LEAST-SQUARES NEURAL NETWORK (LSNN) METHOD FOR LINEAR ADVECTION-REACTION EQUATION: GENERAL DISCONTINUOUS INTERFACE *

ZHIQIANG CAI†, JUNPYO CHOI†, AND MIN LIU‡

Abstract. We studied the least-squares ReLU neural network method (LSNN) for solving linear advection-reaction equation with discontinuous solution in [Cai, Zhiqiang, Jingshuang Chen, and Min Liu. "Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation." Journal of Computational Physics 443 (2021), 110514]. The method is based on a least-squares formulation and uses a new class of approximating functions: ReLU neural network (NN) functions. A critical and additional component of the LSNN method, differing from other NN-based methods, is the introduction of a proper designed discrete differential operator.

In this paper, we study the LSNN method for problems with arbitrary discontinuous interfaces. First, we show that ReLU NN functions with depth $\lceil \log_2(d+1) \rceil + 1$ can approximate any d-dimensional step function on arbitrary discontinuous interfaces with any prescribed accuracy. By decomposing the solution into continuous and discontinuous parts, we prove theoretically that discretization error of the LSNN method using ReLU NN functions with depth $\lceil \log_2(d+1) \rceil + 1$ is mainly determined by the continuous part of the solution provided that the solution jump is constant. Numerical results for both two and three dimensional problems with various discontinuous interfaces show that the LSNN method with enough layers is accurate and does not exhibit the common Gibbs phenomena along the discontinuous interface.

Key words. Least-Squares Method, ReLU Neural Network, Linear Advection-Reaction Equation, Discontinuous Solution

MSC codes. 65N15, 65N99

1. Introduction.

Let Ω be a bounded domain in $\mathbb{R}^d$ ($d \geq 2$) with Lipschitz boundary $\partial\Omega$. Consider the linear advection-reaction equation

$$\begin{cases} u_{\boldsymbol{\beta}} + \gamma\, u &= f, \quad \text{in } \Omega, \\ u &= g, \quad \text{on } \Gamma_-, \end{cases} \tag{1.1}$$

where $\boldsymbol{\beta}(\mathbf{x}) = (\beta_1, \cdots, \beta_d)^T \in C^0(\bar{\Omega})^d$ is a given advective velocity field, $u_{\boldsymbol{\beta}}$ denotes the directional derivative of u along $\boldsymbol{\beta}$, and Γ_- is the inflow part of the boundary $\Gamma = \partial\Omega$ given by

$$\Gamma_- = \{\mathbf{x} \in \Gamma : \boldsymbol{\beta}(\mathbf{x}) \cdot \boldsymbol{n}(\mathbf{x}) < 0\} \tag{1.2}$$

with $\boldsymbol{n}(\mathbf{x})$ the unit outward normal vector to Γ at $\mathbf{x} \in \Gamma$. Assume that $\gamma \in C^0(\bar{\Omega})$, $f \in L^2(\Omega)$, and $g \in L^2(\Gamma_-)$ are given scalar-valued functions.

When the inflow boundary data g is discontinuous, so is the solution of (1.1). The discontinuous interface may be determined by the characteristic curves emanating from where g is discontinuous. By using the location of the interface, one may design an accurate mesh-based numerical method. However, this type of methods is usually limited to linear problems and is difficult to be extended to nonlinear hyperbolic conservation laws.

In [5], we studied the least-squares ReLU neural network method (LSNN) for solving (1.1) with discontinuous solution. The method is based on the $L^2(\Omega)$ norm least-squares formulation analyzed in [12, 4] and employs a new class of approximating functions: multilayer perceptrons with the rectified linear unit (ReLU) activation function, i.e., ReLU neural network (NN) functions.

*Submitted to the editors DATE.

Funding: This work was supported in part by the National Science Foundation under grant DMS-2110571.

†Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067 (caiz@purdue.edu, choi508@purdue.edu).

‡School of Mechanical Engineering, Purdue University, 585 Purdue Mall, West Lafayette, IN 47907-2088(liu66@purdue.edu).

A critical and additional component of the LSNN method, differing from other NN-based methods, is the introduction of a proper designed discrete differential operator.

One of the appealing features of the LSNN method is its ability of automatically approximating the discontinuous solution without using *a priori* knowledge of the location of the interface. Hence, the method is applicable to nonlinear problems (see [8, 7]). Compared to mesh-based numerical methods including various adaptive mesh refinement (AMR) algorithms that locate the discontinuous interface through local mesh refinement (see, e.g., [11, 18, 25]), the LSNN method, a meshfree and pointfree method, is much more effective in terms of the number of the degrees of freedom. Theoretically, it was shown in [5] that a two- or three-layer ReLU NN function in two dimensions is sufficient to well approximate the discontinuous solution of (1.1) without oscillation, provided that the interface consists of a straight line or two-line segments and that the solution jump along the interface is constant.

The assumption on at most two-line segments in [5] is very restrictive even in two dimensions. In general, the discontinuous solution of (1.1) has interfaces that are hyper-surfaces in d dimensions. The purpose of this paper has two-fold. First, we show that any step function with general interface may be approximated by ReLU NNs with at most $\lceil \log_2(d+1) \rceil + 1$ layers for achieving a given approximation accuracy ε (see Lemma 4.3), which extends the approximation result in [5]. This is done by constructing a continuous piecewise linear (CPWL) function with a sharp transition layer of ε width and combining the main results in [32, 33] (see Proposition 2.1). Question on approximating piece-wise smooth functions by ReLU NNs are also arised in data science applications such as classification, etc. Some convergence rates were obtained in [26, 10, 20, 19, 21]. Particularly, for a given C^β ($\beta > 1$) interface, [26] established approximation rates of ReLU NNs with no more than $(3 + \lceil \log_2 \beta \rceil)(11 + 2\beta/d)$ layers.

Second, we establish a new kind of *a priori* error estimates (see Theorem 4.4) for the LSNN method in d dimensions for general discontinuous interface. To do so, we decompose the solution as the sum of the discontinuous and continuous parts (see (4.3)). The continuous part of the solution may be approximated well by (even shallow) ReLU NN functions with standard approximation property (see, e.g., [14, 27, 28, 30, 13]). The discontinuous part of the solution can be approximated accurately by the class of all ReLU NN functions from $\mathbb{R}^d$ to $\mathbb{R}$ with at most $\lceil \log_2(d+1) \rceil + 1$ depth, provided that the solution jump is constant. Hence, the accuracy of the LSNN method is mainly determined by the continuous part of the solution.

The explicit construction in this paper indicates that a ReLU NN function with at most $\lceil \log_2(d+1) \rceil + 1$ depth is sufficient to accurately approximate discontinuous solutions without oscillation. The necessary depth $\lceil \log_2(d+1) \rceil + 1$ of a ReLU NN function is shown numerically through several test problems in both two and three dimensions (two hidden layers for $d = 2, 3$). At the current stage, it is still very expensive to numerically solve the discrete least-squares minimization problem when using the ADAM optimization algorithm [22], even though the DoFs of the LSNN method is much less than mesh-based numerical methods.

Followed by recent success of DNNs in machine learning and artificial intelligence tasks such as computer vision and pattern recognition, there have been active interests in using DNNs for solving partial differential equations (PDEs) (see, e.g., [2, 3, 9, 16, 29, 31]). Due to the fact that the collection of DNN functions is not a space, NN-based methods for solving PDEs may be categorized as the Ritz and least-squares (LS) methods. The former (see, e.g., [16]) requires the underlying problem having a natural minimization principle and hence is not applicable to (1.1).

For a given PDE, there are many least-squares methods and their efficacy depends on norms used for PDE and for its boundary and/or initial conditions. When using NNs as approximating functions, least-squares methods may be traced back at least to 1990s (see, e.g., [15, 23]), where the discrete l^2 norm on a uniform integration mesh were employed for both PDEs of the strong form and for their boundary/initial conditions. Along this line, it is the popular physics-informed neural networks (PINNs) by Raissi-Perdikaris-Karniadakis [29] in 2019 which uses auto-differentiation for

computing the underlying differential operator at each integration point. Since the solution of (1.1) is discontinuous, those NN-based least-squares methods are also not applicable.

The rest of the paper is organized as follows. In Section 2, we describe ReLU NN functions and CPWL functions, and introduce a known result about their relationship. Then we further investigate the structure of ReLU NN functions. Section 3 review the least-squares neural network method and formulate the method on the framework in Section 2. Then we prove the method is capable of locating any discontinuous interfaces of the problem in Section 4. Finally, Section 5 presents numerical results for both two and three dimensional test problems with various discontinuous interfaces.

2. ReLU NN Functions. First we begin with the definition of the rectified linear unit (ReLU) activation function. The ReLU activation function σ is defined by

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

We say a function $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}^c$ with $c, d \in \mathbb{N}$ is a ReLU neural network (NN) function if the function $\mathcal{N}$ has a representation:

$$\mathcal{N} = N^{(L)} \circ \cdots \circ N^{(2)} \circ N^{(1)} \text{ with } L > 1, \tag{2.1}$$

where the symbol $\circ$ denotes the composition of functions, and for each $l = 1, \ldots, L$, $N^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ with $n_l, n_{l-1} \in \mathbb{N}$ ($n_0 = d$, $n_L = c$) given by:

1. For $l = L$, $N^{(L)}(\mathbf{x}) = \boldsymbol{\omega}^{(L)}\mathbf{x} - \mathbf{b}^{(L)}$ for all $\mathbf{x} \in \mathbb{R}^{n_{L-1}}$ for $\boldsymbol{\omega}^{(L)} \in \mathbb{R}^{n_L \times n_{L-1}}$, $\mathbf{b}^{(L)} \in \mathbb{R}^{n_L}$.
2. For each $l = 1, \ldots, L-1$, $N^{(l)}(\mathbf{x}) = \sigma\left(\boldsymbol{\omega}^{(l)}\mathbf{x} - \mathbf{b}^{(l)}\right)$ for all $\mathbf{x} \in \mathbb{R}^{n_{l-1}}$ for $\boldsymbol{\omega}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$, where σ is applied to each component.

We now establish some terminology as follows. Let a ReLU NN function $\mathcal{N}$ have a representation $N^{(L)} \circ \cdots \circ N^{(2)} \circ N^{(1)}$ with $L > 1$ as in (2.1). Then we say:

1. The representation has depth L.
2. The representation is said to have L layers and $L-1$ hidden layers.
3. For each $l = 1, \ldots, L$, the entries of $\boldsymbol{\omega}^{(l)}$ and $\mathbf{b}^{(l)}$ are called weights and biases, respectively.
4. For each $l = 1, \ldots, L$, the natural number n_l is called the width or the number of neurons of the l^{th} layer.

A motivation for this terminology is illustrated in Figure 1.

For a given positive integer n, denote the set of all ReLU NN functions from $\mathbb{R}^d$ to $\mathbb{R}$ that have a representation with depth L and the total number of neurons in the hidden layers n by

$$\mathcal{M}(d, 1, L, n) = \left\{ \mathcal{N} : \mathbb{R}^d \to \mathbb{R} : \mathcal{N} = N^{(L)} \circ \cdots \circ N^{(2)} \circ N^{(1)} \text{ defined in (2.1)} : n = \sum_{l=1}^{L-1} n_l \right\}.$$

Denote the set of all ReLU NN functions from $\mathbb{R}^d$ to $\mathbb{R}$ with a L-layer representation by $\mathcal{M}(d, 1, L)$. Then

$$\mathcal{M}(d, 1, L) = \bigcup_{n \in \mathbb{N}} \mathcal{M}(d, 1, L, n). \tag{2.2}$$

We now introduce another function class, which is the set of continuous piecewise linear functions, and explore a theorem about the relationship between the two function classes. We say a function $f : \mathbb{R}^d \to \mathbb{R}$ with $d \in \mathbb{N}$ is continuous piecewise linear (CPWL) if there exists a finite set of polyhedra with nonempty interior such that:

1. The interiors of any two polyhedra in the set are disjoint.
2. The union of the set is $\mathbb{R}^d$.

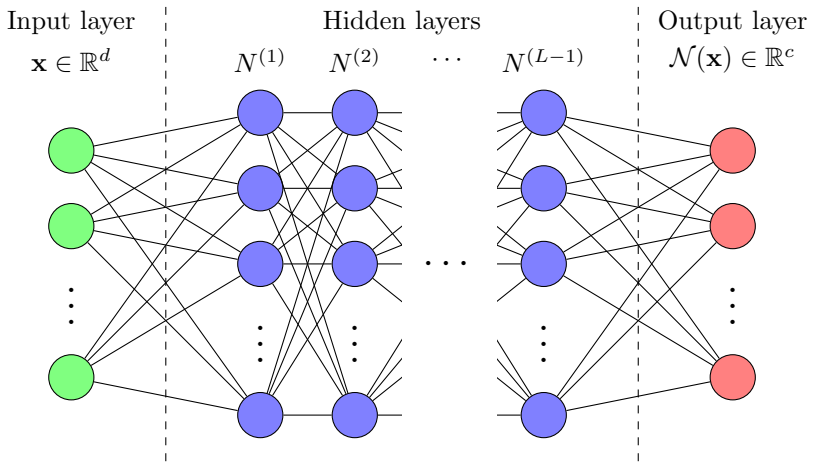


FIG. 1. *ReLU neural network function structure*

3. f is affine linear over each polyhedron in the set, i.e., in each polyhedron in the set, $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$ for $\mathbf{a} \in \mathbb{R}^d$, $b \in \mathbb{R}$.

Here by a polyhedron, we mean a subset of $\mathbb{R}^d$ surrounded by a finite number of hyperplanes, i.e., the solution set of a system of linear inequalities

$$\{\mathbf{x} \in \mathbb{R}^d : \mathbf{A}\mathbf{x} \le \mathbf{b}\} \text{ for } \mathbf{A} \in \mathbb{R}^{m\times d}, \mathbf{b} \in \mathbb{R}^m \text{ with } m \in \mathbb{N}, \tag{2.3}$$

where the inequality is applied to each component. Thus the interior of the polyhedron in (2.3) is

$$\{\mathbf{x} \in \mathbb{R}^d : \mathbf{A}\mathbf{x} < \mathbf{b}\}.$$

PROPOSITION 2.1. *The set of all* CPWL *functions* $f\colon \mathbb{R}^d \to \mathbb{R}$ *is equal to* $\mathcal{M}(d, 1, \lceil \log_2(d+1)\rceil + 1)$*, i.e., the set of all* ReLU NN functions *from* $\mathbb{R}^d$ *to* $\mathbb{R}$ *that have a representation with depth* $\lceil \log_2(d+1)\rceil + 1$.

Proof. $\mathcal{M}(d, 1, \lceil \log_2(d+1)\rceil + 1)$ is clearly a subset of the set of CPWL functions. Conversely, it was proved in [1] that every CPWL function is a ReLU NN function from $\mathbb{R}^d$ to $\mathbb{R}$ that has a representation with depth at most $\lceil \log_2(d+1)\rceil + 1$. Now, the result follows from the fact that $\mathcal{M}(d, 1, L) \subset \mathcal{M}(d, 1, \lceil \log_2(d+1)\rceil + 1)$ for any $L \le \lceil \log_2(d+1)\rceil + 1$. ∎

Proposition 2.1 enables us to employ ReLU NN functions with a few layer representation to the problems where CPWL functions are used, and to only control the number of neurons. Except the case $d = 1$ (see, e.g., [1]), there are currently no known results to give tight bounds on the number of neurons in the hidden layers. Therefore we suggest the following approach. The following proposition is a trivial fact.

PROPOSITION 2.2. $\mathcal{M}(d, 1, L, n) \subset \mathcal{M}(d, 1, L, n+1)$.

Now Proposition 2.1 (2.2), and Proposition 2.2 suggest how we control the number of neurons in the hidden layers, i.e., when approximating a function $\mathbb{R}^d \to \mathbb{R}$ by a CPWL function, we start with a class $\mathcal{M}(d, 1, \lceil \log_2(d+1)\rceil + 1, n)$ with a small n, and then increase n to have a better approximation.

3. LSNN Method. Define the least-squares (LS) functional as

$$\mathcal{L}(v; \mathbf{f}) = \|v_{\boldsymbol{\beta}} + \gamma\, v - f\|^2_{0,\Omega} + \|v - g\|^2_{-\boldsymbol{\beta}}, \tag{3.1}$$

where $\mathbf{f} = (f, g)$ and $\|\cdot\|_{-\boldsymbol{\beta}}$ denotes the weighted $L^2(\Gamma_-)$ norm over the inflow boundary given by

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

Let $V_{\boldsymbol{\beta}} = \{v \in L^2(\Omega) : v_{\boldsymbol{\beta}} \in L^2(\Omega)\}$ that is equipped with the norm as

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

The least-squares formulation of problem (1.1) is to seek $u \in V_{\boldsymbol{\beta}}$ such that

$$\mathcal{L}(u; \mathbf{f}) = \min_{v\in V_{\boldsymbol{\beta}}} \mathcal{L}(v; \mathbf{f}). \tag{3.2}$$

PROPOSITION 3.1 (see [4, 12]). *Assume that either $\gamma = 0$ or there exists a positive constant γ_0 such that*

$$\gamma(\mathbf{x}) - \frac{1}{2}\nabla\cdot\boldsymbol{\beta}(\mathbf{x}) \geq \gamma_0 > 0 \quad \text{for almost all } \mathbf{x} \in \Omega. \tag{3.3}$$

Then 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*

$$\alpha\, |||v|||_{\boldsymbol{\beta}}^2 \leq \mathcal{L}(v; \mathbf{0}) \leq M\, |||v|||_{\boldsymbol{\beta}}^2 \quad \text{for all } v \in V_{\boldsymbol{\beta}}. \tag{3.4}$$

PROPOSITION 3.2 (see [4, 12]). *Problem (3.2) has a unique solution $u \in V_{\boldsymbol{\beta}}$ satisfying the following a priori estimate*

$$|||u|||_{\boldsymbol{\beta}} \leq C\, \left(\|f\|_{0,\Omega} + \|g\|_{-\boldsymbol{\beta}}\right). \tag{3.5}$$

Note that $\mathcal{M}(d, 1, L, n)$ is a subset of $V_{\boldsymbol{\beta}}$. The least-squares approximation is to find $u_{_N} \in \mathcal{M}(d, 1, L, n)$ such that

$$\mathcal{L}\big(u_{_N}; \mathbf{f}\big) = \min_{v\in\mathcal{M}(d,1,L,n)} \mathcal{L}\big(v; \mathbf{f}\big). \tag{3.6}$$

LEMMA 3.3 (see [5]). *Let u and $u_{_N}$ be the solutions of problems (3.2) and (3.6), respectively. Then we have*

$$|||u - u_{_N}|||_{\boldsymbol{\beta}} \leq \left(\frac{M}{\alpha}\right)^{1/2} \inf_{v\in\mathcal{M}(d,1,L,n)} |||u - v|||_{\boldsymbol{\beta}}, \tag{3.7}$$

where α and M are constants in (3.4).

We use numerical integration (the midpoint rule) to implement the scheme in (3.6) (see [5]). Denote the discrete LS functional by $\mathcal{L}_{\mathcal{T}}\big(v; \mathbf{f}\big)$ for $v \in V_\beta$. Then the discrete least-squares approximation of problem (1.1) is to find $u_{\mathcal{T}}^N \in \mathcal{M}(d, 1, L, n)$ such that

$$\mathcal{L}_{\mathcal{T}}\big(u_{\mathcal{T}}^N; \mathbf{f}\big) = \min_{v\in\mathcal{M}(d,1,L,n)} \mathcal{L}_{\mathcal{T}}\big(v; \mathbf{f}\big). \tag{3.8}$$

4. Error Estimate. In this section, we provide error estimates of the ReLU NN function approximation to the solution of the linear advection-reaction equation with an arbitrary discontinuous interface. To this end, we note first that the solution of the problem is discontinuous if the inflow boundary data g is discontinuous.

LEMMA 4.1. *For $d = 2$, assume that the inflow boundary data g is discontinuous at $\mathbf{x}_0 \in \Gamma_-$ with values $g^+(\mathbf{x}_0)$ and $g^-(\mathbf{x}_0)$ from different sides. Let I be the streamline of the vector field $\boldsymbol{\beta}$ emanating from $\mathbf{x}_0$ and $\mathbf{x}(s)$ be a parameterization of I, i.e.,*

$$\frac{d\mathbf{x}(s)}{ds} = \boldsymbol{\beta}(\mathbf{x}(s)), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{4.1}$$

Then the solution u of (1.1) is discontinuous on I with jump described as

$$|u^+(\mathbf{x}(s)) - u^-(\mathbf{x}(s))| = \exp\left(-\int_0^s \gamma(\mathbf{x}(t))\,dt\right)\left|\int_0^s \exp\left(\int_0^t \gamma(\mathbf{x}(r))\,dr\right)(f^+(\mathbf{x}(t)) - f^-(\mathbf{x}(t)))\,dt + g^+(\mathbf{x}_0) - g^-(\mathbf{x}_0)\right|,$$

where $u^+(\mathbf{x}(s))$ and $u^-(\mathbf{x}(s))$ are the solutions, and $f^+(\mathbf{x}(t))$ and $f^-(\mathbf{x}(t))$ are the values of f of (1.1) along I from different sides, respectively.

Proof. Along the interface I, by the definition of the directional derivative, we have

$$u_{\boldsymbol{\beta}}(\mathbf{x}(s)) = \frac{d}{ds}u(\mathbf{x}(s)).$$

Thus the solutions $u^{\pm}(\mathbf{x}(s))$ along the interface I satisfies the linear ordinary differential equations

$$\begin{cases} \frac{d}{ds}u^{\pm}(\mathbf{x}(s)) + \gamma(\mathbf{x}(s))u^{\pm}(\mathbf{x}(s)) = f^{\pm}(\mathbf{x}(s)), & \text{for } s > 0, \\ u^{\pm}(\mathbf{x}(0)) = u^{\pm}(\mathbf{x}_0) = g^{\pm}(\mathbf{x}_0) \end{cases} \tag{4.2}$$

whose solutions are given by

$$u^{\pm}(\mathbf{x}(s)) = \exp\left(-\int_0^s \gamma(\mathbf{x}(t))\,dt\right)\left[\int_0^s \exp\left(\int_0^t \gamma(\mathbf{x}(r))\,dr\right) f^{\pm}(\mathbf{x}(t))\,dt + g^{\pm}(\mathbf{x}_0)\right].$$

Hence, u is discontinuous on I with jump

$$|u^+(\mathbf{x}(s)) - u^-(\mathbf{x}(s))| = \exp\left(-\int_0^s \gamma(\mathbf{x}(t))\,dt\right)\left|\int_0^s \exp\left(\int_0^t \gamma(\mathbf{x}(r))\,dr\right)(f^+(\mathbf{x}(t)) - f^-(\mathbf{x}(t)))\,dt + g^+(\mathbf{x}_0) - g^-(\mathbf{x}_0)\right|.$$

This completes the proof of the lemma. ∎

Remark 4.2. For $d = 3$, assume that the inflow boundary data g is discontinuous along a curve $C(t) \subset \Gamma_-$. Note that the collection of the stream lines $\mathbf{x}(s)$ of the vector field $\boldsymbol{\beta}$ starting at all $\mathbf{x}_0 = C(t)$ forms a surface $I(s,t)$. Then the solution u of (1.1) is discontinuous on the surface $I(s,t)$ with jump as in Lemma 4.1 for every $\mathbf{x}_0 = C(t)$.

Let the discontinuous interface I (in $\mathbb{R}^d$) divide the domain Ω into two nonempty subdomains Ω_1 and Ω_2:

$$\Omega = \Omega_1 \cup \Omega_2 \quad \text{and} \quad I = \partial\Omega_1 \cap \partial\Omega_2,$$

so that u is piecewise smooth with respect to the partition $\{\Omega_1, \Omega_2\}$.

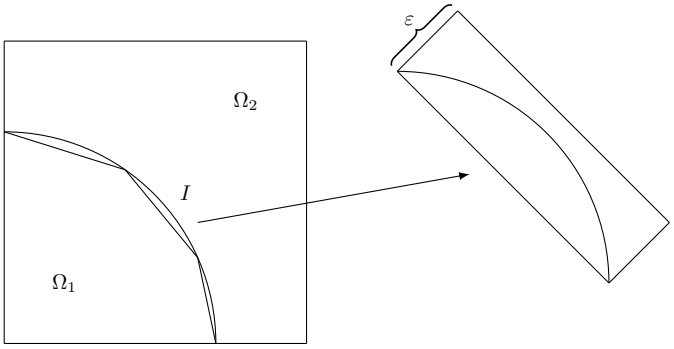
Assume that the jump of the solution is constant. Hence, u can be decomposed into

$$u(\mathbf{x}) = \hat{u}(\mathbf{x}) + \chi(\mathbf{x}), \tag{4.3}$$

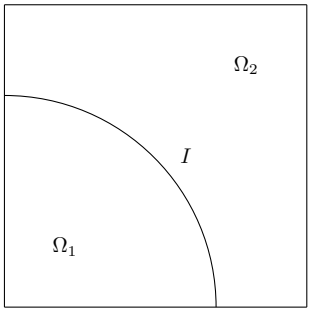
where $\hat{u}$ is continuous and piecewise smooth on Ω, and $\chi(\mathbf{x})$ is a piecewise constant function defined by

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

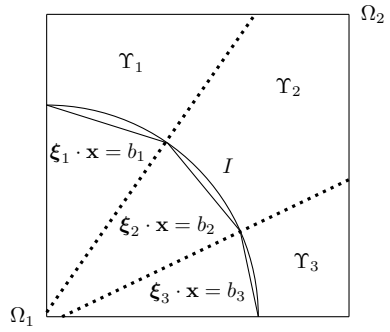
with $\alpha_1 = g^-(\mathbf{x}_0)$ and $\alpha_2 = g^+(\mathbf{x}_0)$ (see Figure 2(b)). By a rotation of the coordinates $\mathbf{x}$, represent the interface I by $s = R(\mathbf{y})$. Next for any $\varepsilon > 0$, approximate R by a CPWL function $s = c(\mathbf{y})$ with linear functions $c_i(\mathbf{y})$ for $i = 1, \ldots, k$ such that the vertical distance from c_i to the corresponding portion of I is less than ε (see Figure 2(a)). Each linear function $c_i(\mathbf{y})$ of R is a hyperplane $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$. By normalizing $\boldsymbol{\xi}_i$, we may assume $|\boldsymbol{\xi}_i| = 1$. Now divide Ω by hyperplanes passing through the intersections of $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$. Let Υ_i be the subdomains determined by this process (see Figure 2(c)).



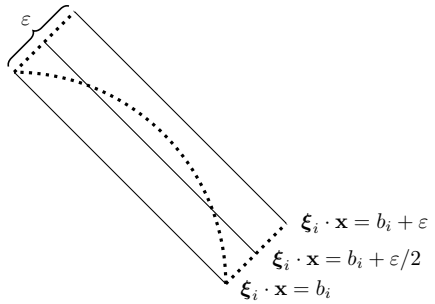
(a) Approximation along the interface I by a CPWL function



(b) Interface I on $\Omega \subset \mathbb{R}^d$



(c) Partition of Ω through the intersections of $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$



(d) Hyperplanes $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$, $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i + \varepsilon$ for $p_i(\mathbf{x})$

FIG. 2. *Lemma* 4.3 *illustruations*

LEMMA 4.3. *Without loss of generality, let $p(\mathbf{x})$ be a CPWL function given by (see Figure 2(d))*

$$p_i(\mathbf{x}) = \alpha_1 + \frac{\alpha_2 - \alpha_1}{\varepsilon}\Big(\sigma(\boldsymbol{\xi}_i \cdot \mathbf{x} - b_i) - \sigma(\boldsymbol{\xi}_i \cdot \mathbf{x} - b_i - \varepsilon)\Big),\ \mathbf{x} \in \Upsilon_i.$$

Then we have

$$|||\chi - p|||_{\boldsymbol{\beta}} \le \sqrt{2|I|}\,\big|\alpha_1 - \alpha_2\big|\,\sqrt{\varepsilon}, \tag{4.4}$$

where $|I|$ is the $d-1$ dimensional measure of the interface I.

Proof. For each i, denote by $|\mathcal{P}_i|$ the $d-1$ dimensional measure of the hyperplane $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$ in Υ_i. It is easy to check that

$$\|\chi - p_i\|^2_{0,\Upsilon_i} \le (\alpha_1 - \alpha_2)^2 |\mathcal{P}_i|\varepsilon,$$

which implies

$$\|\chi - p\|^2_{0,\Omega} = \sum_{i=1}^{k} \|\chi - p_i\|^2_{0,\Upsilon_i} \le \sum_{i=1}^{k} |\mathcal{P}_i|(\alpha_1 - \alpha_2)^2\varepsilon \le |I|\,(\alpha_1 - \alpha_2)^2\,\varepsilon. \tag{4.5}$$

We now prove

$$\|\chi_{\boldsymbol{\beta}} - p_{\boldsymbol{\beta}}\|^2_{0,\Omega} \le |I|\,(\alpha_1 - \alpha_2)^2\,\varepsilon.$$

To do so, for each i, let

$$\Upsilon_i^1 = \{\mathbf{x} \in \Upsilon_i : 0 < \boldsymbol{\xi} \cdot \mathbf{x} - b_i < \varepsilon\} \quad \text{and} \quad \Upsilon_i^2 = \Upsilon_i \setminus \Upsilon_i^1.$$

Clearly, $\chi_{\boldsymbol{\beta}} \equiv 0$ in Ω, and p_i is piecewise constant in Υ_i^2. For each Υ_i, we can a construct vector-field $\boldsymbol{\beta}_i(\mathbf{x})$ in Υ_i such that for each $\mathbf{x} \in \Upsilon_i^1$, $\boldsymbol{\beta}_i(\mathbf{x})$ is parallel to the hyperplane $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$ in Υ_i and that $\boldsymbol{\beta}(\mathbf{x}) - \boldsymbol{\beta}_i(\mathbf{x})$ is parallel to $\boldsymbol{\xi}_i$ in Υ_i^1. Hence, $(p_i)_{\boldsymbol{\beta}_i} \equiv 0$ in Υ_i. Then

$$\|(p_i)_{\boldsymbol{\beta}}\|^2_{0,\Upsilon_i} = \|(p_i)_{\boldsymbol{\beta}} - (p_i)_{\boldsymbol{\beta}_i}\|^2_{0,\Upsilon_i} = \|(p_i)_{\boldsymbol{\beta}-\boldsymbol{\beta}_i}\|^2_{0,\Upsilon_i} = \|(p_i)_{\boldsymbol{\beta}-\boldsymbol{\beta}_i}\|^2_{0,\Upsilon_i^1}$$

$$\le \int_{\Upsilon_i^1} \left(\frac{\alpha_2 - \alpha_1}{\varepsilon}\boldsymbol{\xi}_i \cdot \varepsilon\boldsymbol{\xi}_i\right)^2 d\mathbf{x} \le (\alpha_1 - \alpha_2)^2 |\mathcal{P}_i|\,\varepsilon,$$

where for the second inequality, we used the fact that in Υ_i^1, the gradient of p_i is $\frac{\alpha_2 - \alpha_1}{\varepsilon}\boldsymbol{\xi}_i$ and further assume the magnitude of $\boldsymbol{\beta}(\mathbf{x}) - \boldsymbol{\beta}_i(\mathbf{x})$ is less than or equal to $\varepsilon\boldsymbol{\xi}_i$. Thus

$$\|\chi_{\boldsymbol{\beta}} - p_{\boldsymbol{\beta}}\|^2_{0,\Omega} = \sum_{i=1}^{k} \|(p_i)_{\boldsymbol{\beta}}\|^2_{0,\Upsilon_i} \le \sum_{i=1}^{k} |\mathcal{P}_i|(\alpha_1 - \alpha_2)^2\varepsilon \le |I|\,(\alpha_1 - \alpha_2)^2\,\varepsilon. \tag{4.6}$$

Now (4.4) follows from (4.5) and (4.6). ∎

THEOREM 4.4. *Let u and $u_{_N}$ be the solutions of problems (3.2) and (3.6), respectively. Then we have*

$$|||u - u_{_N}|||_{\boldsymbol{\beta}} \le C\left(\big|\alpha_1 - \alpha_2\big|\,\sqrt{\varepsilon} + \inf_{v \in \mathcal{M}(d,1,L,n)} |||\hat{u} + p - v|||_{\boldsymbol{\beta}}\right), \tag{4.7}$$

where $\hat{u} \in C(\Omega)$ and p are given in (4.3) and Lemma 4.3, respectively. Moreover, if the depth of DNNs in (3.6) is at least $\lceil \log_2(d+1) \rceil + 1$, then for a sufficiently large integer n, there exists an integer $\hat{n} \le n$ such that

$$|||u - u_{_N}|||_{\boldsymbol{\beta}} \le C\left(\big|\alpha_1 - \alpha_2\big|\,\sqrt{\varepsilon} + \inf_{v \in \mathcal{M}(d,n-\hat{n})} |||\hat{u} - v|||_{\boldsymbol{\beta}}\right), \tag{4.8}$$

where $\mathcal{M}(d, n-\hat{n}) = \mathcal{M}(d, 1, \lceil \log_2(d+1) \rceil + 1, n - \hat{n})$.

Proof. For any $v \in \mathcal{M}(d,1,L,n)$, it follows from (4.3), the triangle inequality, and Lemma 4.3 that

$$\begin{aligned}|||u-v|||_{\boldsymbol{\beta}} &= |||\chi - p + \hat{u} + p - v|||_{\boldsymbol{\beta}} \le |||\chi - p|||_{\boldsymbol{\beta}} + |||\hat{u}+p-v|||_{\boldsymbol{\beta}} \\ &\le \sqrt{2|I|}|\alpha_1-\alpha_2|\sqrt{\varepsilon} + |||\hat{u}+p-v|||_{\boldsymbol{\beta}}.\end{aligned}$$

Taking the infimum over all $v \in \mathcal{M}(d,1,L,n)$, (4.7) is then a direct consequence of Lemma 3.3.

To show the last statement, first, note that for a sufficiently large integer n, by Proposition 2.1, there exists an integer $\hat{n} \le n$ such that

$$p \in \mathcal{M}(d,\hat{n}) = \mathcal{M}(d,1,\lceil \log_2(d+1)\rceil + 1, \hat{n}).$$

Obviously we have $v + p \in \mathcal{M}(d,n)$ for any $v \in \mathcal{M}(d,n-\hat{n})$. Now, it follows from the coercivity and continuity of the homogeneous functional $\mathcal{L}\big(v;\mathbf{0}\big)$ in (3.4), problems (1.1), (3.6), (4.3), and the triangle inequality that

$$\begin{aligned}\alpha\,|||u-u_{_N}|||_{\boldsymbol{\beta}}^2 &\le \mathcal{L}\big(u-u_{_N};\mathbf{0}\big) = \mathcal{L}\big(u_{_N};\mathbf{f}\big) \le \mathcal{L}\big(v+p;\mathbf{f}\big) = \mathcal{L}\big(u-v-p;\mathbf{0}\big) \\ &= \mathcal{L}\big((\hat{u}-v)+(\chi-p);\mathbf{0}\big) \le M\,|||(\hat{u}-v)+(\chi-p)|||_{\boldsymbol{\beta}}^2 \\ &\le 2M\left(|||(\hat{u}-v)|||_{\boldsymbol{\beta}}^2 + |||(\chi-p)|||_{\boldsymbol{\beta}}^2\right),\end{aligned}$$

which, together with Lemma 4.3, implies the validity of (4.8). This completes the proof of the theorem. ∎

Remark 4.5. When the continuous part of the solution $\hat{u}$ is smooth, it is known that $\hat{u}$ can be approximated well by a shallow NN, i.e., with depth $L = 2$ (see, e.g., [14, 27, 28, 30, 13, 17]).

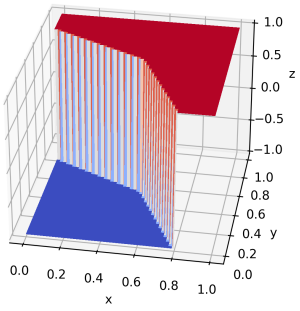
Remark 4.6. The estimate in (4.7) holds even for the shallow neural network ($L = 2$). However, the second term of the upper bound, $\inf_{v\in\mathcal{M}(d,1,2,n)} |||\hat{u}+p-v|||_{\boldsymbol{\beta}}$, depends on the inverse of ε because the p has a sharp transition layer of width ε. For any fixed n, $\inf_{v\in\mathcal{M}(d,1,2,n)} \|p-v\|_{0,\Omega}$ could be large depending on the size of ε, even though the universal approximation theorem (see, e.g., [28]) implies

$$\lim_{n\to\infty} \inf_{v\in\mathcal{M}(d,1,2,n)} \|p-v\|_{0,\Omega} = 0.$$

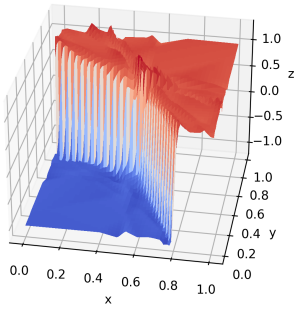
In other words, as ε approaches 0, p approaches χ, which is discontinuous; hence, in practice, the universal approximation theorem does not guarantee the convergence of the problem. On the other hand, by Proposition 2.1, deeper networks (with depth at least $\lceil \log_2(d+1)\rceil + 1$) are capable of approximating the solution. As an illustration, define a CPWL function $p(x,y)$ with a sharp transition layer of width ε in $[0,1]^2$ as follows.

$$p(x,y) = \begin{cases} -1 + \frac{2}{\varepsilon}\left(y + \frac{1}{2}x - 0.8 + \frac{\varepsilon}{2}\right), & y \ge x, \\ -1 + \frac{2}{\varepsilon}\left(\frac{1}{2}y + x - 0.8 + \frac{\varepsilon}{2}\right), & y < x. \end{cases}$$

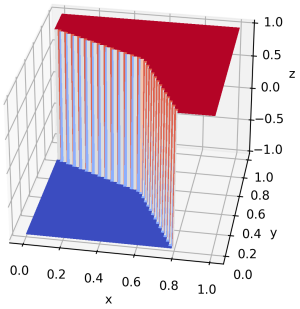
In [5], we proved CPWL functions of the form $p(x,y)$ is a ReLU NN function with the 2-4-4-1 structure. Here and in what follows, by a 2-n_1-n_2-1 structure, we mean a ReLU NN function with depth 3 and n_1, n_2 neurons in the respective hidden layers. Therefore, we can expect the 2-n_1-n_2-1 structure outperforms the 2-n_3-1 structure (ReLU NN functions with depth 2 and n_3 neurons in the hidden layer), and Figure 3 and Table 1 demonstrate this for approximating $p(x,y)$ with $\varepsilon = 0.2, 0.02, 0.002$ by ReLU NN functions with depth 2, 3 and various numbers of nuerons.



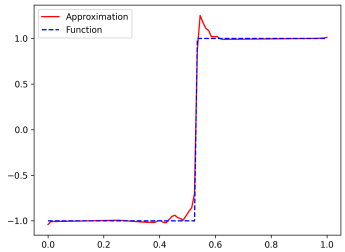
(a) $p(x, y)$ with $\varepsilon = 0.002$



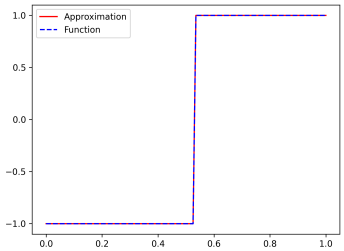
(b) ReLU NN with the 2-158-1 structure



(c) ReLU NN with the 2-4-4-1 structure



(d) Trace of 3(b) on $y = x$



(e) Trace of 3(c) on $y = x$

FIG. 3. *L^2 norm approximation of $p(x, y)$ with $\varepsilon = 0.002$*

LEMMA 4.7. *Let u, $u_{\scriptscriptstyle N}$, and $u_{\mathcal{T}}^N$ be the solutions of problems* (3.2), (3.6), *and* (3.8), *respectively. Then there exist positive constants C_1 and C_2 such that*

$$\begin{aligned} \left|\!\left|\!\left| u - u_{\mathcal{T}}^N \right|\!\right|\!\right|_{\boldsymbol{\beta}} \le C_1 \left(\left| (\mathcal{L} - \mathcal{L}_{\mathcal{T}})(u_{\scriptscriptstyle N} - u_{\mathcal{T}}^N, \mathbf{0}) \right| + \left| (\mathcal{L} - \mathcal{L}_{\mathcal{T}})(u - u_{\scriptscriptstyle N}, \mathbf{0}) \right| \right)^{1/2} \\ + C_2 \left(\left| \alpha_1 - \alpha_2 \right| \sqrt{\varepsilon} + \inf_{v \in \mathcal{M}(d,n)} \left|\!\left|\!\left| \hat{u} + p - v \right|\!\right|\!\right|_{\boldsymbol{\beta}} \right). \end{aligned} \tag{4.9}$$

Proof. By the triangle inequality

$$\left|\!\left|\!\left| u - u_{\mathcal{T}}^N \right|\!\right|\!\right|_{\boldsymbol{\beta}} \le \left|\!\left|\!\left| u - u_{\scriptscriptstyle N} \right|\!\right|\!\right|_{\boldsymbol{\beta}} + \left|\!\left|\!\left| u_{\scriptscriptstyle N} - u_{\mathcal{T}}^N \right|\!\right|\!\right|_{\boldsymbol{\beta}},$$

TABLE 1
Relative errors in the L^2 norm for approximating $p(x, y)$ with $\varepsilon = 0.2, 0.02, 0.002$ by ReLU NN functions with depth 2, 3 and various numbers of neurons

	2-8-1	2-58-1	2-108-1	2-158-1	2-4-4-1
$\varepsilon = 0.2$	0.292446	0.028176	0.021200	0.010259	1.906427×10^{-7}
$\varepsilon = 0.02$	0.254603	0.078549	0.065465	0.030623	7.536140×10^{-6}
$\varepsilon = 0.002$	0.404299	0.102757	0.100136	0.088885	9.473783×10^{-7}

and the fact that

$$\left|\!\left|\!\left| u_{\scriptscriptstyle N} - u_{\mathcal{T}}^{N} \right|\!\right|\!\right|_{\boldsymbol{\beta}} \le C_1 \left(\left(\left| (\mathcal{L} - \mathcal{L}_{\mathcal{T}})(u_{\scriptscriptstyle N} - u_{\mathcal{T}}^{N}, \mathbf{0}) \right| + \left| (\mathcal{L} - \mathcal{L}_{\mathcal{T}})(u - u_{\scriptscriptstyle N}, \mathbf{0}) \right| \right)^{1/2} + \left|\!\left|\!\left| u - u_{\scriptscriptstyle N} \right|\!\right|\!\right|_{\boldsymbol{\beta}} \right)$$

from the proof of Lemma 3.4 in [5], (4.9) is a direct consequence of Theorem 4.4. ∎

5. Numerical Experiments. In this section, we report numerical results for both two and three dimensional test problems with piecewise constant, or variable advection velocity fields. Numerical integration used the midpoint rule on a uniform mesh with mesh size $h = 10^{-2}$. The directional derivative $v_{\boldsymbol{\beta}}$ in the $\boldsymbol{\beta}$ direction was approximated by the backward finite difference quotient multiplied by $|\boldsymbol{\beta}|$

$$v_{\boldsymbol{\beta}}(\mathbf{x}_{\scriptscriptstyle K}) \approx |\boldsymbol{\beta}| \frac{v(\mathbf{x}_{\scriptscriptstyle K}) - v\big(\mathbf{x}_{\scriptscriptstyle K} - \rho \bar{\boldsymbol{\beta}}(\mathbf{x}_{\scriptscriptstyle K})\big)}{\rho}, \tag{5.1}$$

where $\bar{\boldsymbol{\beta}} = \frac{\boldsymbol{\beta}}{|\boldsymbol{\beta}|}$ and $\rho = h/4$. We used the ADAM optimization algorithm [22] to iteratively solve the discrete minimization problem in (3.8). For each numerical experiment, the learning rate started with 0.004 and was reduced by half for every 50000 iterations. Due to the possibility of the neural network getting trapped in a local minimum, we first trained the network with 5000 iterations 10 times, chose the weights and biases with the minimum loss function value, and trained further to get the results.

In Tables 2 to 7, parameters indicate the total number of weights and biases. We used ReLU NN function with width n and depth $3 = \lceil \log_2(d+1) \rceil + 1$ for $d = 2, 3$, the structure of NN functions is denoted by d-n-n-1, where the 1 is the dimension of the output. The basic principle for choosing the number of neurons is to start with a small number and increase the number to obtain a better approximation. For an automatic approach to design the architecture of DNNs for a given problem with a prescribed accuracy, see the recent work on the adaptive neural network method in [24, 6]. In Figures 4 to 9, by the l^{th} layer breaking lines, we mean the set

$$\{\mathbf{x} \in \Omega : \boldsymbol{\omega}^{(l)}(N^{(l-1)} \circ \cdots \circ N^{(2)} \circ N^{(1)}(\mathbf{x})) - \mathbf{b}^{(l)} \text{ has a zero component}\} \text{ with } N^{(0)} = I.$$

Breaking lines are presented to provide a better understanding of the behavior of ReLU NN function approximation along the discontinous interface.

5.1. Two Dimensional Problems. We present numerical results for four two dimensional test problems with piecewise constant or variable advection velocity fields. All four test problems are defined on the domain $\Omega = (0, 1)^2$ with $\gamma = 1$, and the exact solutions are the same as the right-hand side functions, $u(x, y) = f(x, y)$, which are the step functions (except for the fourth test problem) along 3 line segment, 4 line segment, and curve interfaces. By Theorem 4.4, the LSNN method with 3 layer ReLU NN functions leads to

$$\left|\!\left|\!\left| u - u_{\scriptscriptstyle N} \right|\!\right|\!\right|_{\boldsymbol{\beta}} \le C \left| \alpha_1 - \alpha_2 \right| \sqrt{\varepsilon},$$

because the continuous part of the solution $\hat{u}$ is zero (again, except for the fourth test problem).

5.1.1. Problem with a 3line segment interface. This example is a modification of one from [5]. Let $\bar{\Omega} = \bar{\Upsilon}_1 \cup \bar{\Upsilon}_2 \cup \bar{\Upsilon}_3$ and

$$\Upsilon_1 = \{(x,y) \in \Omega : y \geq x\},\ \Upsilon_2 = \{(x,y) \in \Omega : x - \tfrac{a}{2} \leq y < x\},\ \text{and}\ \Upsilon_3 = \{(x,y) \in \Omega : y < x - \tfrac{a}{2}\}$$

with $a = 43/64$. The advective velocity field is a piecewise constant field given by

$$\boldsymbol{\beta}(x,y) = \begin{cases} (-1, \sqrt{2}-1)^T, & (x,y) \in \Upsilon_1, \\ (1-\sqrt{2}, 1)^T, & (x,y) \in \Upsilon_2, \\ (-1, \sqrt{2}-1)^T, & (x,y) \in \Upsilon_3. \end{cases} \tag{5.2}$$

The inflow boundary and the inflow boundary condition are given by

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

$$\text{and}\ \ g(x,y) = \begin{cases} 1, & (x,y) \in \Gamma_-^1 \equiv \{(1,y) : y \in [1 - \sqrt{2} + \frac{\sqrt{2}}{2}a, 1)\}, \\ -1, & (x,y) \in \Gamma_-^2 = \Gamma_- \setminus \Gamma_-^1, \end{cases}$$

respectively. Let

$$\hat{\Upsilon}_1 = \{(x,y) \in \Upsilon_1 : y < (1-\sqrt{2})x + a\},\ \hat{\Upsilon}_2 = \{(x,y) \in \Upsilon_2 : y < \tfrac{1}{1-\sqrt{2}}(x - \tfrac{a}{\sqrt{2}}) + \tfrac{a}{\sqrt{2}}\},$$

$$\text{and}\ \ \hat{\Upsilon}_3 = \{(x,y) \in \Upsilon_3 : y < (1-\sqrt{2})x + \tfrac{\sqrt{2}}{2}a\}.$$

The following right-hand side function is

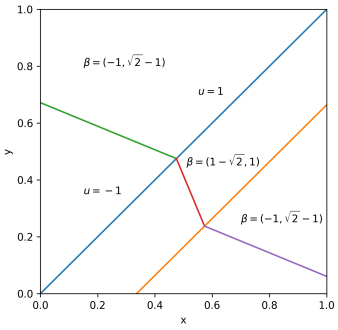
$$f(x,y) = \begin{cases} -1, & (x,y) \in \Omega_1 \equiv \hat{\Upsilon}_1 \cup \hat{\Upsilon}_2 \cup \hat{\Upsilon}_3, \\ 1, & (x,y) \in \Omega_2 = \Omega \setminus \Omega_1. \end{cases} \tag{5.3}$$

The LSNN method with a random initialization and 200000 iterations was implemented with 2-300-1 and 2-5-5-1 ReLU NN functions. The numerical results are presented in Figure 4 and Table 2. The numerical error (Table 2), trace (Figure 4(e)), and approximation graph (Figure 4(c)) of the 2 layer ReLU NN function approximation imply that the 2 layer network structure failed to approximate the solution (Figure 4(b)) especially around the discontinuous interface (Figure 4(a)), although the breaking lines (Figure 4(g)) indicate the approximation roughly formed transition layers around the interface. This and the remaining examples suggest the 2 layer network structure may not be able to approximate discontinuous solutions well as we expected in Remark 4.6. On the other hand, the 3 layer ReLU NN function approximation with the 2-5-5-1 structure with 4% of the number of parameters of the 2 layer one approximates the solution accurately. Again, this and the remaining examples suggest that 3 layer ReLU NN functions may be more efficient than 2 layer ones of even bigger sizes. In this example, because of the shape of the interface and $\hat{u} = 0$, the CPWL function p with small ε, which we constructed in Lemma 4.3 is expected to be a good approximation of the solution, and Figure 4(d) indicates that the approximation in $\mathcal{M}(2,1,3,10)$ is indeed such a function. The second layer breaking lines (Figure 4(h)) also help us to verify that sharp transition layers were generated along the discontinuous interface, which is again consistent with our convergence analysis. The trace (Figure 4(f)) of the 3 layer ReLU NN function approximation exhibits no oscillation.

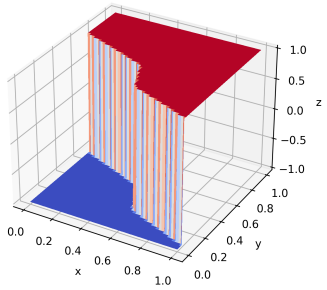
5.1.2. Problem with a 4 line segment interface. Let $\bar{\Omega} = \bar{\Upsilon}_1 \cup \bar{\Upsilon}_2 \cup \bar{\Upsilon}_3 \cup \bar{\Upsilon}_4$ and

$$\Upsilon_1 = \{(x,y) \in \Omega :\ y \geq x+1\},\ \Upsilon_2 = \{(x,y) \in \Omega,\ x \leq y < x+1\},$$

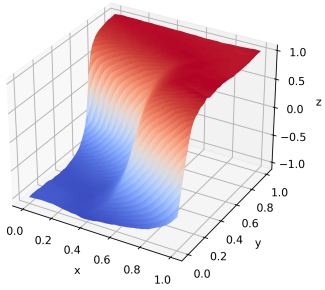
$$\Upsilon_3 = \{(x,y) \in \Omega,\ x-1 \leq y < x\},\ \text{and}\ \Upsilon_4 = \{(x,y) \in \Omega,\ y < x-1\}.$$



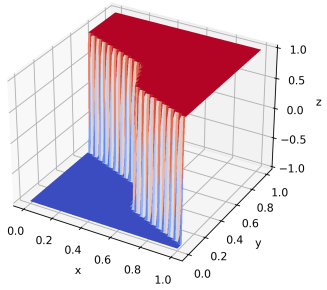
(a) Interface



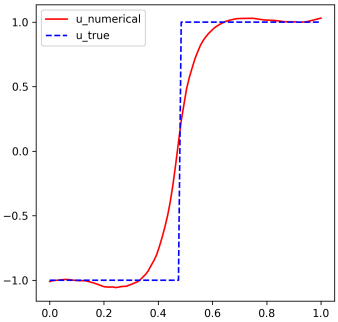
(b) Exact solution



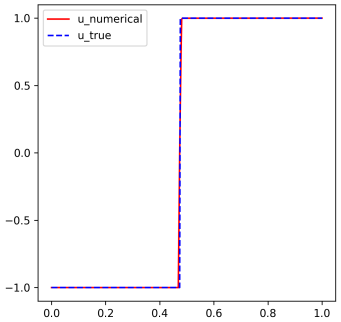
(c) 2 layer NN approximation



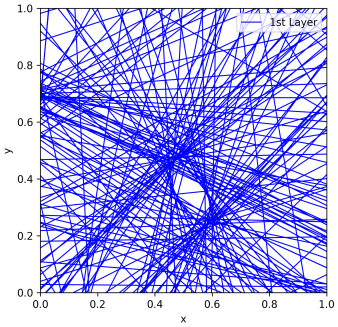
(d) 3 layer NN approximation



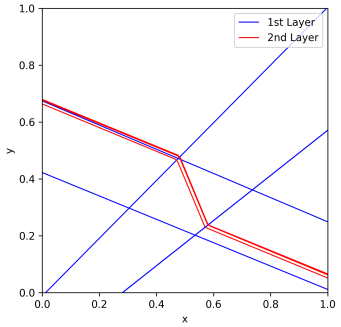
(e) 2 layer NN trace on $y = x$



(f) 3 layer NN trace on $y = x$



(g) 2 layer NN breaking lines



(h) 3 layer NN breaking lines

FIG. 4. *Approximation results of the problem in subsection* 5.1.1

TABLE 2
Relative errors of the problem in subsection 5.1.1

Network structure	$\frac{\|u-u_{\mathcal{T}}^N\|_0}{\|u\|_0}$	$\frac{\|\|\|u-u_{\mathcal{T}}^N\|\|\|_{\boldsymbol{\beta}}}{\|\|\|u\|\|\|_{\boldsymbol{\beta}}}$	$\frac{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{f})}{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{0})}$	Parameters
2-300-1	0.279867	0.404376	0.300774	1201
2-5-5-1	0.074153	0.079193	0.044987	51

The advective velocity field is a piecewise constant field given by

$$\boldsymbol{\beta}(x,y) = \begin{cases} (-1,\sqrt{2}-1)^T, & (x,y)\in\Upsilon_1,\\ (1-\sqrt{2},1)^T, & (x,y)\in\Upsilon_2,\\ (-1,\sqrt{2}-1)^T, & (x,y)\in\Upsilon_3,\\ (1-\sqrt{2},1)^T, & (x,y)\in\Upsilon_4. \end{cases} \tag{5.4}$$

The inflow boundary and the inflow boundary condition are given by

$$\Gamma_- = \{(2,y): y\in(0,2)\}\cup\{(x,0): x\in(0,2)\}$$

$$\text{and}\quad g(x,y) = \begin{cases} 1, & (x,y)\in\Gamma_-^1 \equiv \{(2,y): y\in(0,2)\},\\ -1, & (x,y)\in\Gamma_-^2 = \Gamma_-\setminus\Gamma_-^1, \end{cases}$$

respectively. Let

$$\hat{\Upsilon}_1 = \{(x,y)\in\Upsilon_1 : y < (1-\sqrt{2})x+2\},\ \hat{\Upsilon}_2 = \{(x,y)\in\Upsilon_2 : y < \tfrac{1}{1-\sqrt{2}}(x-1)+1\},$$

$$\hat{\Upsilon}_3 = \{(x,y)\in\Upsilon_3 : y < (1-\sqrt{2})(x-1)+1\},\ \text{and}\ \hat{\Upsilon}_4 = \{(x,y)\in\Upsilon_3 : y < \tfrac{1}{1-\sqrt{2}}x + \tfrac{2}{\sqrt{2}-1}\}.$$

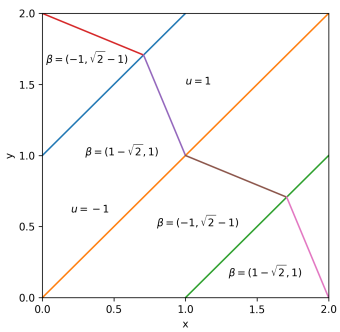
The following right-hand side function is

$$f(x,y) = \begin{cases} -1, & (x,y)\in\Omega_1 \equiv \hat{\Upsilon}_1\cup\hat{\Upsilon}_2\cup\hat{\Upsilon}_3\cup\hat{\Upsilon}_4,\\ 1, & (x,y)\in\Omega_2 = \Omega\setminus\Omega_1. \end{cases} \tag{5.5}$$

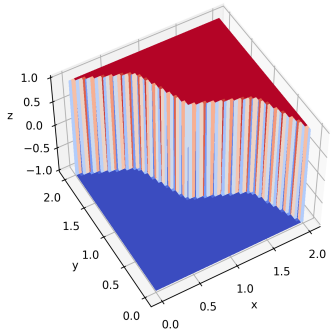
The LSNN method with a random initialization and 200000 iterations was implemented with 2-300-1 and 2-6-6-1 ReLU NN functions. The numerical results are presented in Figure 5 and Table 3. Since the interface has one more line segment than that of Example 5.1.1, we increased the number of hidden neurons to have higher expresiveness. The 2-6-6-1 structure with 6% of the number of parameters of the 2-300-1 structure approximated the solution (Figure 5(b)) accurately and Figure 5(d) indicates that the approximation in $\mathcal{M}(2,1,3,12)$ is the CPWL function p with small ε in Lemma 4.3. The trace (Figure 5(f)) shows no oscillation and the second layer breaking lines (Figure 5(h)) along the discontinuous interface (Figure 5(a)) show where the sharp transition layers were generated. On the other hand, the 2-300-1 ReLU NN function approximation roughly found the location of the interface (Figure 5(g)) but did not approximate the solution well (Figures 5(c) and 5(e) and Table 3).

5.1.3. Problem with a curve interface. The advective velocity field is a variable field given by

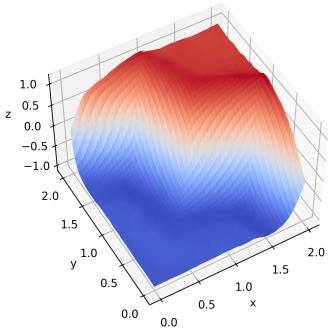
$$\boldsymbol{\beta}(x,y) = (1,2x),\ (x,y)\in\Omega. \tag{5.6}$$



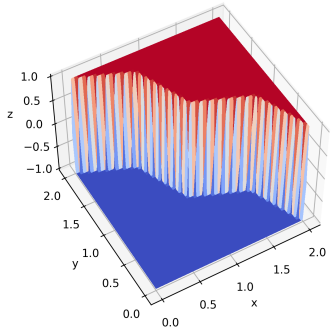
(a) Interface



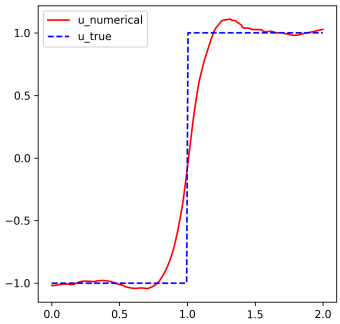
(b) Exact solution



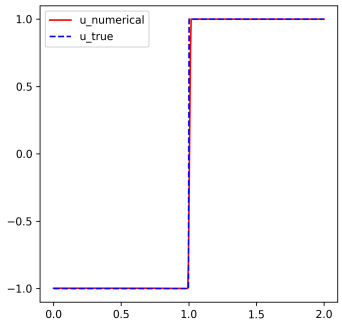
(c) 2 layer NN approximation



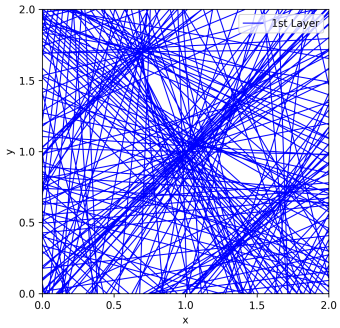
(d) 3 layer NN approximation



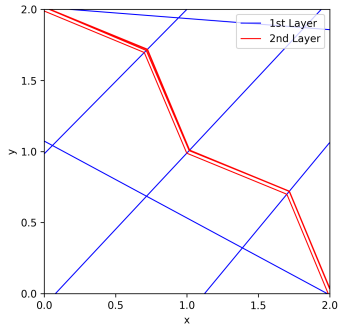
(e) 2 layer NN trace on $y = x$



(f) 3 layer NN trace on $y = x$



(g) 2 layer NN breaking lines



(h) 3 layer NN breaking lines

FIG. 5. *Approximation results of the problem in subsection* 5.1.2

TABLE 3
Relative errors of the problem in subsection 5.1.2

Network structure	$\frac{\|u-u_{\mathcal{T}}^N\|_0}{\|u\|_0}$	$\frac{\|\|\|u-u_{\mathcal{T}}^N\|\|\|_{\boldsymbol{\beta}}}{\|\|\|u\|\|\|_{\boldsymbol{\beta}}}$	$\frac{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{f})}{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{0})}$	Parameters
2-300-1	0.288282	0.358756	0.306695	1201
2-6-6-1	0.085817	0.091800	0.069808	67

The inflow boundary and the inflow boundary condition are given by

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

$$\text{and} \quad g(x,y) = \begin{cases} 1, & (x,y) \in \Gamma_-^1 \equiv \{(0,y) : y \in [\frac{1}{8},1)\}, \\ 0, & (x,y) \in \Gamma_-^2 = \Gamma_- \setminus \Gamma_-^1, \end{cases}$$

respectively. The following right-hand side function is

$$f(x,y) = \begin{cases} 0, & (x,y) \in \Omega_1 \equiv \{(x,y) \in \Omega : y < x^2 + \frac{1}{8}\}, \\ 1, & (x,y) \in \Omega_2 = \Omega \setminus \Omega_1. \end{cases} \tag{5.7}$$

The LSNN method with a random initialization and 300000 iterations was implemented with 2-3000-1 and 2-60-60-1 ReLU NN functions. We again increased the number of hidden neurons for the 3 layer network structure, assuming CPWL functions approximating a curved discontinuous interface (Figure 6(a)) well would be a ReLU NN function with more hidden neurons. The numerical results are presented in Figure 6 and Table 4. Figures 6(c), 6(e), and 6(g) suggest that the 2 layer network structure failed to approximate the solution (Figure 6(b)) around the discontinuous interface with more than three times the number of parameters of the 3 layer network structure. In contrast, the 3 layer network structure shows better numerical errors (Table 4) and pointwise approximations (Figures 6(d) and 6(f)), locating the discontinuous interface (Figure 6(h)).

TABLE 4
Relative errors of the problem in subsection 5.1.3

Network structure	$\frac{\|u-u_{\mathcal{T}}^N\|_0}{\|u\|_0}$	$\frac{\|\|\|u-u_{\mathcal{T}}^N\|\|\|_{\boldsymbol{\beta}}}{\|\|\|u\|\|\|_{\boldsymbol{\beta}}}$	$\frac{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{f})}{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{0})}$	Parameters
2-3000-1	0.134514	0.181499	0.078832	12001
2-60-60-1	0.066055	0.106095	0.030990	3901

5.1.4. Problem with a curve interface and $\hat{u} \neq 0$. The advective velocity field is a variable field given by

$$\boldsymbol{\beta}(x,y) = (-y,x), \ (x,y) \in \Omega. \tag{5.8}$$

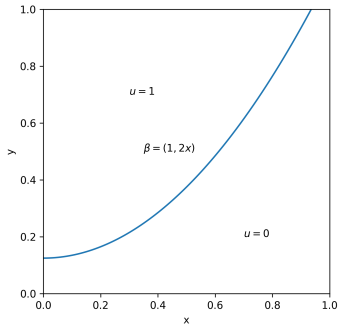
The inflow boundary and the inflow boundary condition are given by

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

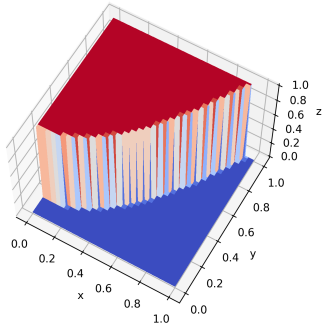
$$\text{and} \quad g(x,y) = \begin{cases} -1 + x^2 + y^2, & (x,y) \in \Gamma_-^1 \equiv \{(x,0) : x \in (0,\frac{2}{3})\}, \\ 1 + x^2 + y^2, & (x,y) \in \Gamma_-^2 = \Gamma_- \setminus \Gamma_-^1, \end{cases}$$

respectively. The following right-hand side function is

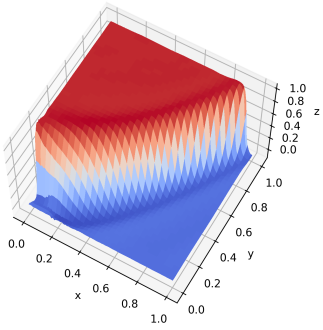
$$f(x,y) = \begin{cases} -1 + x^2 + y^2, & (x,y) \in \Omega_1 \equiv \{(x,y) \in \Omega : y < \sqrt{\frac{4}{9} - x^2}\}, \\ 1 + x^2 + y^2, & (x,y) \in \Omega_2 = \Omega \setminus \Omega_1. \end{cases} \tag{5.9}$$



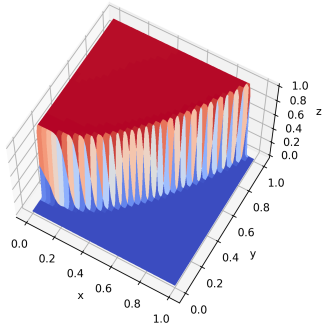
(a) Interface



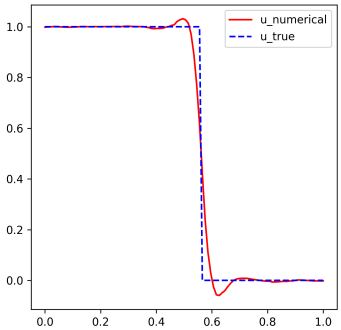
(b) Exact solution



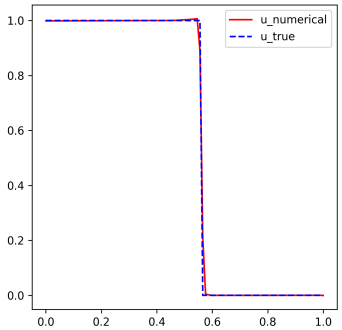
(c) 2 layer NN approximation



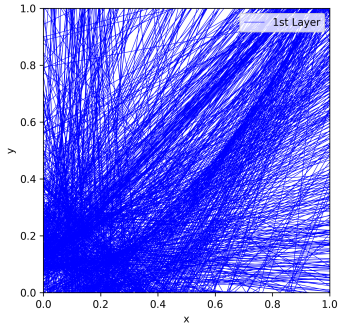
(d) 3 layer NN approximation



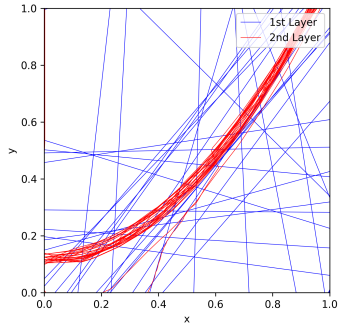
(e) 2 layer NN trace on $y = x$



(f) 3 layer NN trace on $y = x$



(g) 2 layer NN breaking lines



(h) 3 layer NN breaking lines

FIG. 6. *Approximation results of the problem in subsection* 5.1.3

The LSNN method with a random initialization and 200000 iterations was implemented with 2-4000-1 and 2-65-65-1 ReLU NN functions. The numerical results are presented in Figure 7 and Table 5. Table 5 indicates that the 2 layer network structure is capable of approximating the solution (Figure 7(b)) in average, but Figures 7(c), 7(e), and 7(g) show difficulty around the discontinuous interface (Figure 7(a)). Again, the 3 layer network structure with 28% of the number of parameters of the 2 layer network structure presented better error results (Table 5) and approximated the solution accurately pointwise (Figures 7(d) and 7(f)). Unlike other examples, some of the second layer breaking lines (Figure 7(h)) of the approximation spread out on the whole domain in addition to those around the interface, which implies they are necessary for approximating the solution with $\hat{u} \neq 0$.

TABLE 5
Relative errors of the problem in subsection 5.1.4

Network structure	$\frac{\|u-u_{\mathcal{T}}^N\|_0}{\|u\|_0}$	$\frac{\|\|\|u-u_{\mathcal{T}}^N\|\|\|_{\boldsymbol{\beta}}}{\|\|\|u\|\|\|_{\boldsymbol{\beta}}}$	$\frac{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{f})}{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{0})}$	Parameters
2-4000-1	0.088349	0.108430	0.058213	16001
2-65-65-1	0.048278	0.073095	0.015012	4551

5.2. Three Dimensional Problems. We present numerical results for two three dimensional test problems with piecewise constant or variable advection velocity fields whose solutions are piecewise constant along a connected series of planes or surfaces. Both test problems are defined on the domain $\Omega = (0,1)^3$, approximation results are depicted on $z = 0.5$, and again, we have

$$|||u - u_N|||_{\boldsymbol{\beta}} \leq C\left|\alpha_1 - \alpha_2\right|\sqrt{\varepsilon}.$$

5.2.1. Problem with a piecewise plane interface. Let $\gamma = f = 0$, $\bar{\Omega} = \bar{\Upsilon}_1 \cup \bar{\Upsilon}_2$, and

$$\Upsilon_1 = \{(x,y,z) \in \Omega : y < x\},\ \Upsilon_2 = \{(x,y,z) \in \Omega : y \geq x\}.$$

The advective velocity field is a piecewise constant field given by

$$\boldsymbol{\beta}(x,y,z) = \begin{cases} (1-\sqrt{2},1,0)^T, & (x,y,z) \in \Upsilon_1 \\ (-1,\sqrt{2}-1,0)^T, & (x,y,z) \in \Upsilon_2. \end{cases} \tag{5.10}$$

The inflow boundary and the inflow boundary condition are given by

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

$$\text{and}\ \ g(x,y,z) = \begin{cases} 0, & (x,y,z) \in \Gamma_-^1 \equiv \{(x,0,z) : x \in (0,0.7),\ z \in (0,1)\}, \\ 1, & (x,y,z) \in \Gamma_-^2 = \Gamma_- \setminus \Gamma_-^1, \end{cases}$$

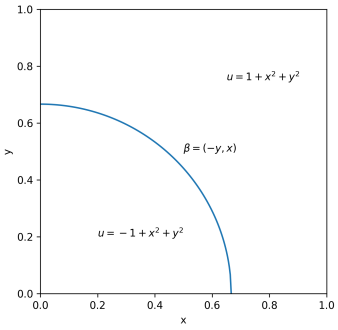
respectively. Let

$$\Omega_1 = \{(x,y,z) \in \Omega : y < (1-\sqrt{2})x + 0.7,\ y < \tfrac{1}{1-\sqrt{2}}(x-0.7)\}.$$

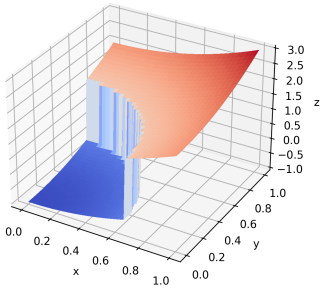
The exact solution is a unit step function in three dimensions

$$u(x,y,z) = \begin{cases} 0, & (x,y,z) \in \Omega_1, \\ 1, & (x,y,z) \in \Omega_2 = \Omega \setminus \Omega_1. \end{cases} \tag{5.11}$$

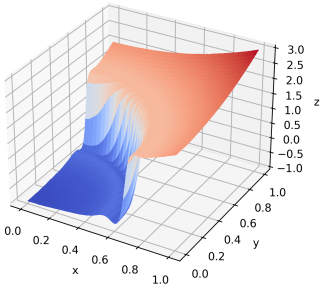
The LSNN method with a random initialization and 100000 iterations was implemented with 3-300-1 and 3-5-5-1 ReLU NN functions (depth $\lceil \log_2(d+1) \rceil + 1 = 3$ for $d = 3$). For three dimensions



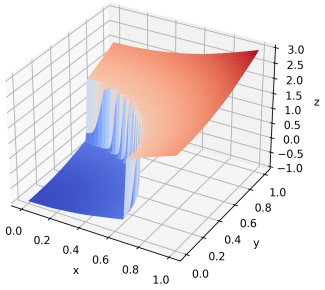
(a) Interface



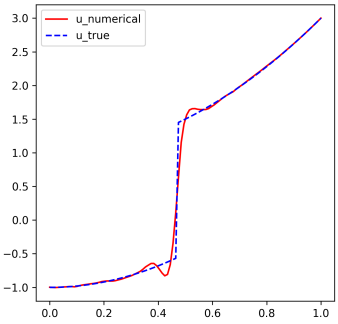
(b) Exact solution



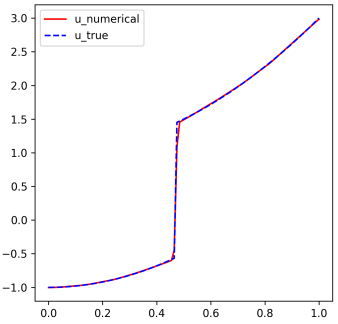
(c) 2 layer NN approximation



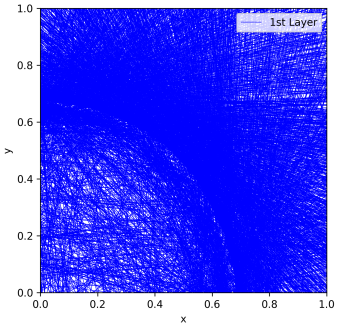
(d) 3 layer NN approximation



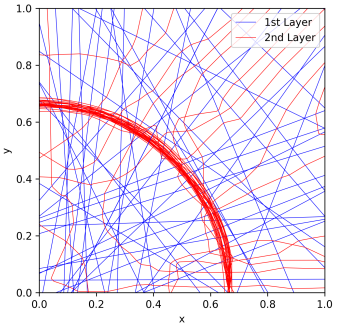
(e) 2 layer NN trace on $y = x$



(f) 3 layer NN trace on $y = x$



(g) 2 layer NN breaking lines



(h) 3 layer NN breaking lines

FIG. 7. *Approximation results of the problem in subsection* 5.1.4

again, the 2 layer network structure with a large number of parameters generated transition layers along the discontinuous interface (Figures 8(a) and 8(g)) but had an issue with approximating the solution (Figure 8(b)) accurately pointwise (Figures 8(c) and 8(e)). The 3 layer network structure with 4% of the number of parameters of the 2 layer network approximated the solution accurately (Table 6). As explained in Example 5.1.1, Figures 8(d), 8(f), and 8(h) also indicate that the function p in Lemma 4.3 appears to be the approximation and in this example, be contained in $\mathcal{M}(3,1,3,10)$.

TABLE 6
Relative errors of the problem in subsection 5.2.1

Network structure	$\frac{\|u-u_{\mathcal{T}}^N\|_0}{\|u\|_0}$	$\frac{\|\|\|u-u_{\mathcal{T}}^N\|\|\|_{\boldsymbol{\beta}}}{\|\|\|u\|\|\|_{\boldsymbol{\beta}}}$	$\frac{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{f})}{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{0})}$	Parameters
3-300-1	0.185006	0.214390	0.189820	1501
3-5-5-1	0.055365	0.055370	0.045902	56

5.2.2. Problem with a cylindrical interface. Let $\gamma = 1$. The advective velocity field is a variable field given by

$$\boldsymbol{\beta}(x,y,z) = (-y,x,0)^T,\ (x,y,z) \in \Omega. \tag{5.12}$$

The inflow boundary and the inflow boundary condition are given by

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

$$\text{and}\ \ g(x,y,z) = \begin{cases} 0, & (x,y,z) \in \Gamma_-^1 \equiv \{(x,0,z) : x \in (0,0.7),\ z \in (0,1)\}, \\ 1, & (x,y,z) \in \Gamma_-^2 = \Gamma_- \setminus \Gamma_-^1. \end{cases}$$

respectively. Let

$$\Omega_1 = \{(x,y,z) \in \Omega : y < \sqrt{0.7^2 - x^2}.\}$$

The following right-hand side function is

$$f(x,y,z) = \begin{cases} 0, & (x,y,z) \in \Omega_1, \\ 1, & (x,y,z) \in \Omega_2 = \Omega \setminus \Omega_1. \end{cases} \tag{5.13}$$

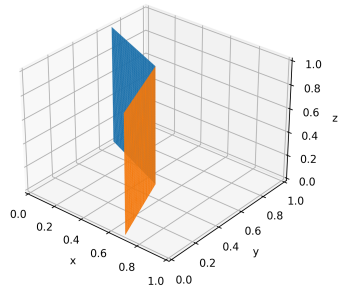
The exact solution is

$$u(x,y,z) = f(x,y,z),\ (x,y,z) \in \Omega.$$

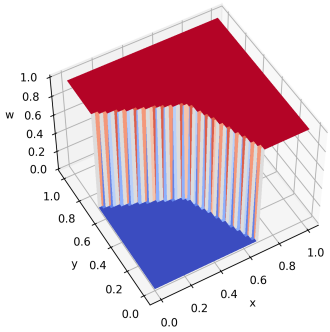
The LSNN method with a random initialization and 150000 iterations was implemented with 3-1500-1 and 3-50-50-1 ReLU NN functions. Again to minimize the loss function over a larger subset of CPWL functions, we increased the number of hidden neurons. The numerical results are presented in Figure 9 and Table 7. Even though the 2 layer ReLU NN function approximation provides an approximate location of the discontinuous interface (Figures 9(a) and 9(g)), the structure failed to approximate the solution (Figure 9(b)) around the interface accurately (Figures 9(c) and 9(e)). The 3 layer network structure with less than 40% of the number of parameters approximated the solution well, locating the discontinuous interface (Figures 9(d), 9(f), and 9(h) and Table 7).

REFERENCES

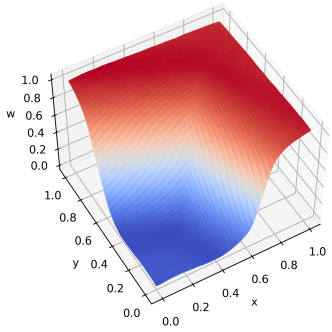
[1] R. ARORA, A. BASU, P. MIANJY, AND A. MUKHERJEE, *Understanding deep neural networks with rectified linear units*, arXiv preprint arXiv:1611.01491, (2016), https://doi.org/10.48550/arXiv.1611.01491.



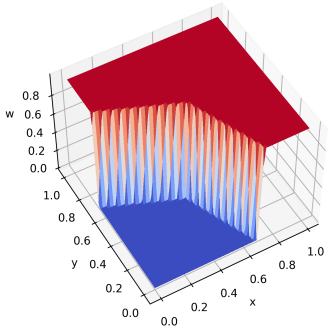
(a) Interface



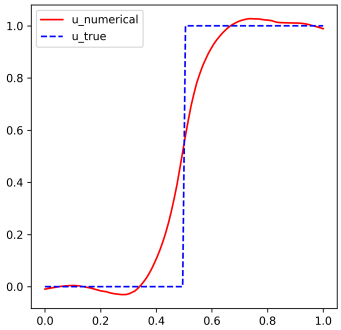
(b) Exact solution



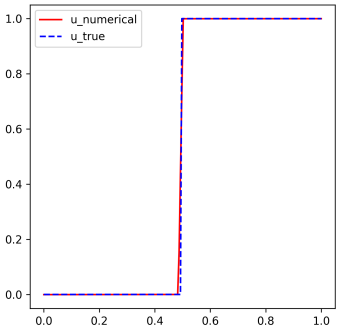
(c) 2 layer NN approximation



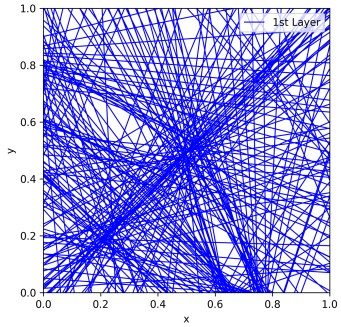
(d) 3 layer NN approximation



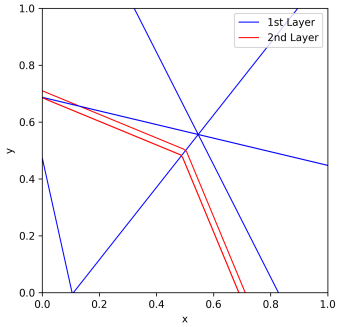
(e) 2 layer NN trace on $y = x$



(f) 3 layer NN trace on $y = x$

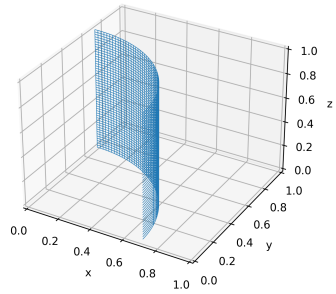


(g) 2 layer NN breaking lines

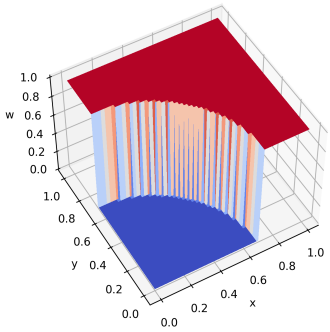


(h) 3 layer NN breaking lines

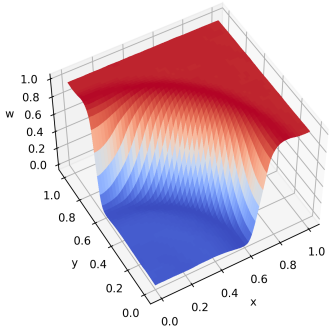
FIG. 8. *Approximation results of the problem in subsection* 5.2.1



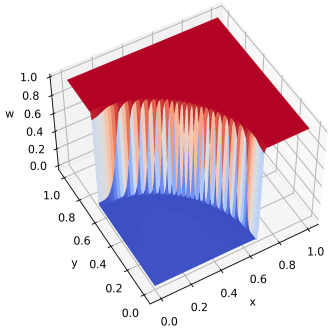
(a) Interface



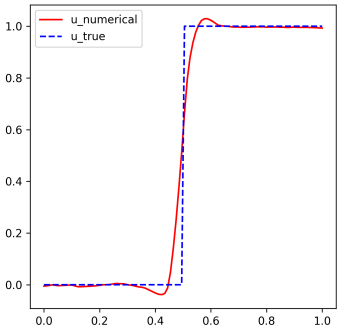
(b) Exact solution



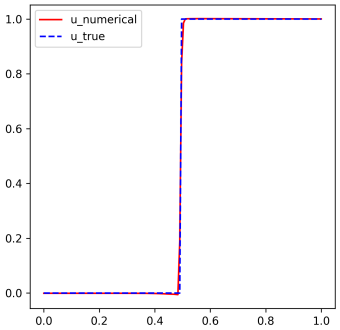
(c) 2 layer NN approximation



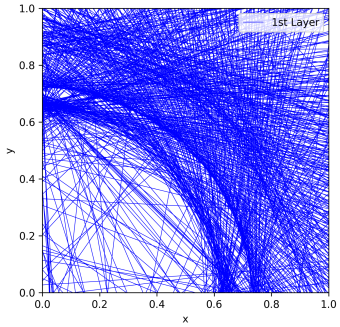
(d) 3 layer NN approximation



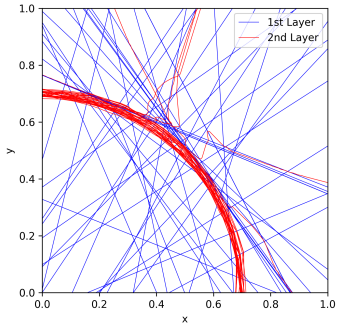
(e) 2 layer NN trace on $y = x$



(f) 3 layer NN trace on $y = x$



(g) 2 layer NN breaking lines



(h) 3 layer NN breaking lines

FIG. 9. *Approximation results of the problem in subsection* 5.2.2

TABLE 7
Relative errors of the problem in subsection 5.2.2

Network structure	$\frac{\|u-u_{\mathcal{T}}^N\|_0}{\|u\|_0}$	$\frac{\|\|\|u-u_{\mathcal{T}}^N\|\|\|_{\boldsymbol{\beta}}}{\|\|\|u\|\|\|_{\boldsymbol{\beta}}}$	$\frac{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{f})}{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{0})}$	Parameters
3-1500-1	0.125142	0.158393	0.117929	7501
3-50-50-1	0.050217	0.073780	0.018976	2801

[2] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner, *Learning data-driven discretizations for partial differential equations*, Proc. Natl. Acad. Sci. USA, 116 (31) (2019), pp. 15344–15349, https://doi.org/10.1073/pnas.1814058116.

[3] J. Berg and K. Nyström, *A unified deep artificial neural network approach to partial differential equations in complex geometries*, Neurocomputing, 317 (2018), pp. 28–41, https://doi.org/10.1016/j.neucom.2018.06.056.

[4] P. Bochev and M. Gunzburger, *Least-squares methods for hyperbolic problems*, in Handb. Numer. Anal., vol. 17, Elsevier, 2016, pp. 289–317, https://doi.org/10.1016/bs.hna.2016.07.002.

[5] Z. Cai, J. Chen, and M. Liu, *Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation*, J. Comput. Phys., 443 (2021), 110514, https://doi.org/10.1016/j.jcp.2021.110514.

[6] Z. Cai, J. Chen, and M. Liu, *Self-adaptive deep neural network: numerical approximation to functions and PDEs*, J. Comput. Phys., 455 (2022), 111021, https://doi.org/10.1016/j.jcp.2022.111021.

[7] Z. Cai, J. Chen, and M. Liu, *Least-squares neural network (LSNN) method for scalar nonlinear hyperbolic conservation laws: discrete divergence operator*, J. Comput. Appl. Math., to appear, arXiv:2110.10895v2 [math.NA] (2021), https://doi.org/10.48550/arXiv.2110.10895.

[8] Z. Cai, J. Chen, and M. Liu, *Least-squares ReLU neural network (LSNN) method for scalar nonlinear hyperbolic conservation law*, Appl. Numer. Math., 174 (2022), pp. 163–176, https://doi.org/10.1016/j.apnum.2022.01.002.

[9] Z. Cai, J. Chen, M. Liu, and X. Liu, *Deep least-squares methods: an unsupervised learning-based numerical method for solving elliptic PDEs*, J. Comput. Phys., 420 (2020), p. 109707, https://doi.org/10.1016/j.jcp.2020.109707.

[10] A. Caragea, P. Petersen, and F. Voigtlaender, *Neural network approximation and estimation of classifiers with classification boundary in a barron class*, arXiv preprint arXiv:2011.09363, (2020), https://doi.org/10.48550/arXiv.2011.09363.

[11] W. Dahmen, C. Huang, C. Schwab, and G. Welper, *Adaptive Petrov–Galerkin methods for first order transport equations*, SIAM J. Numer. Anal., 50 (2012), pp. 2420–2445, https://doi.org/10.1137/110823158.

[12] H. De Sterck, T. A. Manteuffel, S. F. McCormick, and L. Olson, *Least-squares finite element methods and algebraic multigrid solvers for linear hyperbolic PDEs*, SIAM J. Sci. Comput., 26 (2004), pp. 31–54, https://doi.org/10.1137/S106482750240858X.

[13] R. DeVore, B. Hanin, and G. Petrova, *Neural network approximation*, Acta Numer., 30 (2021), pp. 327–444, https://doi.org/10.1017/S0962492921000052.

[14] R. A. DeVore, K. I. Oskolkov, and P. P. Petrushev, *Approximation by feed-forward neural networks*, Annals of Numerical Mathematics, 4 (1996), pp. 261–288.

[15] M. W. M. G. Dissanayake and N. Phan-Thien, *Neural network based approximations for solving partial differential equations*, Communications in Numerical Methods in Engineering, 10 (1994), pp. 195–201.

[16] W. E and B. Yu, *The deep ritz method: A deep learning-based numerical algorithm for solving variational problems*, Commun. Math. Stat., 6 (2018), pp. 1–12, https://doi.org/10.1007/s40304-018-0127-z.

[17] S. Hon and H. Yang, *Simultaneous neural network approximations in Sobolev spaces*, arXiv preprint arXiv:2109.00161v1, (2021), https://doi.org/10.48550/arXiv.2109.00161.

[18] P. Houston, R. Rannacher, and E. Süli, *A posteriori error analysis for stabilised finite element approximations of transport problems*, Comput. Methods Appl. Mech. Engrg., 190 (2000), pp. 1483–1508, https://doi.org/10.1016/S0045-7825(00)00174-2.

[19] M. Imaizumi and K. Fukumizu, *Deep neural networks learn non-smooth functions effectively*, in The 22nd international conference on artificial intelligence and statistics, PMLR, 2019, pp. 869–878.

[20] M. Imaizumi and K. Fukumizu, *Advantage of deep neural networks for estimating functions with singularity on hypersurfaces*, Journal of Machine Learning Research, 23 (2022), pp. 1–53.

[21] Y. Kim, I. Ohn, and D. Kim, *Fast convergence rates of deep neural networks for classification*, Neural Networks, 138 (2021), pp. 179–197, https://doi.org/10.1016/j.neunet.2021.02.012.

[22] D. P. Kingma and J. Ba, *ADAM: A method for stochastic optimization*, in International Conference on Representation Learning, San Diego, 2015.

[23] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Transactions 941 on Neural Networks, 9 (1998), p. 987–1000.

[24] M. Liu, Z. Cai, and J. Chen, *Adaptive two-layer ReLU neural network: I. best least-squares approximation*,

Comput. Math. Appl., 113 (2022), pp. 34–44, https://doi.org/10.1016/j.camwa.2022.03.005.
[25] Q. Liu and S. Zhang, *Adaptive least-squares finite element methods for linear transport equations based on an H(div) flux reformulation*, Comput. Methods Appl. Mech. Engrg., (366 (2020) 113041), https://doi.org/10.1016/j.cma.2020.113041.
[26] P. Petersen and F. Voigtlaender, *Optimal approximation of piecewise smooth functions using deep relu neural networks*, Neural Networks, 108 (2018), pp. 296–330, https://doi.org/10.1016/j.neunet.2018.08.019.
[27] P. P. Petrushev, *Approximation by ridge functions and neural networks*, SIAM J. Math. Anal., 30 (1998), pp. 155–189, https://doi.org/10.1137/S0036141097322959.
[28] A. Pinkus, *Approximation theory of the MLP model in neural networks*, Acta Numer., 8 (1999), pp. 143–195, https://doi.org/10.1017/S0962492900002919.
[29] M. Raissi, P. Perdikaris, and G. 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), pp. 686–707, https://doi.org/10.1016/j.jcp.2018.10.045.
[30] Z. Shen, H. Yang, and S. Zhang, *Deep network approximation characterized by number of neurons*, arXiv preprint arXiv:1906.05497, (2019), https://doi.org/10.48550/arXiv.1906.05497.
[31] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, J. Comput. Phys., 375 (2018), pp. 1139–1364, https://doi.org/10.1016/j.jcp.2018.08.029.
[32] J. Tarela and M. Martinez, *Region configurations for realizability of lattice piecewise-linear models*, Math. Comput. Modelling, 30 (1999), pp. 17–27, https://doi.org/10.1016/S0895-7177(99)00195-8.
[33] S. Wang and X. Sun, *Generalization of hinging hyperplanes*, IEEE Trans. Inform. Theory, 51 (2005), pp. 4425–4431, https://doi.org/10.1109/TIT.2005.859246.