.1. *Let u and u_N be the solutions of problems (2.7) and (3.4), respectively. Then we have*

$$|||u - u_N|||_\beta \le \left(\frac{M}{\alpha}\right)^{1/2} \inf_{v \in \mathcal{M}(\boldsymbol{\theta}, L)} |||u - v|||_\beta, \tag{3.5}$$

where α and M are constants in (2.9).

Proof. For any $v \in \mathcal{M}(\boldsymbol{\theta}, L) \subset V_\beta$, it follows from the coercivity and continuity of the homogeneous functional $\mathcal{L}(v;\, \mathbf{0})$ in (2.9), problem (2.2), and (3.4) that

$$\begin{aligned} \alpha\, |||u - u_N|||_\beta^2 &\le \mathcal{L}\big(u - u_N;\, \mathbf{0}\big) = \mathcal{L}\big(u_N(\mathbf{x}; \boldsymbol{\theta}^*);\, \mathbf{f}\big) \\ &\le \mathcal{L}\big(v(\mathbf{x}; \boldsymbol{\theta});\, \mathbf{f}\big) = \mathcal{L}\big(u - v;\, \mathbf{0}\big) \le M\, |||u - v|||_\beta^2, \end{aligned}$$

which implies (3.5). This completes the proof of the lemma. □

For a given vector $\boldsymbol{\xi} \in \mathbb{R}^d$ and $c \in \mathbb{R}$, assume that the hyper-plane $\mathcal{P} : \boldsymbol{\xi} \cdot \mathbf{x} = c$ divides the domain Ω into two non-empty subdomains Ω_1 and Ω_2, i.e.,

$$\Omega_1 = \{\mathbf{x} \in \Omega : \boldsymbol{\xi} \cdot \mathbf{x} < c\} \quad \text{and} \quad \Omega_2 = \{\mathbf{x} \in \Omega : \boldsymbol{\xi} \cdot \mathbf{x} > c\}.$$

Let $\chi(\mathbf{x}; \boldsymbol{\xi}, c)$ be a piece-wise constant function defined by

$$\chi(\mathbf{x}; \boldsymbol{\xi}, c) = \begin{cases} \alpha_1, & \mathbf{x} \in \Omega_1, \\ \alpha_2, & \mathbf{x} \in \Omega_2. \end{cases}$$

Lemma 3.2. *Let $p(\mathbf{x})$ be a two-layer neural network function given by*

$$p(\mathbf{x}) = \alpha_1 + \frac{\alpha_2 - \alpha_1}{2\varepsilon}\Big(\sigma(\boldsymbol{\xi} \cdot \mathbf{x} - c + \varepsilon) - \sigma(\boldsymbol{\xi} \cdot \mathbf{x} - c - \varepsilon)\Big)$$

for any $\varepsilon > 0$ such that intersections between the domain Ω and the hyper-planes $\boldsymbol{\xi} \cdot \mathbf{x} = c \pm \varepsilon$ are not empty. Then we have

$$\|\chi - p\|_{0,\Omega} = \Big(\|\chi - p\|_{0,\Omega}^2 + \|\chi_{\boldsymbol{\eta}} - p_{\boldsymbol{\eta}}\|_{0,\Omega}^2\Big)^{1/2} \le \frac{1}{\sqrt{6}}\, D^{(d-1)/2} \left|\alpha_1 - \alpha_2\right| \sqrt{\varepsilon}, \tag{3.6}$$

where $\boldsymbol{\eta}$ is a vector normal to $\boldsymbol{\xi}$ and D is the diameter of the domain Ω.

Proof. Let

$$\Omega_\varepsilon = \Omega_{\varepsilon,1} \cup \Omega_{\varepsilon,2} \equiv \{\mathbf{x} \in \Omega : c - \varepsilon < \boldsymbol{\xi} \cdot \mathbf{x} < c\} \cup \{\mathbf{x} \in \Omega : c < \boldsymbol{\xi} \cdot \mathbf{x} < c + \varepsilon\}.$$

The equality in (3.6) follows from the fact that $\chi_{\boldsymbol{\eta}} - p_{\boldsymbol{\eta}} = 0$. To show the validity of the inequality in (3.6), first we have

$$\chi - p = \begin{cases} \dfrac{\alpha_1 - \alpha_2}{2\varepsilon}\left(\boldsymbol{\xi} \cdot \mathbf{x} - c + \varepsilon\right), & \mathbf{x} \in \Omega_{\varepsilon,1}, \\ \dfrac{\alpha_1 - \alpha_2}{2\varepsilon}\left(\boldsymbol{\xi} \cdot \mathbf{x} - c - \varepsilon\right), & \mathbf{x} \in \Omega_{\varepsilon,2}, \\ 0, & \mathbf{x} \in \Omega \setminus \Omega_\varepsilon. \end{cases}$$

By a rotation of the coordinates, $\mathbf{x} = (s, \mathbf{y})$, it is easy to see that the domain $\Omega_{\varepsilon,1}$ is bounded by the hyper-planes $s = c - \varepsilon$ and $s = c$ and the hyper-surfaces $\varphi_1(s)$ and $\varphi_2(s)$ on $\partial\Omega$. Hence, we have

$$\int_{\Omega_{\varepsilon,1}} \left(\boldsymbol{\xi} \cdot \mathbf{x} - c + \varepsilon\right)^2 d\mathbf{x} = \int_{c-\varepsilon}^{c} \int_{\varphi_1(s)}^{\varphi_2(s)} \left(s - c + \varepsilon\right)^2 d\mathbf{y}\, ds \le D^{d-1} \int_{c-\varepsilon}^{c} \left(s - c + \varepsilon\right)^2 ds = \frac{D^{d-1}}{3}\, \varepsilon^3.$$

In a similar fashion,

$$\int_{\Omega_{\varepsilon,2}} \left(\boldsymbol{\xi} \cdot \mathbf{x} - c - \varepsilon\right)^2 d\mathbf{x} \le \frac{D^{d-1}}{3} \varepsilon^3.$$

The above two inequalities imply

$$\|\chi - p\|_{0,\Omega}^2 \le \left(\frac{\alpha_1 - \alpha_2}{2\varepsilon}\right)^2 \frac{2D^{d-1}}{3} \varepsilon^3 = \frac{D^{d-1}(\alpha_1 - \alpha_2)^2 \varepsilon}{6}.$$

This proves the inequality in (3.6) and, hence, the lemma. □

Assume that u is a piece-wise smooth function with respect to the partition $\{\Omega_1, \Omega_2\}$ such that the jump of u on the interface $\mathcal{P} = \Omega_1 \cap \Omega_2$ is a constant $\alpha_2 - \alpha_1$, i.e.,

$$[\![u]\!]_{\mathcal{P}} \equiv u_2|_{\mathcal{P}} - u_1|_{\mathcal{P}} = \alpha_2 - \alpha_1.$$

Then u has the following decomposition

$$u = \chi(\mathbf{x}; \boldsymbol{\xi}, c) + \hat{u}, \tag{3.7}$$

where $\boldsymbol{\xi}$ is a vector normal to $\boldsymbol{\beta}$. It is easy to see that $\hat{u}$ is continuous in Ω and piece-wise smooth.

Theorem 3.3. *Assume that the advection velocity field $\boldsymbol{\beta}$ is a constant vector field and that $f \in C(\Omega)$. Let u and u_N be the solutions of problems (2.7) and (3.4), respectively. Then we have*

$$\left|\!\left|\!\left| u - u_N \right|\!\right|\!\right|_{\boldsymbol{\beta}} \le C \left(\left|\alpha_1 - \alpha_2\right| \sqrt{\varepsilon} + \inf_{v \in \mathcal{M}(\boldsymbol{\theta}, L)} \left|\!\left|\!\left| \hat{u} - v \right|\!\right|\!\right|_{\boldsymbol{\beta}} \right), \tag{3.8}$$

where $\hat{u} \in C(\Omega)$ is given in (3.7).

Proof. The assumptions on $\boldsymbol{\beta}$ and f imply that the exact solution u has the decomposition in (3.7). Now, (3.8) is a direct consequence of Lemmas 3.1 and 3.2. □

Similar to [7], we evaluate the LS functional numerically. To this end, let

$$\mathcal{T} = \{K : K \text{ is an open subdomain of } \Omega\}$$

be a partition of the domain Ω. Then

$$\mathcal{E}_- = \{E = \partial K \cap \Gamma_- : K \in \mathcal{T}\}$$

is a partition of the inflow boundary Γ_-. Let $\mathbf{x}_K$ and $\mathbf{x}_E$ be the centroids of $K \in \mathcal{T}$ and $E \in \mathcal{E}_-$, respectively. Define the discrete LS functional as follows:

$$\mathcal{L}_{\mathcal{T}}\big(v(\mathbf{x};\boldsymbol{\theta}); \mathbf{f}\big) = \sum_{K \in \mathcal{T}} \left(v_{\boldsymbol{\beta}} + \hat{\gamma}\, v - f\right)^2 (\mathbf{x}_K; \boldsymbol{\theta})\, |K| + \sum_{E \in \mathcal{E}_-} \big(|\boldsymbol{\beta} \cdot \boldsymbol{n}| (v-g)^2\big)(\mathbf{x}_E; \boldsymbol{\theta}) |E|, \tag{3.9}$$

where $|K|$ and $|E|$ are the d and $d-1$ dimensional measures of K and E, respectively. Then the discrete least-squares approximation is to find $u_{\mathcal{T}}^N(\mathbf{x}, \boldsymbol{\theta}^*) \in \mathcal{M}(\boldsymbol{\theta}, L)$ such that

$$\mathcal{L}_{\mathcal{T}}\big(u_{\mathcal{T}}^N(\mathbf{x}, \boldsymbol{\theta}^*); \mathbf{f}\big) = \min_{v \in \mathcal{M}(\boldsymbol{\theta}, L)} \mathcal{L}_{\mathcal{T}}\big(v(\mathbf{x};\boldsymbol{\theta});\, \mathbf{f}\big) = \min_{\boldsymbol{\theta} \in \mathbb{R}^N} \mathcal{L}_{\mathcal{T}}\big(v(\mathbf{x};\boldsymbol{\theta}); \mathbf{f}\big). \tag{3.10}$$

Lemma 3.4. *Let u, u_N, and $u_{\mathcal{T}}^N$ be the solutions of problems (2.7), (3.4), and (3.10), respectively. Then there exists a positive constant C such that*

$$\left|\!\left|\!\left| u - u_{\mathcal{T}}^N \right|\!\right|\!\right|_{\boldsymbol{\beta}} \le C \left(\inf_{v \in \mathcal{M}(\boldsymbol{\theta}, L)} |\!|\!| u - v |\!|\!|_{\boldsymbol{\beta}} + \left|(\mathcal{L} - \mathcal{L}_{\mathcal{T}})(u_N - u_{\mathcal{T}}^N; \mathbf{0})\right| + \left|(\mathcal{L} - \mathcal{L}_{\mathcal{T}})(u - u_N; \mathbf{0})\right| \right). \tag{3.11}$$

Proof. By the triangle inequality, the fact that $\mathcal{L}_{\mathcal{T}}(u_{\mathcal{T}}^N; \mathbf{f}) \le \mathcal{L}_{\mathcal{T}}(u_N; \mathbf{f})$, and the continuity of the homogeneous functional $\mathcal{L}\big(v; \mathbf{0}\big)$ in (2.9), we have

$$\begin{aligned}
\frac{1}{2}\mathcal{L}_{\mathcal{T}}(u_N - u_{\mathcal{T}}^N; \mathbf{0}) &\le \mathcal{L}_{\mathcal{T}}(u_N - u; \mathbf{0}) + \mathcal{L}_{\mathcal{T}}(u - u_{\mathcal{T}}^N; \mathbf{0}) = \mathcal{L}_{\mathcal{T}}(u_N; \mathbf{f}) + \mathcal{L}_{\mathcal{T}}(u_{\mathcal{T}}^N; \mathbf{f}) \\
&\le 2\,\mathcal{L}_{\mathcal{T}}(u_N; \mathbf{f}) = 2\left((\mathcal{L}_{\mathcal{T}} - \mathcal{L})(u_N - u; \mathbf{0}) + \mathcal{L}(u_N - u; \mathbf{0})\right) \\
&\le 2\,(\mathcal{L}_{\mathcal{T}} - \mathcal{L})(u_N - u; \mathbf{0}) + 2M \left|\!\left|\!\left| u - u_N \right|\!\right|\!\right|_{\boldsymbol{\beta}}^2,
\end{aligned}$$

which, together with the coercivity of the homogeneous functional $\mathcal{L}\big(v; \mathbf{0}\big)$ in (2.9), implies that

$$\begin{aligned}
\alpha \left|\!\left|\!\left| u_N - u_{\mathcal{T}}^N \right|\!\right|\!\right|_{\boldsymbol{\beta}}^2 &\le \mathcal{L}\big(u_N - u_{\mathcal{T}}^N; \mathbf{0}\big) = \big(\mathcal{L} - \mathcal{L}_{\mathcal{T}}\big)\big(u_N - u_{\mathcal{T}}^N; \mathbf{0}\big) + \mathcal{L}_{\mathcal{T}}\big(u_N - u_{\mathcal{T}}^N; \mathbf{0}\big) \\
&\le \big(\mathcal{L} - \mathcal{L}_{\mathcal{T}}\big)\big(u_N - u_{\mathcal{T}}^N; \mathbf{0}\big) + 4\,(\mathcal{L}_{\mathcal{T}} - \mathcal{L})(u_N - u; \mathbf{0}) + 4M \left|\!\left|\!\left| u - u_N \right|\!\right|\!\right|_{\boldsymbol{\beta}}^2.
\end{aligned}$$

Now, (3.11) is a direct consequence of the triangle inequality, the above inequality, and Lemma 3.1. This completes the proof of the lemma. □

Lemma 3.4 indicates that the total error of the LSNN approximation with numerical integration is bounded by the approximation error of the neural network and the error of the numerical integration.

4. Initialization of two-layer neural network

The nonlinear optimization in (3.9) usually has many solutions, and the desired one is obtained only if we start from a close enough first approximation. In this section, we briefly describe the initialization process introduced in [21] for two-layer neural network.

To this end, a two-layer ReLU NN with n_1 neurons produces the following set of functions:

$$\mathcal{M}(\theta, 2) = \left\{ c_0 + \sum_{i=1}^{n_1} c_i \sigma(\boldsymbol{\omega}_i \cdot \mathbf{x} - b_i) \,:\, c_i,\, b_i \in \mathbb{R},\ \boldsymbol{\omega}_i \in \mathcal{S}^{d-1} \right\}, \tag{4.1}$$

where $\mathcal{S}^{d-1}$ is the unit sphere in $\mathbb{R}^d$. Let

$$\varphi_0(\mathbf{x}) = 1 \quad \text{and} \quad \varphi_i(\mathbf{x}) = \sigma(\boldsymbol{\omega}_i \cdot \mathbf{x} - b_i) \quad \text{for } i = 1, ..., n_1.$$

For a given input weights and bias

$$\boldsymbol{\omega} = (\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_{n_1}) \quad \text{and} \quad \mathbf{b} = (b_1, ..., b_{n_1}),$$

problem (2.11) may be approximated by finding $u_{n_1} = \sum_{i=0}^{n_1} c_i \varphi_i(\mathbf{x})$ such that

$$a(u_{n_1}, \varphi_j) = f(\varphi_j) \quad \text{for } j = 0, 1, ..., n_1, \tag{4.2}$$

for $j = 0, 1, ..., n_1$. Let

$$A(\boldsymbol{\omega}, \mathbf{b}) = \big(a(\varphi_j, \varphi_i)\big)_{(n_1+1)\times(n_1+1)} \quad \text{and} \quad F(\boldsymbol{\omega}, \mathbf{b}) = \big(f(\varphi_j)\big)_{(n_1+1)\times 1},$$

then the coefficients, $\mathbf{c} = (c_0, c_1, ..., c_n)$, of u_{n_1} is the solution of the system of linear algebraic equations

$$A(\boldsymbol{\omega}, \mathbf{b})\, \mathbf{c} = F(\boldsymbol{\omega}, \mathbf{b}). \tag{4.3}$$

Lemma 4.1. *Assume that hyper-planes $\{\boldsymbol{\omega}_i \cdot \mathbf{x} = b_i\}_{i=1}^{n_1}$ are distinct. Then the coefficient matrix $A(\boldsymbol{\omega}, \mathbf{b})$ is symmetric, positive definite.*

Proof. Obviously, $A(\boldsymbol{\omega}, \mathbf{b})$ is symmetric. Positive definiteness of $A(\boldsymbol{\omega}, \mathbf{b})$ follows from the lower bound in (2.9) and the linear independence of $\{\varphi_i\}_{i=0}^{n_1}$ (see Lemma 2.1 of [21]). □

As discussed in [21], the (breaking) hyper-planes

$$\mathcal{P}_i : \boldsymbol{\omega}_i \cdot \mathbf{x} - b_i = 0 \quad \text{for } i = 1, ..., n_1$$

and the boundary of the domain Ω form a physical partition of the domain Ω. It is then natural to initialize the input weights $\boldsymbol{\omega}$ and bias $\mathbf{b}$ such that the corresponding hyper-planes $\{\mathcal{P}_i\}_{i=1}^{n_1}$ form a uniform partition of the domain Ω. The initial for the output weights and bias $\mathbf{c}$ may be chosen to be the solution of problem (4.3).

5. Numerical experiments

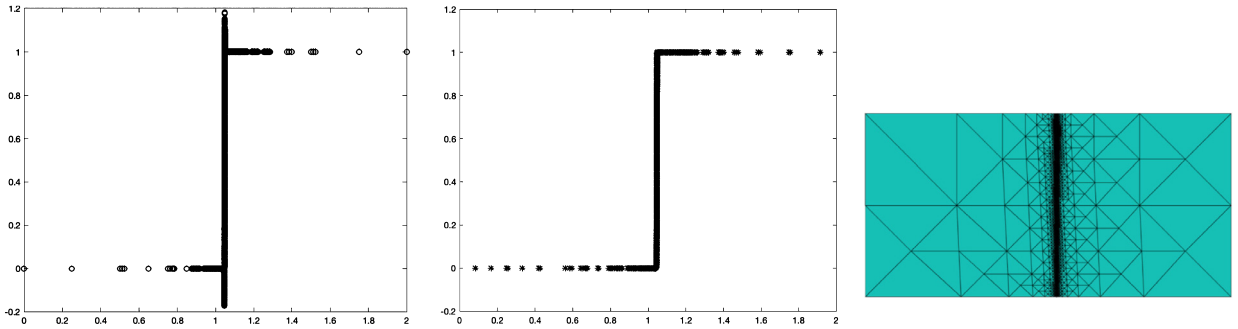
In this section, we present numerical results for test problems with constant, piece-wise constant, or variable advection velocity fields. The solutions of these test problems are discontinuous along an interface which is a line segment, a piece-wise line segment, or a curve.

In all experiments, the integration mesh $\mathcal{T}$ is obtained by uniformly partitioning the domain Ω into identical squares with mesh size $h = 10^{-2}$. The directional derivative in the least-squares functional is approximated by the backward finite difference quotient

$$v_{\boldsymbol{\beta}}(\mathbf{x}_K) \approx \frac{v(\mathbf{x}_K) - v\big(\mathbf{x}_K - \rho\bar{\boldsymbol{\beta}}(\mathbf{x}_K)\big)}{\rho} \tag{5.1}$$

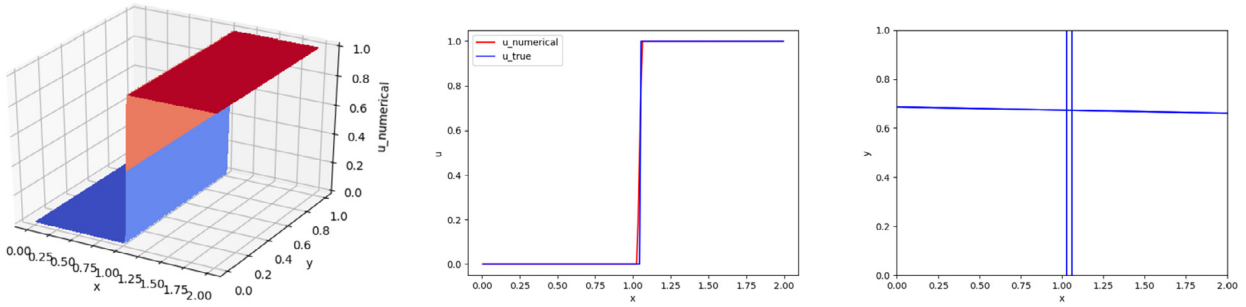
where $\rho \in \mathbb{R}$ is chosen to be smaller than the integration mesh size h, and $\bar{\boldsymbol{\beta}}$ is the unit vector in the $\boldsymbol{\beta}$ direction. The minimization problem in (3.9) is solved numerically by the Adam version of gradient descent [19], and variable learning rate is used during the training.

Let u be the exact solution of problem (2.2) and $\bar{u}_{\mathcal{T}}^N$ be the LSNN approximation. Tables 1–6 report the numerical errors in the relative L^2, $V_{\boldsymbol{\beta}}$, and graph norms. In these tables, a network structure is expressed by 2-n-1 for a two-layer network with n neurons, by 2-n_1-n_2-1 for a three-layer network with n_1 and n_2 neurons in the respective first and second layers, and so on. Figs. 2–7 depict the traces of the exact solution and the numerical approximation on a plane perpendicular to both the x_1x_2-plane and the discontinuous interface, which accurately illustrate the quality of the numerical approximation.



(a) C-LSFEM vertical cross section (b) flux-based LSFEM vertical cross section (c) An adaptively refined mesh

Fig. 1. Numerical results in [22] of the problem with discontinuity along a vertical line segment.



(a) Numerical solution $\bar{u}_{\mathcal{T}}^{N}$ (b) Vertical cross section on $y = 1$ (c) Network breaking lines

Fig. 2. Approximation results of the problem with discontinuity along a vertical line segment.

5.1. Problems with a constant advection velocity fields

In this section, we present numerical results for two test problems with constant advection velocity fields whose solutions are piece-wise constants (see, e.g., [22]). A two-layer neural network is employed and the network is initialized by the method described in section 4.

5.1.1. Discontinuity along a vertical line segment

The first test problem is the equation in (2.2) with the domain $\Omega = (0, 2) \times (0, 1)$, the inflow boundary $\Gamma_- = \{(x, 0) : x \in (0, 2)\}$, a constant advection velocity field $\boldsymbol{\beta} = (0, 1)^T$, $\gamma = f = 0$, and the inflow boundary data $g(x) = 0$ for $x \in (0, \pi/3)$ and $g(x) = 1$ for $x \in (\pi/3, 2)$. Let $\Omega_1 = \{(x, y) \in \Omega : 0 < x < \pi/3\}$ and $\Omega_2 = \{(x, y) \in \Omega : \pi/3 < x < 2\}$, it is then easy to see that the exact solution is a piece-wise constant given by

$$u(x, y) = \begin{cases} 0, & (x, y) \in \Omega_1, \\ 1, & (x, y) \in \Omega_2. \end{cases}$$

The discontinuous interface is the vertical line $x = \pi/3$.

This problem was used to test various adaptive least-squares finite element methods in [22]. In particular, the discontinuous interface $x = \pi/3$ was chosen so that if the initial mesh does not align with the interface, so is the mesh generated by either global or local mesh refinements.

Numerical results in [22] (see Fig. 1) showed that the conforming least-squares finite element method (C-LSFEM) exhibits the Gibbs phenomena even with very fine mesh and that the newly developed flux-based LSFEM in [22] using a pair of the lowest-order elements is able to avoid overshooting on an adaptively refined mesh.

The LSNN method is implemented with $\rho = h/2$ and a fixed learning rate 0.003 with 20000 iterations. Our first set of experiments are done by using networks: 2-200-1 and 2-25-15-15-1. These two networks have 601 and 705 parameters, respectively, and provide good approximations (similar to Fig. 2(a,b)) to the exact solution.

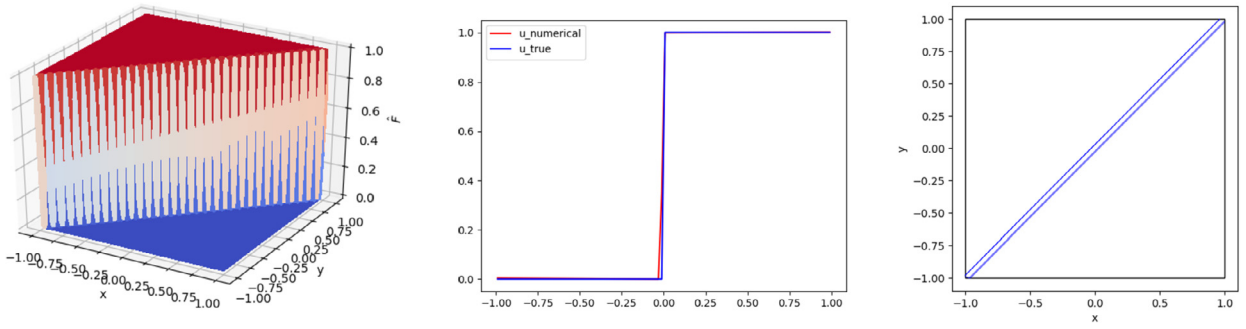
Lemma 3.2 indicates that a two-layer network with 2 neurons is sufficient to approximate the exact solution well. Our second set of experiments are done by using networks: 2-2-1 and 2-4-1 with the respective 7 and 13 parameters. The 2-2-1 network fails to approximate the exact solution when the initial breaking lines are chosen to be the vertical line $x = 1$ and the horizontal line $y = 1/2$. This is because the iterative solver of the nonlinear optimization is not able to move the initial

Table 1
Relative errors of the problem with discontinuity along a vertical line segment.

Network structure	$\frac{\|u-\bar{u}^N_{\mathcal{T}}\|_0}{\|u\|_0}$	$\frac{\|\|u-\bar{u}^N_{\mathcal{T}}\|\|_\beta}{\|\|u\|\|_\beta}$	$\frac{\mathcal{L}^{1/2}(\bar{u}^N_{\mathcal{T}};\mathbf{f})}{\mathcal{L}^{1/2}(\bar{u}^N_{\mathcal{T}};\mathbf{0})}$	Parameters
2-4-1	0.058046	0.058304	0.050491	13
2-200-1	0.058745	0.058926	0.048537	601

Table 2
Relative errors of the problem with discontinuity along the diagonal.

Network structure	$\frac{\|u-\bar{u}^N_{\mathcal{T}}\|_0}{\|u\|_0}$	$\frac{\|\|u-\bar{u}^N_{\mathcal{T}}\|\|_\beta}{\|\|u\|\|_\beta}$	$\frac{\mathcal{L}^{1/2}(\bar{u}^N_{\mathcal{T}};\mathbf{f})}{\mathcal{L}^{1/2}(\bar{u}^N_{\mathcal{T}};\mathbf{0})}$	Parameters
2-4-1	0.393864	0.393871	0.126095	13
2-6-1	0.073534	0.073826	0.067531	19



(a) Network approximation $\bar{u}^N_{\mathcal{T}}$ (b) Vertical cross section along $y = -x$ (c) Network breaking lines

Fig. 3. Approximation results of the problem with discontinuity along the diagonal.

horizontal breaking line to the right place. The initial breaking lines for the 2-4-1 network are chosen to be the vertical lines $x = 2/3$ and $x = 4/3$ and the horizontal lines $y = 1/3$ and $y = 2/3$.

Errors of numerical results are presented in Table 1. The second and third columns in Table 1 show that the approximation of the small network is slightly more accurate than that of the large network while the values of the loss functions are reversed. This indicates that the large network is trapped in a local minimum. The numerical solution of the 4-neuron network is depicted in Fig. 2(a). The traces of the exact and numerical solutions on the plane $y = 1$ are depicted in Fig. 2(b), which shows no oscillation. Fig. 2(c) displays breaking lines of the network with two vertical lines $x = 1.02882$ and $x = 1.06114$ closing to the interface $x = \pi/3$. This indicates that breaking lines of neural network are capable of automatically adapting to the discontinuous interface. This simple test problem shows that the LSNN method out-performs the traditional mesh-based numerical methods.

5.1.2. Discontinuity along the diagonal

The second test problem is again equation (2.2) with a constant advection velocity vector and a piece-wise constant inflow boundary condition. Specifically, $\boldsymbol{\beta} = (1,1)^T/\sqrt{2}$, $\Omega = (-1,1)^2$, $\Gamma_- = \Gamma^1_- \cup \Gamma^2_- \equiv \{(-1,y) : y \in (-1,1)\} \cup \{(x,-1) : x \in (-1,1)\}$, $\gamma = 1$, g and f are piece-wise constants given by

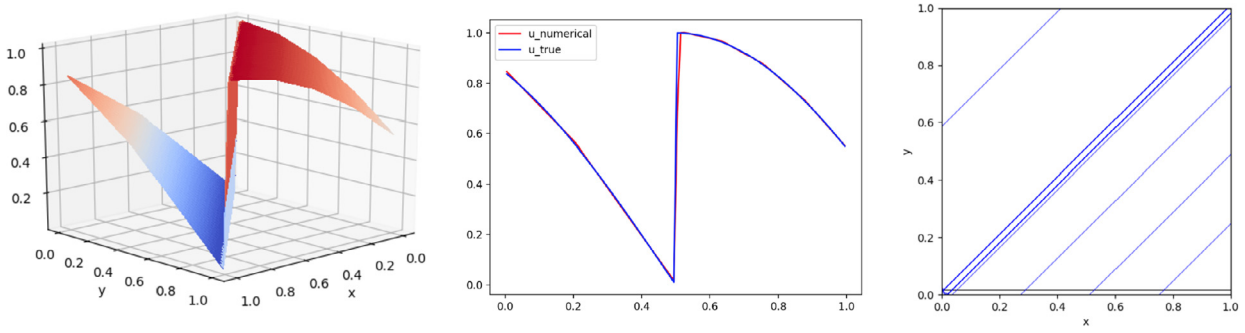
$$g(x,y) = \begin{cases} 1, & (x,y) \in \Gamma^1_-, \\ 0, & (x,y) \in \Gamma^2_-, \end{cases} \quad \text{and} \quad f(x,y) = \begin{cases} 1, & (x,y) \in \Omega_1, \\ 0, & (x,y) \in \Omega_2, \end{cases}$$

where $\Omega_1 = \{(x,y) \in \Omega : y > x\}$ and $\Omega_2 = \{(x,y) \in \Omega : y < x\}$. The exact solution of the test problem is $u(x,y) = f(x,y)$ with the discontinuous interface: $y = x$.

The LSNN method is implemented with $\rho = h/2$ and a fixed learning rate 0.003 with 20000 iterations for two networks: 2-4-1 and 2-6-1. The numerical results are presented in Table 2 which imply that the 2-4-1 network fails to accurately approximate the solution. Fig. 3 shows the NN approximation of the 2-6-1 network. The traces of the exact and numerical solutions on the plane $y = -x$ are depicted in Fig. 3(b). Clearly, the LSNN method with only 19 parameters approximates the exact solution accurately without the Gibbs phenomena. This test problem shows that the LSNN method is able to rotate and shift the initial breaking lines to approximate the discontinuous interface.

Table 3
Relative errors of the problem with a piece-wise smooth solution.

Network structure	$\frac{\|u-\tilde{u}^N_{\mathcal{T}}\|_0}{\|u\|_0}$	$\frac{\|\|u-\tilde{u}^N_{\mathcal{T}}\|\|_\beta}{\|\|u\|\|_\beta}$	$\frac{\mathcal{L}^{1/2}(\tilde{u}^N_{\mathcal{T}};\mathbf{f})}{\mathcal{L}^{1/2}(\tilde{u}^N_{\mathcal{T}};\mathbf{0})}$	Parameters
2-20-1	0.110745	0.110754	0.035571	61
2-30-1	0.107525	0.107641	0.013568	91
2-40-1	0.101411	0.101413	0.003509	121



(a) Network approximation $\tilde{u}^N_{\mathcal{T}}$ (b) Vertical cross section along $y = 1 - x$ (c) Network breaking lines

Fig. 4. Approximation results of the problem with a piece-wise smooth solution.

5.2. Problem with a piecewise smooth solution

The third test problem is a modification of the second test problem by changing the inflow boundary condition from the piece-wise constant to a discontinuous piece-wise smooth function and the domain from $\Omega = (-1, 1)^2$ to $\Omega = (0, 1)^2$, i.e.,

$$g(x, y) = \begin{cases} \sin(y), & (x, y) \in \Gamma^1_- = \{(0, y) : y \in (0, 1)\}, \\ \cos(x), & (x, y) \in \Gamma^2_- = \{(x, 0) : x \in (0, 1)\}. \end{cases}$$

Set $\gamma = f = 0$, the exact solution of this test problem is

$$u(x, y) = \begin{cases} \sin(y - x), & (x, y) \in \Omega_1 = \{(x, y) \in (0, 1)^2 : y > x\}, \\ \cos(x - y), & (x, y) \in \Omega_2 = \{(x, y) \in (0, 1)^2 : y < x\}. \end{cases}$$

The LSNN method is employed with $\rho = h/2$ and a fixed learning rate 0.003 for 30000 iterations. Numerical results of three network models are reported in Table 3 and the first two models fail to approximate the solution well. Fig. 4 presents the NN approximation of the 2-40-1 network. The traces of the exact and numerical solutions on the plane $y = 1 - x$ are depicted in Fig. 4(b), which exhibits no oscillation. It is expected that the network with additional neurons is needed in order to approximate the solution well since the solution of the test problem is a piece-wise smooth function. Moreover, this test problem conforms Theorem 3.3 that a piece-wise smooth function having a constant jump along a line segment discontinuous interface may be approximated well by a two-layer network.

5.3. Problem with two discontinuous interfaces

The fourth test problem is again a modification of the second test problem by changing the domain to $\Omega = (-1, 1) \times (0, 1)$, the inflow boundary condition to a combination of jumps and smooth function

$$g(x, y) = \begin{cases} \sin\left(\frac{\pi(x-y+0.9)}{0.3}\right), & (x, y) \in \Gamma^1_- = \{(x, 0) : x \in (-0.9, -0.6)\}, \\ -1, & (x, y) \in \Gamma^2_- = \{(x, 0) : x \in (-0.2, 0.1)\}, \\ 0, & (x, y) \in \Gamma_- \setminus (\Gamma^1_- \cup \Gamma^2_-) \end{cases}$$

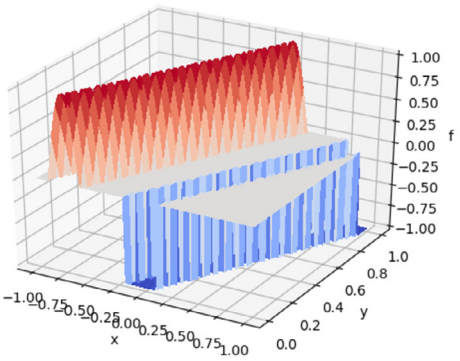
with the inflow boundary

$$\Gamma_- = \{(x, 0) : x \in (-1, 1)\} \cup \{(-1, 0)\} \cup \{(-1, y) : y \in (0, 1)\}.$$

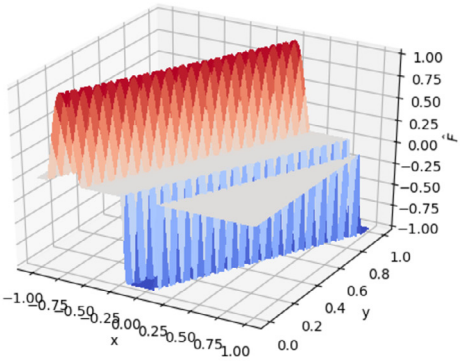
Set f as

Table 4
Relative errors of the problem with two discontinuous interfaces.

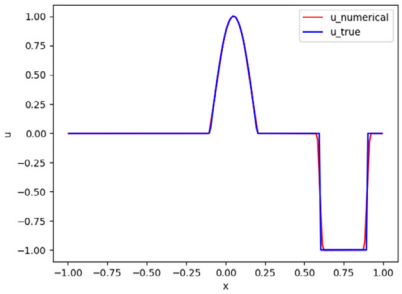
Network structure	$\frac{\|u-\bar{u}^N_{\mathcal{T}}\|_0}{\|u\|_0}$	$\frac{\|\|u-\bar{u}^N_{\mathcal{T}}\|\|_\beta}{\|\|u\|\|_\beta}$	$\frac{\mathcal{L}^{1/2}(\bar{u}^N_{\mathcal{T}};\mathbf{f})}{\mathcal{L}^{1/2}(\bar{u}^N_{\mathcal{T}};\mathbf{0})}$	Parameters
2-20-1	0.363573	0.392153	0.393907	61
2-30-1	0.147767	0.152132	0.132542	91
2-34-1	0.117451	0.120213	0.112463	103



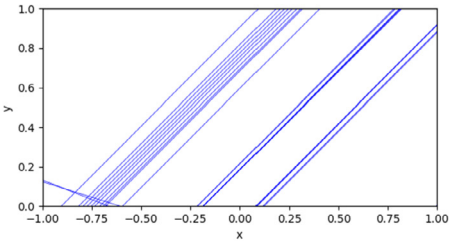
(a) Exact solution u



(b) Network approximation $\bar{u}^N_{\mathcal{T}}$



(c) Traces of the exact solution and approximation $\bar{u}^N_{\mathcal{T}}$ on the plane $y = 0.8$



(d) Network breaking lines

Fig. 5. Approximation results of the problem with two discontinuous interfaces.

$$f(x,y) = \begin{cases} \sin\left(\frac{\pi(x-y+0.9)}{0.3}\right), & (x,y) \in \Upsilon_1 = \{(x,y) \in \Omega : -0.9 < x-y < -0.6\}, \\ -1, & (x,y) \in \Upsilon_2 = \{(x,y) \in \Omega : -0.2 < x-y < 0.1\}, \\ 0, & (x,y) \in \Omega \setminus (\Upsilon_1 \cup \Upsilon_2), \end{cases}$$

then the exact solution of the test problem is $u(x,y) = f(x,y)$ with two discontinuous interfaces $y = x+0.2$ and $y = x-0.1$, respectively.

The LSNN method is implemented with $\rho = h/2$ and an adaptive learning rate which starts with 0.01 and decreases by 0.002 for every 20000 iterations. The total number of iterations is 80000. We observed from the experiment that adding a weight α to the inflow boundary loss in (3.9) is helpful for the training. Empirically, we choose $\alpha = 10$ and report the numerical results for three respective network structures in Table 4. The results suggest that the first 2-20-1 network model fails to approximate the solution well due to the possibility of training and/or insufficient number of neurons. Starting with a 2-30-1 network and applying the adaptive neuron enhancement strategy [21] once, the 2-34-1 network provides an accurate approximation (see Table 4 and Fig. 5). The traces of the exact and numerical solutions are depicted on the plane $y = 0.8$ in Fig. 5(c). This test problem shows that the LSNN method using a small number of DoF is capable of approximating a discontinuous solution containing a smooth extrema without oscillations.

Table 5
Relative errors of the problem with a piece-wise constant advection velocity field.

Network structure	$\frac{\|u-\tilde{u}^N_{\mathcal{T}}\|_0}{\|u\|_0}$	$\frac{\|\|u-\tilde{u}^N_{\mathcal{T}}\|\|_\beta}{\|\|u\|\|_\beta}$	$\frac{\mathcal{L}^{1/2}(\tilde{u}^N_{\mathcal{T}};\mathbf{f})}{\mathcal{L}^{1/2}(\tilde{u}^N_{\mathcal{T}};\mathbf{0})}$	Parameters
2-30-1	0.487306	0.556949	0.386919	91
2-200-1	0.317839	0.402699	0.259592	601
2-5-5-1	0.086122	0.086131	0.016945	46

5.4. Problem with a piece-wise constant advection velocity field

The fifth test problem is equation (2.2) defined on $\Omega = (0,1)^2$ with $\gamma = f = 0$ and a piece-wise constant advection velocity field. Specifically, the advection velocity field is given by

$$\boldsymbol{\beta} = \begin{cases} (1-\sqrt{2},1)^T, & (x,y) \in \Upsilon_1 = \{(x,y) \in \Omega : y < x\}, \\ (-1,\sqrt{2}-1)^T, & (x,y) \in \Upsilon_2 = \{(x,y) \in \Omega : y \geq x\}. \end{cases} \tag{5.2}$$

and, hence, the inflow boundary of the problem is

$$\Gamma_- = \{(x,0) : x \in (0,1)\} \cup \{(1,0)\} \cup \{(1,y) : y \in (0,1)\}. \tag{5.3}$$

Let $\Gamma^1_- = \{(x,0) : x \in (0,a)\}$ with $a = 43/64$. For the inflow boundary condition

$$g(x,y) = \begin{cases} -1, & (x,y) \in \Gamma^1_-, \\ 1, & (x,y) \in \Gamma^2_- = \Gamma_- \setminus \Gamma^1_-, \end{cases} \tag{5.4}$$

the exact solution is a piece-wise constant: $u = -1$ in Ω_1 and $u = 1$ in Ω_2, where $\Omega_2 = \Omega \setminus \bar{\Omega}_1$ and

$$\Omega_1 = \{\mathbf{x} \in \Upsilon_1 : \boldsymbol{\xi}_1 \cdot \mathbf{x} < a\} \cup \{\mathbf{x} \in \Upsilon_2 : \boldsymbol{\xi}_2 \cdot \mathbf{x} < a\}.$$

Here, $\boldsymbol{\xi}_1 = (1, \sqrt{2}-1)^T$ and $\boldsymbol{\xi}_2 = (\sqrt{2}-1, 1)^T$ are vectors normal to $\boldsymbol{\beta}|_{\Upsilon_1}$ and $\boldsymbol{\beta}|_{\Upsilon_2}$, respectively.

The LSNN method with $\rho = h/2$ and a fixed learning rate 0.003 with 50000 iterations is implemented for networks: 2-30-1, 2-200-1, and 2-5-5-1. Initialization of the first layer is done by the approach described in section 4, and that of the subsequent layers are randomly generated. The numerical results are presented in Table 5 and Fig. 6, and the figures of the two-layer network is for the 2-200-1 model. The traces of the exact and numerical solutions on the plane $x = 0$ and the breaking lines of these two networks are depicted in Fig. 6(c,d) and Fig. 6(e,f), respectively.

Clearly, the two-layer network with 200 neurons (over 600 parameters) fails to approximate the solution well in average (see Table 5) and point-wise (see Fig. 6). A three-layer network with less than 8% of parameters outperforms this large two-layer network in every aspects including breaking lines. Comparing these two networks, a three-layer network is more suitable than a two-layer network to accurately approximate the solution having a constant jump along a piece-wise line segment discontinuous interface.

Remark 5.1. Due to the random generation of some parameters, the training of 2-5-5-1 network is replicated five times and the best result is reported. We observe from the training process that the network may get trapped in a local minimum and fails to accurately approximate the solution. To address such issue, we introduce an adaptive process in [6] for obtaining a good initialization which is crucial for nonlinear optimization problems.

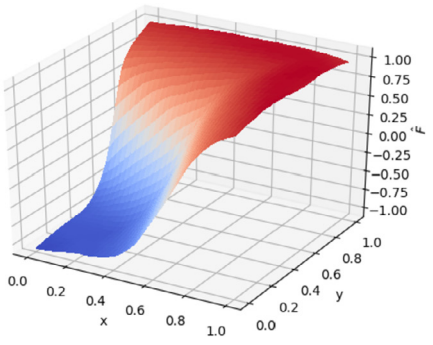
Below we show theoretically that a three-layer neural network is sufficient for approximating the solution well (see Lemma 5.2 below). To make it slightly general, let

$$\chi = \begin{cases} \alpha_1, & \mathbf{x} \in \Omega_1, \\ \alpha_2, & \mathbf{x} \in \Omega_2. \end{cases}$$

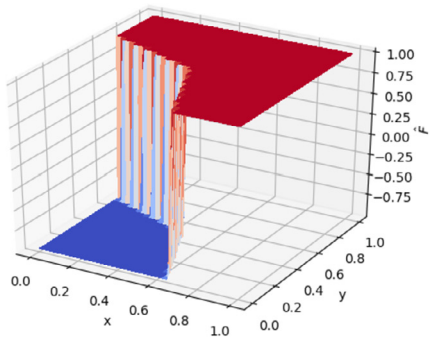
Without loss of generality, assume that $\alpha_1 < \alpha_2$. Let $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$ be two-layer neural network functions given by

$$p_i(\mathbf{x}) = \alpha_1 + \frac{\alpha_2 - \alpha_1}{2\varepsilon}\Big(\sigma(\boldsymbol{\xi}_i \cdot \mathbf{x} - a + \varepsilon) - \sigma(\boldsymbol{\xi}_i \cdot \mathbf{x} - a - \varepsilon)\Big)$$

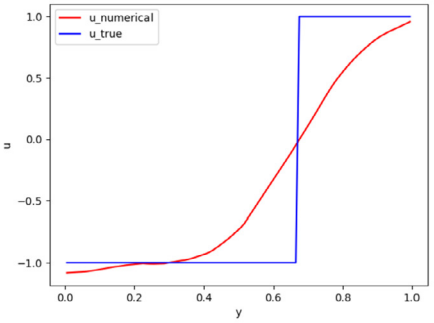
for any $\varepsilon > 0$ such that intersections between the domain Ω and the hyper-planes $\boldsymbol{\xi}_i \cdot \mathbf{x} = a \pm \varepsilon$ are not empty.



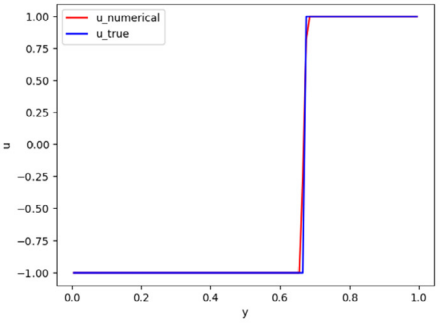
(a) 2-layer NN approximation $\bar{u}_{\mathcal{T}}^N$



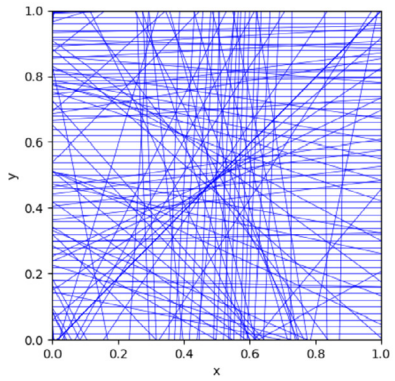
(b) 3-layer NN approximation $\bar{u}_{\mathcal{T}}^N$



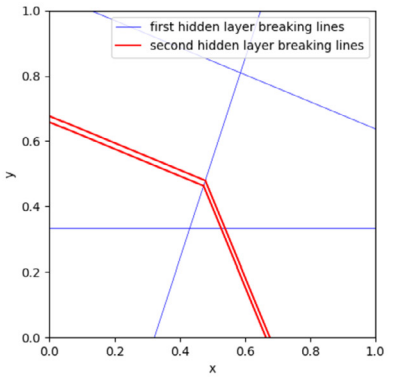
(c) 2-layer NN vertical cross section on $x = 0$



(d) 3-layer NN vertical cross section on $x = 0$



(e) 2-layer NN breaking lines



(f) 3-layer NN breaking lines

Fig. 6. Approximation results of the problem with a piece-wise constant advection velocity field.

Lemma 5.2. *Let* $p(\mathbf{x}) = \max\{p_1(\mathbf{x}), p_2(\mathbf{x})\}$, *then we have*

$$\|\chi - p\|_{0,\Omega} = \left(\|\chi - p\|_{0,\Omega}^2 + \|\chi_\beta - p_\beta\|_{0,\Omega}^2\right)^{1/2} \leq \sqrt{\frac{2}{3}}\, D^{(d-1)/2} |\alpha_1 - \alpha_2| \sqrt{\varepsilon}, \tag{5.5}$$

where D *is the diameter of the domain* Ω.

Proof. Since $p(\mathbf{x}) = p_i(\mathbf{x})$ in Υ_i for $i = 1, 2$ and $\Omega = \Upsilon_1 \cup \Upsilon_2$, we have

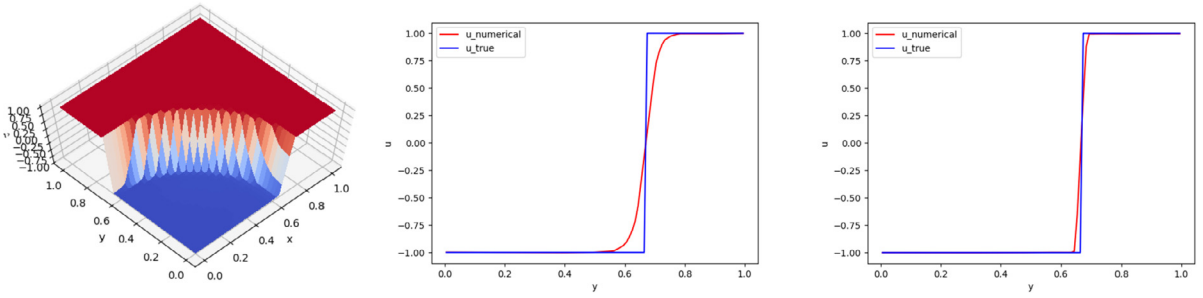
$$\|\chi - p\|_{0,\Omega}^2 = \|\chi - p_1\|_{0,\Upsilon_1}^2 + \|\chi - p_2\|_{0,\Upsilon_2}^2.$$

Combining with the fact that $\chi_\beta - p_\beta = 0$ in Ω, (5.5) is then a direct consequence of Lemma 3.2. □

Similar as the discussion in [14], the maximum operation can be constructed by using an additional hidden layer of the ReLU network with 4 neurons:

Table 6
Relative errors of the problem with a variable advection velocity field.

Network structure	$\frac{\|u-\tilde{u}^N_{\mathcal{T}}\|_0}{\|u\|_0}$	$\frac{\|\|u-\tilde{u}^N_{\mathcal{T}}\|\|_\beta}{\|\|u\|\|_\beta}$	$\frac{\mathcal{L}^{1/2}(\tilde{u}^N_{\mathcal{T}};\mathbf{f})}{\mathcal{L}^{1/2}(\tilde{u}^N_{\mathcal{T}};\mathbf{0})}$	Parameters
2-40-40-1	0.146226	0.187823	0.108551	1761
2-30-30-30-1	0.109266	0.122252	0.039993	1951



(a) 4-layer network approximation (b) 3-layer NN cross section on $x = 0$ (c) 4-layer NN cross section on $x = 0$

Fig. 7. Approximation results of the problem with a variable advection velocity field.

$$\max\{a,b\} = \frac{a+b}{2} + \frac{|a-b|}{2} = \mathbf{v}\sigma\left(\boldsymbol{\omega}\begin{bmatrix} a \\ b \end{bmatrix}\right)$$

where the row vector and the 4×2 matrix are given by

$$\mathbf{v} = \frac{1}{2}[1,-1,1,1] \quad \text{and} \quad \boldsymbol{\omega} = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix},$$

respectively. Then this lemma indicates that a three-layer neural network is sufficient when the interface consists of two line segments.

Remark 5.3. In a similar fashion, a three-layer network can be constructed to approximate the solution with the interface consisting of more than two line segments.

5.5. Problem with a variable advection velocity field

The last test problem is equation (2.2) defined on the domain $\Omega = (0,1)^2$ with a *variable* advection velocity field $\boldsymbol{\beta} = (-y,x)^T$ and $\gamma = f = 0$ (see, e.g., [2,23]). With the inflow boundary condition g given in (5.4), the exact solution is a piece-wise constant given by

$$u(x,y) = \begin{cases} -1, & (x,y) \in \Omega_1, \\ 1, & (x,y) \in \Omega_2, \end{cases} \tag{5.6}$$

where $\Omega_1 = \{(x,y) \in \Omega : x^2 + y^2 < a^2\}$ and $\Omega_2 = \{(x,y) \in \Omega : x^2 + y^2 > a^2\}$.

For the LSNN method, again we use a uniform integration mesh $\mathcal{T}$ with the mesh size $h = 10^{-2}$; the finite difference quotient in (5.1) is calculated with $\rho = h/10$ to avoid using values on both sides of the interface. Instead of intricately choosing the ρ value, a robust approach will be developed in a forthcoming paper. Besides, the parameters are initialized by the method described in section 4 for the first layer and randomly for the subsequent layers. The learning rate starts with 0.005, and is reduced by half for every 50000 iterations. This learning rate decay strategy is used with 150000 iterations. Due to the random initialization of some parameters, numerical experiments are replicated three times and the best results for the three- and four-layer networks are reported in Table 6 and Fig. 7. The traces of the exact and numerical solutions at the plane $x = 0$ are depicted in Fig. 7 (b) and (c) for the respective three- and four-layer networks. As shown in Fig. 7 (b), the LSNN approximation of the three-layer network with 40 neurons at each layer smears the discontinuity. A careful examination of the iterative process, it seems to us that the smear is due to the initialization (see Fig. 9).

6. Method of model continuation

As observed from our numerical experiments for the test problem with a curved discontinuous interface, initial of the parameters plays an important role in training neural networks. This is because the high dimensional nonlinear optimization

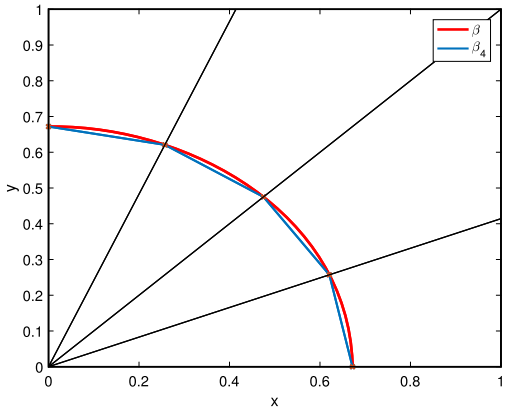


Fig. 8. Discontinuous interface.

usually have many solutions. Without a good initial, our previous simulations rely on over-parameterized neural networks to approximate the underlying problem well. The strategy of over-parameterization is computationally expensive.

Based on our numerical experiments in the previous sections, to generate a good initial for the parameters, we introduce the method of continuation through models for the advection-reaction problem in (2.2) with a variable advection velocity field $\boldsymbol{\beta}(\mathbf{x})$. To this end, let $\{\boldsymbol{\beta}_n(\mathbf{x})\}$ be a sequence of piece-wise constant vector fields. Consider the following advection-reaction problem with the advection velocity field $\boldsymbol{\beta}_n(\mathbf{x})$:

$$\begin{cases} (u_n)_{\boldsymbol{\beta}_n} + \hat{\gamma}\, u_n &= f, \quad \text{in } \Omega, \\ u_n &= g, \quad \text{on } \Gamma_-. \end{cases} \tag{6.1}$$

Let u be the solution of (2.2), it is easy to see that $u - u_n$ satisfies

$$\begin{cases} (u-u_n)_{\boldsymbol{\beta}_n} + \hat{\gamma}\,(u-u_n) &= u_{\boldsymbol{\beta}_n} - u_{\boldsymbol{\beta}}, \quad \text{in } \Omega, \\ u - u_n &= 0, \quad \text{on } \Gamma_-, \end{cases} \tag{6.2}$$

which, together with the stability estimate in (2.10), implies

$$\|u-u_n\|_{0,\Omega} \le |||u-u_n|||_{\boldsymbol{\beta}_n} \le C\, \|u_{\boldsymbol{\beta}_n} - u_{\boldsymbol{\beta}}\|_{0,\Omega} = C \left(\int_\Omega \left((\boldsymbol{\beta}_n - \boldsymbol{\beta}) \cdot \nabla u \right) d\mathbf{x} \right)^{1/2}.$$

Hence, if $\boldsymbol{\beta}_n$ is a good approximation to $\boldsymbol{\beta}$, then u_n is a good approximation to u. This indicates that (6.1) provides a continuation process on the parameter n for (2.2).

For the test problem in section 5.4, since streamlines of the advection velocity field $\boldsymbol{\beta} = (-y, x)^T$ are quarter circles in $\Omega = (0,1)^2$ oriented counterclockwise, it is natural to approximate the quarter-circle by n line segments. To this end, let $t_i = \dfrac{i\pi}{2n}$ for $i = 0, 1, ..., n$ and

$$\Upsilon_{i+1} = \{(x,y) \in \Omega : (\sin t_i)x < (\cos t_i)y \ \text{ and } \ (\sin t_{i+1})x \ge (\cos t_{i+1})y\}.$$

Then $\{\Upsilon_{i+1}\}_{i=0}^{n-1}$ forms a partition of Ω (see Fig. 8 for $n = 4$). This type of approximations leads to

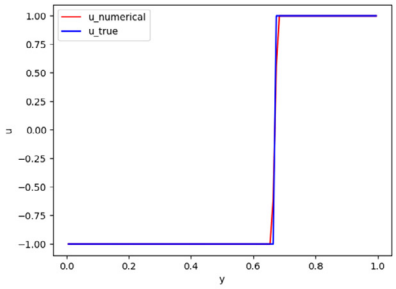
$$\boldsymbol{\beta}_n = (\cos t_{i+1} - \cos t_i, \sin t_{i+1} - \sin t_i)^T \quad \text{in } \Upsilon_{i+1}$$

for $i = 0, 1, ..., n-1$. Note that $\boldsymbol{\beta}_2$ is the same vector field given in (5.2). Hence, (6.1) with $n = 2$ and the test problem in section 5.3 are the same.

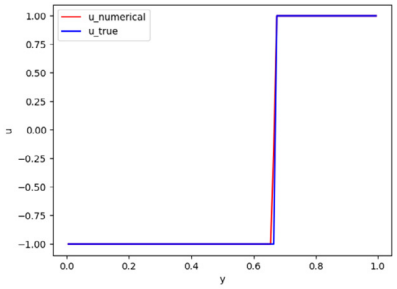
The method of model continuation starts with a three-layer neural network (2-5-5-1) to approximate u_2 (see the third row of Table 5 and Fig. 6 (b,d)). This trained network is used as an initial for the parameters in the hidden layers of the 2-6-6-1 network to approximate u_3 by randomly generated the parameters of new neurons. The initial for the output weights and bias may be chosen as the solution of the system (4.3). The adaptive learning rate strategy which starts with 0.01 and decays by 20% for every 50000 iterations is implemented with the method. The networks for u_n with $n = 4, 5$ and for the test problem in section 5.4 are initialized sequentially in a similar fashion. Numerical results for approximating u_n and u are reported in Table 7, and the traces of the exact and numerical solutions at the plane $x = 0$ are depicted in Fig. 9. The third and fourth columns show that the difficulty of the corresponding problems increase as the number of line segments increase. The fifth column shows that u_n approaches to u monotonously. Comparing Table 5 with the last row of Table 6, it is clear that the method of model continuation is capable of reducing the size of the network significantly.

Table 7
Relative errors of the problem with discontinuity along line segments.

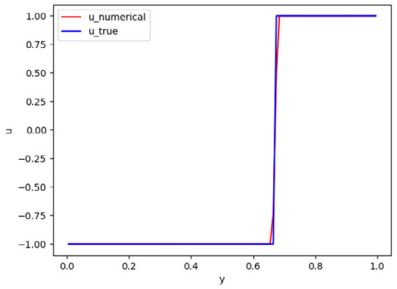
n	Network structure	$\frac{\|u_n-\tilde{u}_{\mathcal{T}}^N\|_0}{\|u_n\|_0}$	$\frac{\|\|\|u_n-\tilde{u}_{\mathcal{T}}^N\|\|\|}{\|\|\|u_n\|\|\|}$	$\frac{\|u-\tilde{u}_{\mathcal{T}}^N\|}{\|u\|}$	$\frac{\mathcal{L}^{1/2}(\tilde{u}_{\mathcal{T}}^N;\mathbf{f})}{\mathcal{L}^{1/2}(\tilde{u}_{\mathcal{T}}^N;\mathbf{0})}$	Parameters
3	2-6-6-1	0.075817	0.080026	0.244483	0.059422	61
4	2-6-6-1	0.104372	0.110954	0.216481	0.064744	61
5	2-8-8-1	0.097836	0.109648	0.135606	0.049938	97
curve	2-25-25-1	0.141261	0.187616	0.141261	0.077233	726



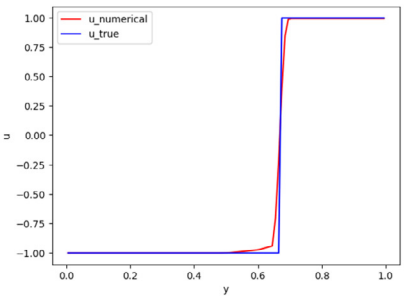
(a) Vertical cross section of the problem (6.1) with $n = 3$ on the plane $x = 0$



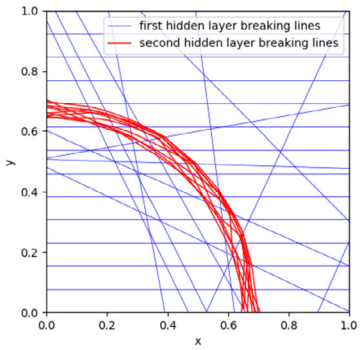
(b) Vertical cross section of the problem (6.1) with $n = 4$ on the plane $x = 0$



(c) Vertical cross section of the problem (6.1) with $n = 5$ on the plane $x = 0$



(d) Vertical cross section of the original problem (5.6) on the plane $x = 0$



(e) Breaking lines of the original problem (5.6)

Fig. 9. Approximation results using the method of model continuation.

7. Discussions and conclusions

We proposed the LSNN method for solving the linear advection-reaction problem. The least-squares formulation, based on a direct application of the least-squares principle to the underlying problem, does not require additional smoothness of the solution if $f \in L^2(\Omega)$. In the V_β norm, the LSNN approximation is proved to be quasi-optimal, i.e., the error of the LSNN approximation is bounded above by the approximation error of the network.

A major challenge in numerical simulation of hyperbolic partial differential equations is the discontinuity of their solutions. For the linear transport problem in two dimensions, by decomposing the discontinuous solution into the sum of a piece-wise constant function and a continuous piece-wise smooth function, we are able to show theoretically and numerically that the LSNN method using a (at most) three-layer ReLU neural network is capable of approximating the discontinuous solution accurately without oscillation. In particular, the piece-wise constant solution can be approximated well by a ReLU network with a small number of neurons.

Numerical results presented in section 5 show that it is important to use a proper neural network in order to accurately approximate the solution of the underlying problem with fewer parameters. How to automatically design such a proper network, in terms of their width and depth, is an open and fundamental question for numerically solving partial differential equations within the prescribed accuracy. Following our recent paper on adaptive neuron enhancement method [21], this will be addressed in the forthcoming paper.

The procedure of training the value of the parameters is a problem in non-convex optimization which usually has many solutions and are complicated and computationally demanding. In order to obtain a desired solution, we introduced a

method of model continuation for providing a good first approximation. Numerical results show that this method is effective for reducing the number of the parameters of the network. Moreover, a good initial is very helpful in training as well.

Nevertheless, training is still a challenging problem since the learning rate of the methods of the gradient type is difficult to tune. A reasonably good learning rate can only be discovered through the method of trial and error. Using NNs to solve PDEs is relatively new, developing fast solvers is an open and challenging problem and requires lots of efforts from numerical analysts. Because of its great potential and many difficulties at the same time, machine learning is a hot research topic in scientific computing.

CRediT authorship contribution statement

Conception and design of study: Z. Cai, J. Chen, M. Liu. Acquisition of data: J. Chen, M. Liu. Analysis and/or interpretation of data: Z. Cai, J. Chen, M. Liu. Drafting the manuscript: Z. Cai, J. Chen, M. Liu. Revising the manuscript critically for important intellectual content: Z. Cai, J. Chen. Approval of the version of the manuscript to be published: Z. Cai, J. Chen, M. Liu.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P. Bochev, J. Choi, Improved least-squares error estimates for scalar hyperbolic problems, Comput. Methods Appl. Math. 1 (2) (2001) 115–124.
[2] P. Bochev, M. Gunzburger, Least-squares methods for hyperbolic problems, in: Handbook of Numerical Analysis, vol. 17, Elsevier, 2016, pp. 289–317.
[3] P.B. Bochev, J. Choi, A comparative study of least-squares, supg and Galerkin methods for convection problems, Int. J. Comput. Fluid Dyn. 15 (2) (2001) 127–146.
[4] F. Brezzi, L.D. Marini, E. Süli, Discontinuous Galerkin methods for first-order hyperbolic problems, Math. Models Methods Appl. Sci. 14 (12) (2004) 1893–1903.
[5] E. Burman, A posteriori error estimation for interior penalty finite element approximations of the advection-reaction equation, SIAM J. Numer. Anal. 47 (5) (2009) 3584–3607.
[6] Z. Cai, J. Chen, M. Liu, Adaptive deep neural network: best LS approximation and application to PDEs, manuscript, 2021.
[7] Z. Cai, J. Chen, M. Liu, X. Liu, Deep least-squares methods: an unsupervised learning-based numerical method for solving elliptic PDEs, J. Comput. Phys. 420 (2020) 109707.
[8] G.F. Carey, B. Jianng, Least-squares finite elements for first-order hyperbolic systems, Int. J. Numer. Methods Eng. 26 (1) (1988) 81–93.
[9] W. Dahmen, C. Huang, C. Schwab, G. Welper, Adaptive Petrov–Galerkin methods for first order transport equations, SIAM J. Numer. Anal. 50 (5) (2012) 2420–2445.
[10] H. De Sterck, T.A. Manteuffel, S.F. McCormick, L. Olson, Least-squares finite element methods and algebraic multigrid solvers for linear hyperbolic pdes, SIAM J. Sci. Comput. 26 (1) (2004) 31–54.
[11] H. De Sterck, T.A. Manteuffel, S.F. McCormick, L. Olson, Numerical conservation properties of H(div)-conforming least-squares finite element methods for the Burgers equation, SIAM J. Sci. Comput. 26 (5) (2005) 1573–1597.
[12] L. Demkowicz, J. Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. Part I: The transport equation, Comput. Methods Appl. Mech. Eng. 199 (23–24) (2010) 1558–1572.
[13] D. Gottlieb, C.-W. Shu, On the Gibbs phenomenon and its resolution, SIAM Rev. 39 (4) (1997) 644–668.
[14] J. He, L. Li, J. Xu, C. Zheng, ReLU deep neural networks and linear finite elements, arXiv preprint arXiv:1807.03973, 2018.
[15] J.S. Hesthaven, Numerical Methods for Conservation Laws: From Analysis to Algorithms, SIAM, 2017.
[16] J.S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Springer Science & Business Media, 2007.
[17] P. Houston, J.A. Mackenzie, E. Süli, G. Warnecke, A posteriori error analysis for numerical approximations of Friedrichs systems, Numer. Math. 82 (3) (1999) 433–470.
[18] P. Houston, R. Rannacher, E. Süli, A posteriori error analysis for stabilised finite element approximations of transport problems, Comput. Methods Appl. Mech. Eng. 190 (11–12) (2000) 1483–1508.
[19] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Representation Learning, San Diego, 2015.
[20] R.J. LeVeque, R.J. Leveque, Numerical Methods for Conservation Laws, vol. 3, Springer, 1992.
[21] M. Liu, Z. Cai, J. Chen, Adaptive two-layer ReLU neural network, Comp. Math. Appl., submitted, 2021.
[22] Q. Liu, S. Zhang, Adaptive least-squares finite element methods for linear transport equations based on an H(div) flux reformulation, Comput. Methods Appl. Mech. Eng. 366 (2020) 113041.
[23] L. Mu, X. Ye, A simple finite element method for linear hyperbolic problems, J. Comput. Appl. Math. 330 (2018) 330–339.
[24] A. Pinkus, Approximation theory of the mlp model in neural networks, Acta Numer. 8 (1) (1999) 143–195.
[25] 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.
[26] Z. Shen, H. Yang, S. Zhang, Deep network approximation characterized by number of neurons, arXiv preprint arXiv:1906.05497, 2019.
[27] J.W. Siegel, J. Xu, Approximation rates for neural networks with general activation functions, Neural Netw. (2020).
[28] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1139–1364.