SIAM J. SCI. COMPUT.
Vol. 46, No. 4, pp. C448–C478

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

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

Abstract. We studied the least-squares ReLU neural network (LSNN) method for solving a linear advection-reaction equation with discontinuous solution in [Z. Cai et al., *J. Comput. Phys.,* 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 properly designed and physics preserved discrete differential operator. In this paper, we study the LSNN method for problems with discontinuity 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 a discontinuity interface generated by a vector field as streamlines with any prescribed accuracy. By decomposing the solution into continuous and discontinuous parts, we prove theoretically that the 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 test problems with various discontinuity interfaces show that the LSNN method with enough layers is accurate and does not exhibit the common Gibbs phenomena along discontinuity interfaces.

Key words. least-squares method, ReLU neural network, linear advection-reaction equation, discontinuous solution

MSC codes. 65N15, 65N99

DOI. 10.1137/23M1568107

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

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

where $\boldsymbol{\beta}(\mathbf{x}) = (\beta_1, \ldots, \beta_d)^T \in C^0(\bar{\Omega})^d$ is a given advective velocity field, $u_{\boldsymbol{\beta}} = \boldsymbol{\beta} \cdot \nabla u$ 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 $\boldsymbol{n}(\mathbf{x})$ being the unit outward normal vector to Γ at $\mathbf{x} \in \Gamma$. We assume that the reaction coefficient $\gamma \in C^0(\bar{\Omega})$, the source term $f \in L^2(\Omega)$, and $g \in L^2(\Gamma_-)$.

When the inflow boundary data g is discontinuous, so is the solution of (1.1) as the solution is entirely determined by the characteristic curves (see Lemma 4.1),

*Submitted to the journal's Machine Learning Methods for Scientific Computing section April 24, 2023; accepted for publication (in revised form) June 5, 2024; published electronically August 12, 2024.

https://doi.org/10.1137/23M1568107

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

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

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

along which the discontinuities are propagated across the domain. The discontinuity 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 method is usually limited to linear problems and is difficult to extend to nonlinear hyperbolic conservation laws (HCLs).

In [6], we studied the least-squares rectified linear unit (ReLU) neural network (LSNN) for solving (1.1) with a discontinuous solution. The method is based on the $L^2(\Omega)$ norm least-squares formulation analyzed in [13, 4] and employs a new class of approximating functions: multilayer perceptrons with the ReLU activation function, i.e., 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 properly 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, 5]). Compared to mesh-based numerical methods including various adaptive mesh refinement algorithms that locate the discontinuity interface through local mesh refinement (see, e.g., [12, 19, 28]), the LSNN method, a meshfree and pointfree method, is much more effective in terms of the number of degrees of freedom. Theoretically, it was shown in [6] 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 [6] is very restrictive even in two dimensions. In general, the discontinuous solution of (1.1) has interfaces that are hypersurfaces in d dimensions. The purpose of this paper is twofold. First, we show that any step function with interface that is generated by a vector field as streamlines 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 [6]. 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 [1, 39, 40] (see Proposition 2.1). A question on approximating piecewise smooth functions by ReLU NNs also arises in data science applications such as classification, etc. Some convergence rates were obtained in [33, 10, 21, 20, 22]. Particularly, for a given C^β ($\beta > 0$) interface, [33] established approximation rates of ReLU NNs with no more than $L(\beta, d) = (3 + \lceil \log_2 \beta \rceil)(11 + 2\beta/d)$ layers. This upper bound is not applicable to a C^0 interface and becomes large for a very smooth interface.

Second, we establish a new kind of a priori error estimates (see Theorem 4.5) for the LSNN method in d dimensions for a discontinuity interface. To do so, we decompose the solution as the sum of the discontinuous and continuous parts (see (4.4)). The continuous part of the solution may be approximated well by (even shallow) ReLU NN functions with a standard approximation property (see, e.g., [15, 34, 35, 37, 14]). 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, which is high dimensional and nonconvex, when using stochastic gradient descent, Adam [23], etc., even though the degrees of freedom of the LSNN method are much less than those of mesh-based numerical methods.

Followed by recent success of deep neural networks (DNNs) in machine learning and artificial intelligence tasks such as computer vision and pattern recognition, there has been active interest in using DNNs for solving partial differential equations (PDEs) (see, e.g., [2, 3, 9, 17, 36, 38]). Due to the fact that the collection of DNN functions is not a linear space, NN-based methods for solving PDEs may be categorized as the Ritz and least-squares methods. The former (see, e.g., [17]) 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 the 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 the 1990s (see, e.g., [16, 24]), where the discrete L^2 norm on a uniform integration mesh was employed for both PDEs of the strong form and their boundary/initial conditions. Along this line, it is the popular physics-informed neural networks (PINNs) by Raissi, Perdikaris, and Karniadakis [36] in 2019 which uses autodifferentiation 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 reviews the LSNN method in [6] and formulates the method based on the framework in section 2. Then we prove that the method is capable of locating any discontinuity interfaces of the problem in section 4. Finally, section 5 presents numerical results for both two- and three-dimensional test problems with various discontinuity interfaces.

2. ReLU NN functions. First we begin with the definition of the ReLU activation function. The ReLU activation function σ is defined by

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

We say that a function $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}^c$ with $c, d \in \mathbb{N}$ is a ReLU 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 $l = 1, \ldots, L-1$, $N^{(l)}(\mathbf{x}) = \sigma(\boldsymbol{\omega}^{(l)}\mathbf{x} - \mathbf{b}^{(l)})$ for all $\mathbf{x} \in \mathbb{R}^{n_{l-1}}$ for $\boldsymbol{\omega}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$, 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 $L > 1$ (not unique) as in (2.1). Then we say

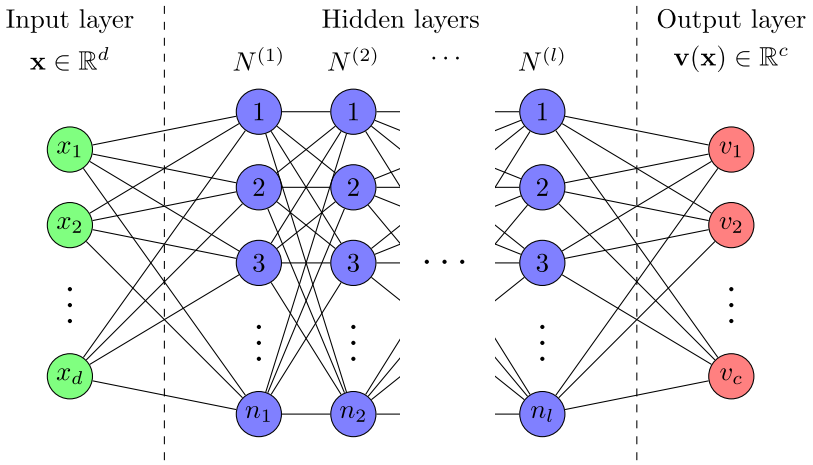


FIG. 1. *The NN function structure.*

1. $N^{(l)}$ is the lth layer (or also the lth hidden layer when $l < L$) of the representation, and the representation has L layers or depth L, and $L-1$ hidden layers;
2. the entries of $\boldsymbol{\omega}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases, respectively, of the lth layer (or also the lth hidden layer when $l < L$);
3. the natural number n_l is the width or the number of neurons of the lth layer (or also the lth hidden layer when $l < L$).

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 representations with depth L and the total number of neurons of the hidden layers n by

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

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

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

We now introduce another function class, which is the set of CPWL functions, and explore a theorem about the relationship between the two function classes. We say that a function $f : \mathbb{R}^d \to \mathbb{R}$ with $d \in \mathbb{N}$ is 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 $\mathbb{R}^d$,
3. f is affine linear on each polyhedron in the set, i.e., on each polyhedron in the set, $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$ for all $\mathbf{x} \in \mathbb{R}^d$ 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 $f:\mathbb{R}^d\to\mathbb{R}$ is equal to $\mathcal{M}(\lceil\log_2(d+1)\rceil+1)$, i.e., the set of all ReLU NN functions from $\mathbb{R}^d$ to $\mathbb{R}$ that have representations with depth $\lceil\log_2(d+1)\rceil+1$.*

Proof. $\mathcal{M}(\lceil\log_2(d+1)\rceil+1)$ is clearly a subset of the set of CPWL functions. Conversely, it is 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 $\mathcal{M}(L)\subset\mathcal{M}(\lceil\log_2(d+1)\rceil+1)$ 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 representations to 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 of the hidden layers. Therefore we suggest the following approach. The following proposition is a trivial fact.

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

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

Finally, in Figures 3 to 7 and 9 to 11, by the lth (hidden) layer breaking hyperplanes of a given representation as in (2.1) (with the output dimension being 1), we shall mean the set

- $\{\mathbf{x}\in\Omega : \boldsymbol{\omega}^{(1)}\mathbf{x}-\mathbf{b}^{(1)}$ has a zero component$\}$ when $l=1$,
- $\{\mathbf{x}\in\Omega : \boldsymbol{\omega}^{(l)}(N^{(l-1)}\circ\cdots\circ N^{(2)}\circ N^{(1)}(\mathbf{x}))-\mathbf{b}^{(l)}$ has a zero component$\}$ when $2\le l<L$.

Breaking hyperplanes give a partition of $\mathbb{R}^d$, and on each element in the partition, the ReLU NN function is affine linear. (Breaking hyperplanes correspond to boundaries of linear regions of NNs as introduced in [31, 32].) They will be presented to help our understanding of the graphs of ReLU NN function approximations, especially along discontinuity interfaces.

3. The LSNN method. We define the least-squares functional as

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

where $\mathbf{f}=(f,g)$, and $\|\cdot\|_{0,\Omega}$ and $\|\cdot\|_{-\boldsymbol{\beta}}$ denote, respectively, the $L^2(\Omega)$ norm and 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, 13]). *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 least-squares 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

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

The norm equivalence (3.4) implies that problem (3.2) is well posed.

PROPOSITION 3.2 (see [4, 13]). *Problem (3.2) has a unique solution $u \in V_{\boldsymbol{\beta}}$ satisfying the following a priori estimate:*

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

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

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

LEMMA 3.3 (see [6]). *Let u and $u_{_N}$ be the solutions of problems (3.2) and (3.6), respectively. Then we have*

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

where α and M are constants in (3.4).

Optimization methods for solving the LSNN discretization problem in (3.6) repeatedly compute the integration

$$\int_\Omega \left(v_{\boldsymbol{\beta}} + \gamma\, v - f\right)^2 (\mathbf{x})\, d\ \mathbf{x} \tag{3.8}$$

for function v in $\mathcal{M}(L,n)$. In practice, the integration in (3.8) is approximated by a numerical integration. Unlike conventional numerical methods using fixed meshes, designing an efficient and accurate numerical integration for (3.8) is a nontrivial task. Apparently, the commonly used numerical integration of Monte Carlo type in scientific machine learning is inaccurate for problems with local features. It is also obvious that an accurate numerical integration should be based on the integral of the exact solution u, i.e.,

$$\int_\Omega \left(u_{\boldsymbol{\beta}} + \gamma\, u - f\right)^2 (\mathbf{x})\, d\ \mathbf{x}. \tag{3.9}$$

However, the exact solution u and hence the integrand in (3.9) are unknown.

To circumvent this difficulty, adaptive numerical integration was proposed and studied in the context of the deep Ritz method for the linear elasticity equation in [27]. Adaptive numerical integration is based on a composite numerical integration

$$\sum_{K\in\mathcal{T}} \mathcal{Q}_K(w) \approx \int_\Omega w(\mathbf{x})\,d\mathbf{x} = \sum_{K\in\mathcal{T}} \int_K w(\mathbf{x})\,d\mathbf{x},$$

where $\mathcal{T} = \{K : K \text{ is an open subdomain of } \Omega\}$ is a partition of Ω and $\mathcal{Q}_K(w) \approx \int_K w(\mathbf{x})\,d\mathbf{x}$ denotes a quadrature rule over K. First, $\mathcal{Q}_K$ may vary on $K \in \mathcal{T}$. Second, its choice is one of the standard quadrature rules like the Gaussian quadrature or Newton–Cotes formulas such as the midpoint, trapezoidal, or Simpson rule (see [5]). In the case of the midpoint rule for all $K \in \mathcal{T}$, $\mathcal{Q}_K(w) = w(\mathbf{x}_K)|K|$, where $\mathbf{x}_K$ is the centroid of K and $|K|$ is the d-dimensional measure of K.

Remark 3.4. The LSNN approximation $u_N \in \mathcal{M}(L, n)$ defined in (3.6) is CPWL with respect to a partition of the domain Ω, referred to as the physical partition in [26, 25, 7]. The partition $\mathcal{T}$ is a "mesh" for numerical integration and is completely different from the physical partition of u_N. Therefore, the partition $\mathcal{T}$ differs from meshes of traditional numerical methods. Nevertheless, the partition $\mathcal{T}$ and the corresponding quadrature $\mathcal{Q}_K$ are important for accuracy of the approximation u_N by providing accurate information of the exact solution.

The integrand in (3.9) has a derivative term $u_{\boldsymbol{\beta}}(\mathbf{x})$. Where u is differentiable, we have

$$u_{\boldsymbol{\beta}}(\mathbf{x}) = \sum_{i=1}^{d} \beta_i(\mathbf{x}) \frac{\partial u(\mathbf{x})}{\partial x_i}. \tag{3.10}$$

Obviously, (3.10) is invalid where the solution u is discontinuous, and hence any NN method such as the PINNs in [36] using discrete or autodifferentiation based on (3.10) would lead to an unreasonable approximation to a discontinuous solution. This phenomenon was already reported by several researchers, e.g., [11] for (1.1) with $\boldsymbol{\beta} = (1, 1)$ and [18] for scalar nonlinear HCLs. This essential difficulty was overcome by introduction of a physics preserved discrete differential operator: the discrete directional differentiation operator for (1.1) in [6] and the discrete divergence operator for nonlinear HCLs in [5]. For any $\mathbf{x} \in \Omega$, the discrete differential operator $D_{\boldsymbol{\beta}}$ is defined by

$$D_{\boldsymbol{\beta}} v(\mathbf{x}) := \frac{v(\mathbf{x}) - v\big(\mathbf{x} - \rho\bar{\boldsymbol{\beta}}(\ \mathbf{x})\big)}{\rho/|\boldsymbol{\beta}(\mathbf{x})|} \approx v_{\boldsymbol{\beta}}(\mathbf{x}), \tag{3.11}$$

where $|\boldsymbol{\beta}(\mathbf{x})|$ is the magnitude of $\boldsymbol{\beta}(\mathbf{x})$, $\bar{\boldsymbol{\beta}}(\mathbf{x}) = \frac{\boldsymbol{\beta}(\mathbf{x})}{|\boldsymbol{\beta}(\mathbf{x})|}$ is the unit vector along $\boldsymbol{\beta}(\mathbf{x})$, and $0 < \rho \ll 1$. That is, the directional derivative $v_{\boldsymbol{\beta}}$ in the $\boldsymbol{\beta}$ direction is approximated by the backward finite difference quotient with the "mesh" size $\rho/|\boldsymbol{\beta}(\mathbf{x})|$. Fundamentally, the discrete differentiation operator $D_{\boldsymbol{\beta}}$ ensures that the derivative is computed without crossing the discontinuous interface.

For each $E \in \mathcal{E}_- = \{E = \partial K \cap \Gamma_- : K \in \mathcal{T}\}$, let $\mathcal{Q}_E(w)$ denote a quadrature rule for integrand w defined on E. The discrete least-squares functional is defined by

$$\mathcal{L}_\tau\big(v;\ \mathbf{f}\big) = \sum_{K\in\mathcal{T}} \mathcal{Q}_K\left((D_{\boldsymbol{\beta}} v + \gamma v - f)^2\right) + \sum_{E\in\mathcal{E}_-} \mathcal{Q}_E\left(|\boldsymbol{\beta}\cdot\boldsymbol{n}|(v-g)^2\right). \tag{3.12}$$

Then the discrete least-squares NN approximation of problem (1.1) is to find $u_\tau^N \in \mathcal{M}(L,n)$ such that

$$\mathcal{L}_\tau\big(u_\tau^N;\mathbf{f}\big) = \min_{v\in\mathcal{M}(L,n)} \mathcal{L}_\tau\big(v;\mathbf{f}\big). \tag{3.13}$$

Remark 3.5. The discrete least-squares NN approximation defined in (3.13) enforces the inflow boundary condition through penalization. Instead, we may impose the inflow boundary condition through the discrete differentiation operator $D_{\boldsymbol{\beta}}$ defined in (3.11) by choosing proper ρ for integration point $\mathbf{x}$, that is, close to the inflow boundary, so that $\mathbf{x}-\rho\bar{\boldsymbol{\beta}}(\mathbf{x})$ belongs to $\mathcal{E}_-$. In this way, the boundary term of $\mathcal{L}_\tau\big(v;\mathbf{f}\big)$ in (3.12) may be dropped.

4. Error estimates. In this section, we provide error estimates for approximation by ReLU NN functions of the solution of the linear advection-reaction equation with a discontinuity 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 $d=2$, we 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 let $\mathbf{x}(s)$ 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*

$$\begin{aligned} &|u^+(\mathbf{x}(s)) - u^-(\mathbf{x}(s))| \\ &\quad = \exp\left(-\int_0^s \gamma(\mathbf{x}(t))\,dt\right) \\ &\qquad \times \left|\int_0^s \exp\left(\int_0^t \gamma(\mathbf{x}(r))\,dr\right)\left(f^+(\mathbf{x}(t)) - f^-(\mathbf{x}(t))\right)dt + g^+(\mathbf{x}_0) - g^-(\mathbf{x}_0)\right|, \end{aligned} \tag{4.2}$$

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 satisfy the linear ordinary differential equations

$$\begin{cases} \dfrac{d}{ds}u^\pm(\mathbf{x}(s)) + \gamma(\mathbf{x}(s))u^\pm(\mathbf{x}(s)) = f^\pm(\mathbf{x}(s)) & \text{for } s>0, \\ u^\pm(\mathbf{x}(0)) = u^\pm(\mathbf{x}_0) = g^\pm(\mathbf{x}_0) \end{cases} \tag{4.3}$$

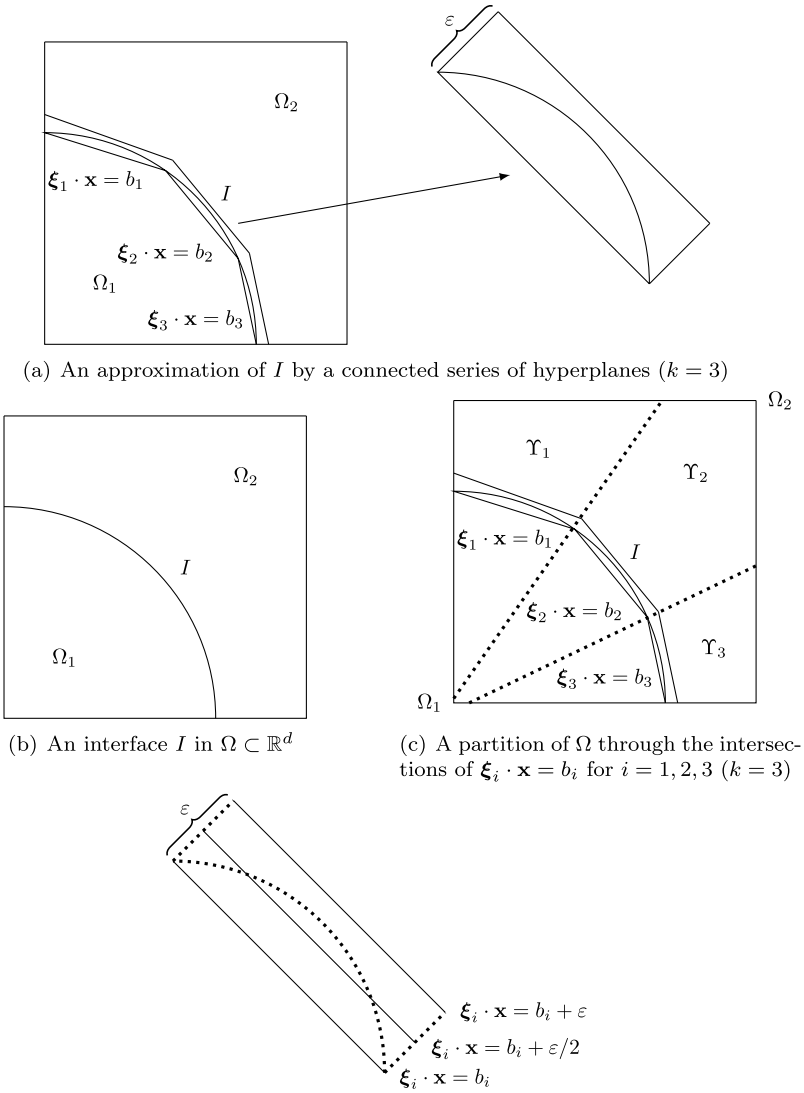
whose solutions are given by

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

Hence, u is discontinuous on I with the jump (4.2). This completes the proof of the lemma. ∎

Remark 4.2. For $d=3$, we assume that the inflow boundary data g is discontinuous along a curve $C(t) \subset \Gamma_-$. In this case, the collection of the streamlines $\mathbf{x}(s)$ 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 (4.2) for every $\mathbf{x}_0 = C(t)$.

Let the discontinuity interface I (in $\mathbb{R}^d$) as the union of streamlines of the vector field $\boldsymbol{\beta}$ as in Remark 4.2 divide the domain Ω into two nonempty subdomains Ω_1 and Ω_2 (see Figure 2(b) for $d=2$),



(a) An approximation of I by a connected series of hyperplanes ($k=3$)

(b) An interface I in $\Omega \subset \mathbb{R}^d$

(c) A partition of Ω through the intersections of $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$ for $i = 1, 2, 3$ ($k=3$)

(d) The hyperplanes $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$, $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i + \varepsilon$ from $p_i(\mathbf{x})$

FIG. 2. *An illustration of Lemma* 4.3.

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

so that the solution u is piecewise smooth with respect to the partition $\{\Omega_1, \Omega_2\}$. We assume that every streamline has a finite length.

Furthermore, We assume that the jump of the solution is constant. Hence u can be decomposed into

$$u(\mathbf{x}) = \hat{u}(\mathbf{x}) + \chi(\mathbf{x}), \tag{4.4}$$

where $\hat{u}$ is continuous and piecewise smooth on Ω, and $\chi(\mathbf{x})$ is the 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 $\alpha_2 = g^+(\mathbf{x}_0)$.

For a given $\varepsilon > 0$, we assume that the interface I can be approximated by a connected series of hyperplanes $\boldsymbol{\xi}_i \cdot \mathbf{x} - b_i = 0$ in Ω_1 for $i = 1, 2, \ldots, k$ such that $\boldsymbol{\xi}_i$ points toward Ω_2 and that the subdomain generated by translating $\boldsymbol{\xi}_i \cdot \mathbf{x} - b_i = 0$ in the direction of $\boldsymbol{\xi}_i$ by ε contains I (see Figure 2(a)). By normalizing $\boldsymbol{\xi}_i$, we may assume $|\boldsymbol{\xi}_i| = 1$. Obviously, the number of hyperplanes k depends on the interface I and the ε. Hence, for complicated interfaces and small ε, the k could be large.

We now divide Ω by hyperplanes passing through the intersections of $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$. Let Υ_i denote the subdomains determined by this process (see Figure 2(c), where two dotted lines divide the domain into three subdomains Υ_i for $i = 1, 2, 3$). For each i, let $|\mathcal{P}_i|$ denote the $d-1$ dimensional measure of the hyperplane $\boldsymbol{\xi}_i \cdot \mathbf{x} = b_i$ in Υ_i. With the approximation assumption on the I, clearly, there exists a positive constant C such that

$$\sum_{i=1}^{k} |\mathcal{P}_i| \le C\,|I|, \tag{4.5}$$

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

LEMMA 4.3. *Let $p(\mathbf{x})$ be the CPWL function (see Figure 2(d)) defined by*

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

For any sufficiently small $\varepsilon > 0$, there exists a positive constant C such that

$$|||\chi - p|||_{\boldsymbol{\beta}} \le C\,\big|\alpha_1 - \alpha_2\big|\,|I|^{1/2}\sqrt{\varepsilon}. \tag{4.6}$$

Proof. It is easy to check that

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

which, together with (4.5), implies

$$\|\chi - p\|^2_{0,\Omega} = \sum_{i=1}^{k} \|\chi - p_i\|^2_{0,\Upsilon_i} \le \sum_{i=1}^{k} |\mathcal{P}_i|(\alpha_1 - \alpha_2)^2\varepsilon \le C\,|I|(\alpha_1 - \alpha_2)^2\varepsilon. \tag{4.7}$$

We now prove

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

To this end, for each i, let

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

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

$$\begin{aligned}\|(p_i)_{\boldsymbol{\beta}}\|_{0,\Upsilon_i}^2 &= \|(p_i)_{\boldsymbol{\beta}} - (p_i)_{\boldsymbol{\beta}_i}\|_{0,\Upsilon_i}^2 = \|(p_i)_{\boldsymbol{\beta}-\boldsymbol{\beta}_i}\|_{0,\Upsilon_i}^2 = \|(p_i)_{\boldsymbol{\beta}-\boldsymbol{\beta}_i}\|_{0,\Upsilon_i^1}^2 \\ &\le \int_{\Upsilon_i^1} \left(\frac{\alpha_2 - \alpha_1}{\varepsilon} \boldsymbol{\xi}_i \cdot \varepsilon \boldsymbol{\xi}_i\right)^2 d\mathbf{x} \le (\alpha_1 - \alpha_2)^2 |\mathcal{P}_i| \varepsilon,\end{aligned}$$

where for the first inequality, we used the fact that on Υ_i^1, the gradient of p_i is $\frac{\alpha_2 - \alpha_1}{\varepsilon} \boldsymbol{\xi}_i$, and that the magnitude of $\boldsymbol{\beta}(\mathbf{x}) - \boldsymbol{\beta}_i(\mathbf{x})$ is less than or equal to $\varepsilon \boldsymbol{\xi}_i$. (Since $\boldsymbol{\beta}(\mathbf{x})$ is a tangent vector of a streamline, when $\varepsilon > 0$ is sufficiently small, the secant vector $\boldsymbol{\beta}_i(\mathbf{x})$ approaches $\boldsymbol{\beta}(\mathbf{x})$, i.e., $|\boldsymbol{\beta}(\mathbf{x}) - \boldsymbol{\beta}_i(\mathbf{x})| \le |\varepsilon \boldsymbol{\xi}_i|$.) Thus

$$\|\chi_{\boldsymbol{\beta}} - p_{\boldsymbol{\beta}}\|_{0,\Omega}^2 = \sum_{i=1}^k \|(p_i)_{\boldsymbol{\beta}}\|_{0,\Upsilon_i}^2 \le \sum_{i=1}^k |\mathcal{P}_i| (\alpha_1 - \alpha_2)^2 \varepsilon \le C\,|I| (\alpha_1 - \alpha_2)^2 \varepsilon. \tag{4.8}$$

Now (4.6) follows from (4.7) and (4.8). ∎

Remark 4.4. In Lemma 4.3, when there is a subdomain Υ_i either in Ω_1 or Ω_2 that does not contain any hyperplane $\boldsymbol{\chi}_i \cdot \mathbf{x} - b_i = 0$, we may define $p(\mathbf{x}) = \alpha_1$ or α_2 on Υ_i. Then $|||\chi - p|||_{\boldsymbol{\beta}} = 0$ on Υ_i. Hence, without loss of generality, we assumed that each Υ_i contains a hyperplane $\boldsymbol{\chi}_i \cdot \mathbf{x} - b_i = 0$ as in Figure 2(c).

THEOREM 4.5. *Let u and u_N be the solutions of problems* (3.2) *and* (3.6)*, respectively. Then we have*

$$|||u - u_N|||_{\boldsymbol{\beta}} \le C \left(|\alpha_1 - \alpha_2| \sqrt{\varepsilon} + \inf_{v \in \mathcal{M}(L,n)} |||\hat{u} + p - v|||_{\boldsymbol{\beta}} \right), \tag{4.9}$$

where $\hat{u} \in C(\Omega)$ and p are given in (4.4) *and Lemma* 4.3*, respectively. Moreover, if the depth of ReLU NN functions 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*

$$|||u - u_N|||_{\boldsymbol{\beta}} \le C \left(|\alpha_1 - \alpha_2| \sqrt{\varepsilon} + \inf_{v \in \mathcal{M}(\log, n - \hat{n})} |||\hat{u} - v|||_{\boldsymbol{\beta}} \right), \tag{4.10}$$

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

Proof. For any $v \in \mathcal{M}(L,n)$, it follows from (4.4), the triangle inequality, and Lemma 4.3 that

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

Taking the infimum over all $v \in \mathcal{M}(L,n)$, (4.9) is then a direct consequence of Lemma 3.3.

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

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

Obviously we have $v+p \in \mathcal{M}(\log, n)$ for any $v \in \mathcal{M}(\log, n-\hat{n})$. 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), (4.4), and the triangle inequality that

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

which, together with Lemma 4.3, implies the validity of (4.10). This completes the proof of the theorem. ∎

Remark 4.6. Theorem 4.5 mainly focused on the depth of NNs. A ReLU NN architecture with less than $\lceil \log_2(d+1) \rceil + 1$ layers, prescribed widths, exact weights and biases to approximate piecewise constant functions will be addressed in a forthcoming paper.

Remark 4.7. 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 $L=2$. For error estimates in the L^2 norm, see, e.g., [15, 30, 35]. Error estimates in a stronger norm like the $\left|\!\left|\!\left| \cdot \right|\!\right|\!\right|_{\boldsymbol{\beta}}$ or H^1 norm is possibly an open problem.

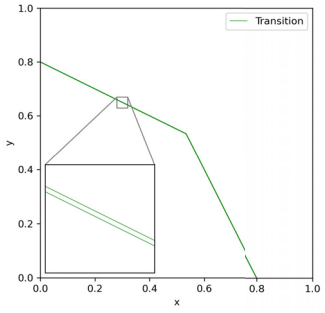
Remark 4.8. The estimate in (4.9) holds even for the shallow NN ($L=2$). However, the second term of the upper bound, $\inf_{v\in\mathcal{M}(2,n)} \left|\!\left|\!\left| \hat{u}+p-v \right|\!\right|\!\right|_{\boldsymbol{\beta}}$, depends on the inverse of ε (as well as the norm of the directional derivative of $\hat{u}+p-v$) because the p has a sharp transition layer of width ε. For any fixed n, $\inf_{v\in\mathcal{M}(2,n)} \|p-v\|_\infty$ could be large depending on the size of ε, even though the universal approximation theorem implies

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

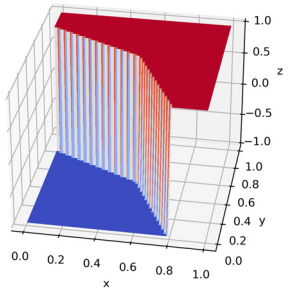
Moreover, as ε approaches 0, p approaches χ, which is discontinuous; hence, in practice, the universal approximation theorem does not guarantee the convergence of the problem. On the other hand, by Proposition 2.1, deeper networks (with depth at least $\lceil \log_2(d+1) \rceil + 1$) are capable of approximating such functions. As an illustration of this, we define a CPWL function $p(x,y)$ with a sharp transition layer of width $4\varepsilon/\sqrt{5}$ on $[0,1]^2$ as follows (see Figure 3(b)):

$$p(x,y) = \begin{cases} -1+\dfrac{1}{\varepsilon}\left(y+\dfrac{1}{2}x-\dfrac{4}{5}+\varepsilon\right) & \text{if } y \ge x,\ y > -\dfrac{1}{2}x+\dfrac{4}{5}-\varepsilon,\ y \le -\dfrac{1}{2}x+\dfrac{4}{5}+\varepsilon,\\ -1+\dfrac{1}{\varepsilon}\left(\dfrac{1}{2}y+x-\dfrac{4}{5}+\varepsilon\right) & \text{if } y < x,\ y > -2(x+\varepsilon)+\dfrac{8}{5},\ y \le -2(x-\varepsilon)+\dfrac{8}{5},\\ -1 & \text{if } y \le -\dfrac{1}{2}x+\dfrac{4}{5}-\varepsilon,\ y \le -2(x+\varepsilon)+\dfrac{8}{5},\\ 1 & \text{otherwise.}\end{cases}$$

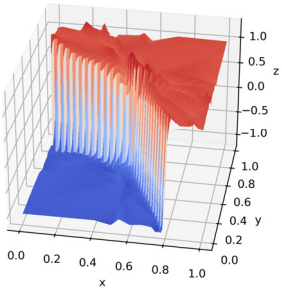
In [6], we proved CPWL functions of the form $p(x,y)$ are 2–4–4–1 ReLU NN functions. Here and in what follows, by a d–n_1– $\cdots$ –n_{L-1}–c ReLU NN function, we mean a ReLU NN function from $\mathbb{R}^d$ to $\mathbb{R}^c$ with an L-layer representation with width n_l of the lth hidden layer of the representation for $l=1,\ldots,L-1$. Therefore, we can expect 2–n_1–n_2–1 (depth $3=\lceil \log_2(d+1) \rceil+1$ for $d=2$) ReLU NN functions outperform 2–n_3–1 ReLU NN functions, and Figure 3 and Table 1 demonstrate this for approximating $p(x,y)$ with $\varepsilon = 0.1, 0.01, 0.001$ by ReLU NN functions with depth 2, 3 and various



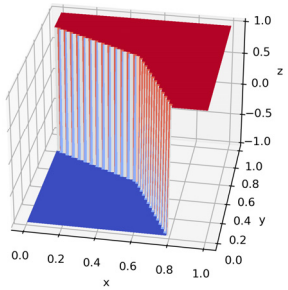
(a) The location of the transition layer



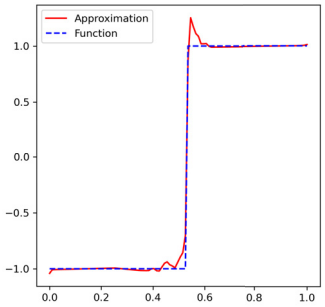
(b) $p(x, y)$ with $\varepsilon = 0.001$



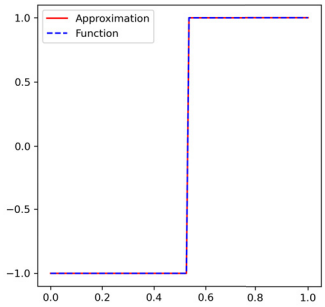
(c) A 2–158–1 ReLU NN function approximation



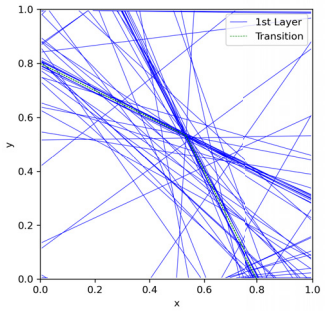
(d) A 2–4–4–1 ReLU NN function approximation



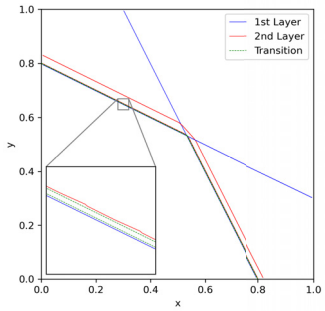
(e) The trace of Figure 3(c) on $y = x$



(f) The trace of Figure 3(d) on $y = x$



(g) The breaking hyperplanes of the approximation in Figure 3(c)



(h) The breaking hyperplanes of the approximation in Figure 3(d)

FIG. 3. *L^2 norm approximation results of $p(x, y)$ with $\varepsilon = 0.001$.*

TABLE 1
Relative errors in the L^2 norm for approximating $p(x,y)$ with $\varepsilon = 0.1, 0.01, 0.001$ 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
$\varepsilon = 0.1$	0.292446	0.028176	0.021200	0.010259	1.906427×10^{-7}
$\varepsilon = 0.01$	0.254603	0.078549	0.065465	0.030623	7.536140×10^{-6}
$\varepsilon = 0.001$	0.404299	0.102757	0.100136	0.088885	9.473783×10^{-7}

numbers of neurons where we minimized the squared L^2 norm loss function (the midpoint rule on a uniform mesh with mesh size $h = 10^{-2}$) using the Adam optimization algorithm for 100000 iterations with the learning rate 0.004. Even though the relative errors for the one-hidden-layer ReLU NN function approximations decrease as the number of neurons increases (Table 1), the graphs (Figures 3(c) and 3(e)) exhibit oscillations near the location where the transition layer is formed (Figure 3(a)). In contrast, the two-hidden-layer ReLU NN functions approximate the target function well (Figures 3(b), 3(d), and 3(f)). The breaking hyperplanes (Figures 3(g) and 3(h)) show where the transition layers are formed for both approximations. In particular, if we zoom in on Figure 3(h), breaking hyperplanes are right around the location of the transition layer (two green dotted lines with width $4\varepsilon/\sqrt{5}$), which implies the two almost vertical planes in Figure 3(d).

Finally, we also note that $\mathcal{M}(2) \subsetneq \mathcal{M}(\lceil \log_2(d+1) \rceil + 1)$ when $d \geq 2$ (see [14]).

LEMMA 4.9. *Let u, $u_{\scriptscriptstyle N}$, and $u_{\mathcal{T}}^{\scriptscriptstyle N}$ be the solutions of problems (3.2), (3.6), and (3.13), respectively. Then there exist positive constants C_1 and C_2 such that*

$$\begin{aligned} \left|\!\left|\!\left| u - u_{\mathcal{T}}^{\scriptscriptstyle N} \right|\!\right|\!\right|_{\boldsymbol{\beta}} \leq\, & C_1 \left(\left| (\mathcal{L} - \mathcal{L}_{\mathcal{T}})(u_{\scriptscriptstyle N} - u_{\mathcal{T}}^{\scriptscriptstyle N}, \mathbf{0}) \right| + \left| (\mathcal{L} - \mathcal{L}_{\mathcal{T}})(u - u_{\scriptscriptstyle N}, \mathbf{0}) \right| \right)^{1/2} \\ & + C_2 \left(\left| \alpha_1 - \alpha_2 \right| \sqrt{\varepsilon} + \inf_{v \in \mathcal{M}(L,n)} \left|\!\left|\!\left| \hat{u} + p - v \right|\!\right|\!\right|_{\boldsymbol{\beta}} \right). \end{aligned} \tag{4.11}$$

Proof. By the triangle inequality

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

and the fact that

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

from the proof of Lemma 3.4 in [6], (4.11) is a direct consequence of Theorem 4.5. ∎

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. All numerical experiments have rectangular domains, and we used numerical integration (the midpoint rule) to implement the scheme in (3.6) (see [6]). Numerical integration used a uniform mesh with mesh size $h = 10^{-2}$. In (3.11), we set $\rho = h/4$ (except for the last test problem, which used $\rho = h/12$). We used the Adam optimization algorithm [23] to iteratively solve the discrete minimization problem (3.13). 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 NN 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 9, parameters indicate the total number of weights and biases, and $1/2$ in $\mathcal{L}^{1/2}$ for the relative error in the least-squares functional indicates the square root. We employed ReLU NN functions with width n and depth $3 = \lceil \log_2(d+1) \rceil + 1$ for $d = 2, 3$. 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 NN method in [26, 7].

5.1. Two-dimensional problems. We present numerical results for five two-dimensional test problems with piecewise constant or variable advection velocity fields. The fifth test problem compares the LSNN method to other relevant methods. All five test problems are defined on the domain $\Omega = (0,1)^2$ with $\gamma = 1$ (except for the fifth test problem with $\gamma = 0.1$), and the exact solutions are the same as the right-hand-side functions, $u(x,y) = f(x,y)$, which are step functions (except for the fourth and fifth test problems) along a 3-line segment, a 4-line segment, or curved interfaces. By Theorem 4.5, the LSNN method with 3-layer ReLU NN functions leads to

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

because the continuous part of the solution $\hat{u}$ in (4.4) is zero (again, except for the fourth and fifth test problems).

5.1.1. A problem with a 3-line segment interface. This example is a modification of one from [6]. Let $\bar{\Omega} = \bar{\Upsilon}_1 \cup \bar{\Upsilon}_2 \cup \bar{\Upsilon}_3$ and

$$\Upsilon_1 = \{(x,y) \in \Omega : y \ge x\},\ \Upsilon_2 = \left\{(x,y) \in \Omega : x - \frac{a}{2} \le y < x\right\},\ \text{and}$$
$$\Upsilon_3 = \left\{(x,y) \in \Omega : y < x - \frac{a}{2}\right\}$$

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

$$\boldsymbol{\beta}(x,y) = \begin{cases} (-1, \sqrt{2}-1)^T, & (x,y) \in \Upsilon_1, \\ (1-\sqrt{2}, 1)^T, & (x,y) \in \Upsilon_2, \\ (-1, \sqrt{2}-1)^T, & (x,y) \in \Upsilon_3. \end{cases} \tag{5.1}$$

The inflow boundary and the inflow boundary condition are given by

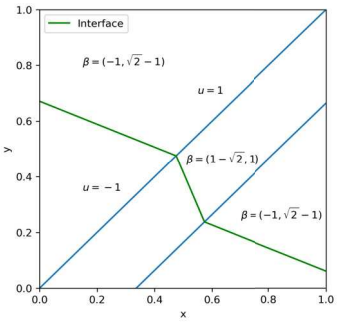
$$\Gamma_- = \{(1,y) : y \in (0,1)\} \cup \{(x,0) : x \in (0,1)\}$$
$$\text{and}\ \ g(x,y) = \begin{cases} 1, & (x,y) \in \Gamma_-^1 \equiv \left\{(1,y) : y \in \left[1 - \sqrt{2} + \frac{\sqrt{2}}{2}a, 1\right)\right\}, \\ -1, & (x,y) \in \Gamma_-^2 = \Gamma_- \setminus \Gamma_-^1, \end{cases}$$

respectively. Let

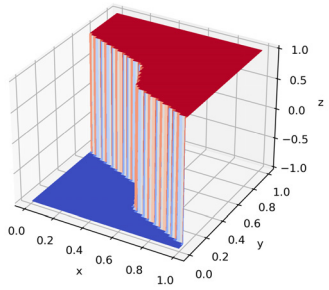
$$\hat{\Upsilon}_1 = \left\{(x,y) \in \Upsilon_1 : y < (1-\sqrt{2})x + a\right\},$$
$$\hat{\Upsilon}_2 = \left\{(x,y) \in \Upsilon_2 : y < \frac{1}{1-\sqrt{2}}\left(x - \frac{a}{\sqrt{2}}\right) + \frac{a}{\sqrt{2}}\right\},$$
$$\text{and}\ \ \hat{\Upsilon}_3 = \left\{(x,y) \in \Upsilon_3 : y < \left(1-\sqrt{2}\right)x + \frac{\sqrt{2}}{2}a\right\}.$$

The following right-hand-side function is (see Figure 4(b))

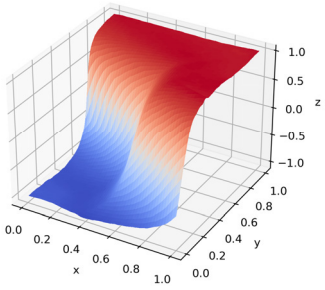
$$f(x,y) = \begin{cases} -1, & (x,y) \in \Omega_1 \equiv \hat{\Upsilon}_1 \cup \hat{\Upsilon}_2 \cup \hat{\Upsilon}_3, \\ 1, & (x,y) \in \Omega_2 = \Omega \setminus \Omega_1. \end{cases} \tag{5.2}$$



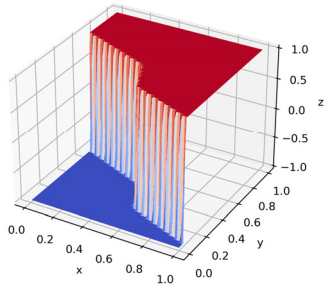
(a) The interface



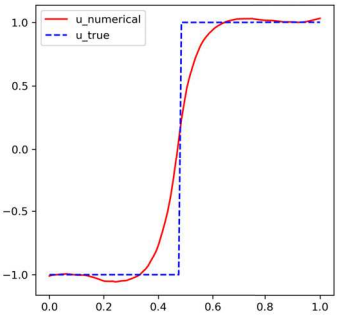
(b) The exact solution



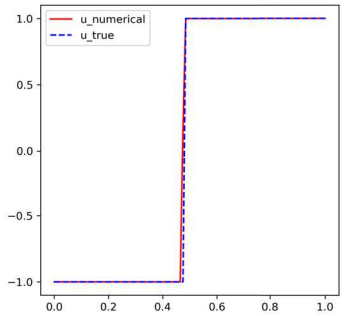
(c) A 2–300–1 ReLU NN function approximation



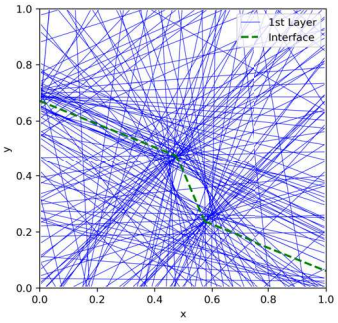
(d) A 2–5–5–1 ReLU NN function approximation



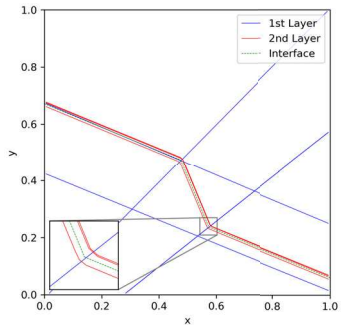
(e) The trace of Figure 4(c) on $y = x$



(f) The trace of Figure 4(d) on $y = x$



(g) The breaking hyperplanes of the approximation in Figure 4(c)



(h) The breaking hyperplanes of the approximation in Figure 4(d)

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

200000 iterations were 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 errors (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 discontinuity interface (Figure 4(a)), although the breaking hyperplanes (Figure 4(g)) indicate that the approximation roughly formed the transition layer 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 4.8. 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 $\mathcal{M}(3, 10)$ is indeed such a function. The second-layer breaking hyperplanes (Figure 4(h)) also help us to verify that a sharp transition layer was generated along the discontinuity 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. A 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

$$\Upsilon_1 = \{(x,y) \in \Omega : \ y \geq x+1\}, \ \Upsilon_2 = \{(x,y) \in \Omega, \ x \leq y < x+1\},$$
$$\Upsilon_3 = \{(x,y) \in \Omega, \ x-1 \leq y < x\}, \text{ and } \Upsilon_4 = \{(x,y) \in \Omega, \ y < x-1\}.$$

The advective velocity field is a piecewise constant field given by

$$\boldsymbol{\beta}(x,y) = \begin{cases} (-1, \sqrt{2}-1)^T, & (x,y) \in \Upsilon_1, \\ (1-\sqrt{2}, 1)^T, & (x,y) \in \Upsilon_2, \\ (-1, \sqrt{2}-1)^T, & (x,y) \in \Upsilon_3, \\ (1-\sqrt{2}, 1)^T, & (x,y) \in \Upsilon_4. \end{cases} \tag{5.3}$$

The inflow boundary and the inflow boundary condition are given by

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

respectively. Let

$$\hat{\Upsilon}_1 = \{(x,y) \in \Upsilon_1 : y < (1-\sqrt{2})x+2\}, \ \hat{\Upsilon}_2 = \left\{(x,y) \in \Upsilon_2 : y < \frac{1}{1-\sqrt{2}}(x-1)+1\right\},$$
$$\hat{\Upsilon}_3 = \{(x,y) \in \Upsilon_3 : y < (1-\sqrt{2})(x-1)+1\},$$
$$\text{and } \hat{\Upsilon}_4 = \left\{(x,y) \in \Upsilon_3 : y < \frac{1}{1-\sqrt{2}}x + \frac{2}{\sqrt{2}-1}\right\}.$$

The following right-hand-side function is (see Figure 5(b))

$$f(x,y)=\begin{cases}-1, & (x,y)\in\Omega_1\equiv\hat{\Upsilon}_1\cup\hat{\Upsilon}_2\cup\hat{\Upsilon}_3\cup\hat{\Upsilon}_4,\\ 1, & (x,y)\in\Omega_2=\Omega\setminus\Omega_1.\end{cases} \tag{5.4}$$

200000 iterations were 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 $\mathcal{M}(3,12)$ 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 hyperplanes (Figure 5(h)) along the discontinuity interface (Figure 5(a)) show where a sharp transition layer was 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. A problem with a curved interface. The advective velocity field is the variable field given by

$$\boldsymbol{\beta}(x,y)=(1,2x),\ (x,y)\in\Omega. \tag{5.5}$$

The inflow boundary and the inflow boundary condition are given by

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

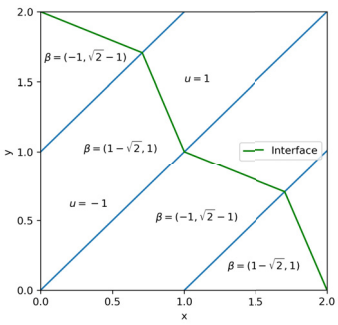
respectively. The following right-hand-side function is (see Figure 6(b))

$$f(x,y)=\begin{cases}0, & (x,y)\in\Omega_1\equiv\left\{(x,y)\in\Omega:y<x^2+\frac{1}{8}\right\},\\ 1, & (x,y)\in\Omega_2=\Omega\setminus\Omega_1.\end{cases} \tag{5.6}$$

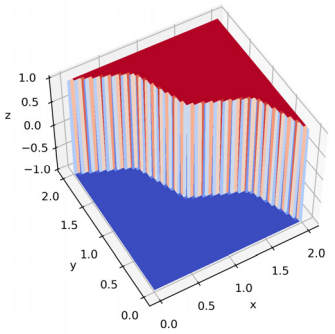
300000 iterations were implemented with 2–3000–1 and 2–60–60–1 ReLU NN functions. The numerical results are presented in Figure 6 and Table 4. We again increased the number of hidden neurons for the 3-layer network structure, assuming CPWL functions approximating a curved discontinuity interface (Figure 6(a)) well would be a ReLU NN function with more hidden neurons. 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 discontinuity 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 discontinuity interface (Figure 6(h)).

5.1.4. A problem with a curved interface and $\hat{u}\neq 0$ in (4.4). The advective velocity field is a variable field given by

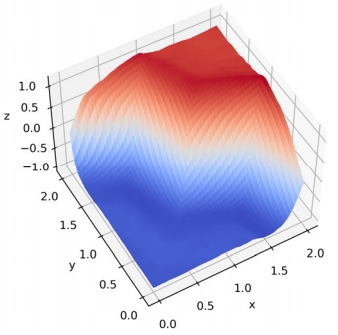
$$\boldsymbol{\beta}(x,y)=(-y,x),\ (x,y)\in\Omega. \tag{5.7}$$



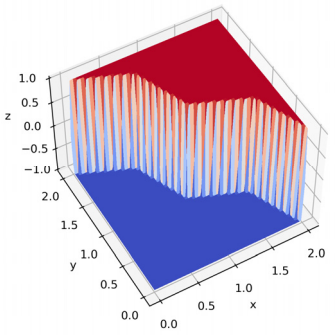
(a) The interface



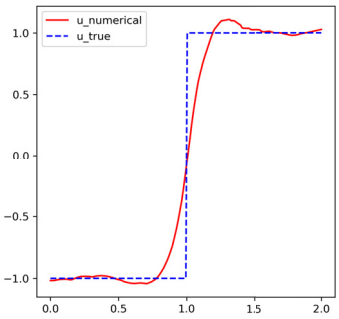
(b) The exact solution



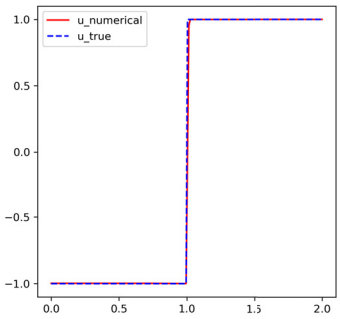
(c) A 2–300–1 ReLU NN function approximation



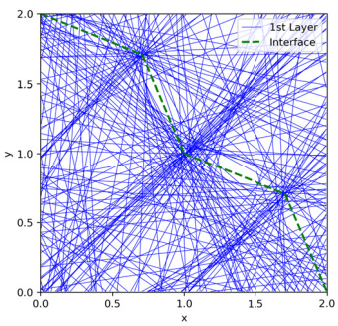
(d) A 2–6–6–1 ReLU NN function approximation



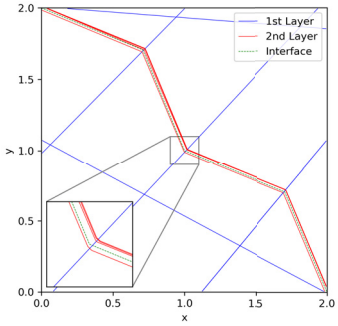
(e) The trace of Figure 5(c) on $y = x$



(f) The trace of Figure 5(d) on $y = x$



(g) The breaking hyperplanes of the approximation in Figure 5(c)



(h) The breaking hyperplanes of the approximation in Figure 5(d)

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

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

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

respectively. The following right-hand-side function is (see Figure 7(b))

$$f(x,y) = \begin{cases} -1 + x^2 + y^2, & (x,y) \in \Omega_1 \equiv \left\{(x,y) \in \Omega : y < \sqrt{\frac{4}{9} - x^2}\right\}, \\ 1 + x^2 + y^2, & (x,y) \in \Omega_2 = \Omega \setminus \Omega_1. \end{cases} \tag{5.8}$$

200000 iterations were 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)) on average, but Figures 7(c), 7(e), and 7(g) show difficulty around the discontinuity 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 pointwisely (Figures 7(d) and 7(f)). Unlike other examples, some of the second-layer breaking hyperplanes (Figure 7(h)) of the approximation spread out on the whole domain in addition to those around the interface, which implies that they are necessary for approximating the solution with $\hat{u} \neq 0$ ($\hat{u} = x^2 + y^2$ in this example).

5.1.5. A problem with a sharp transition layer. The advective velocity field is a variable field given by

$$\boldsymbol{\beta}(x,y) = (y+1, -x)/\sqrt{x^2 + (y+1)^2}, \ (x,y) \in \Omega, \tag{5.9}$$

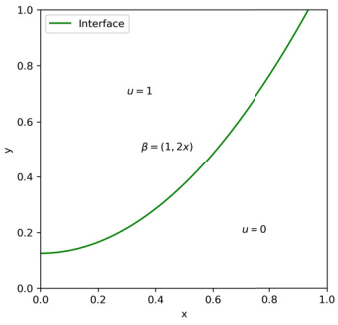
the inflow boundary is given by

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

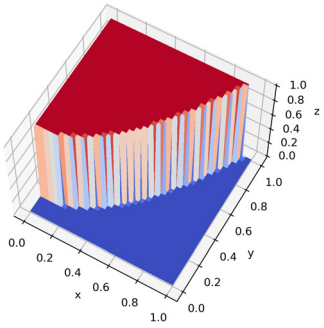
and $f = 0$. We choose an inflow boundary condition g such that the exact solution is (see Figure 8(b))

$$u(x,y) = \frac{1}{4} \exp\left(\gamma r \arcsin\left(\frac{y+1}{r}\right)\right) \arctan\left(\frac{r - 1.5}{\varepsilon}\right) \ \text{with } r = \sqrt{x^2 + (y+1)^2}. \tag{5.10}$$

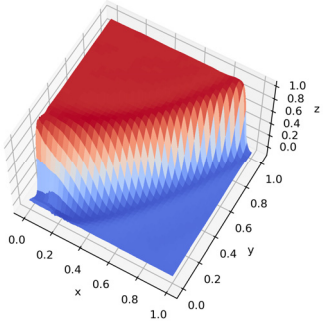
300000 iterations were implemented with 2–70–70–1 ReLU NN functions to approximate u with $\varepsilon = 10^{-10}$ in (5.10). The numerical results are presented in Figure 8 and Table 6. The approximation results are similar to those of the previous examples.



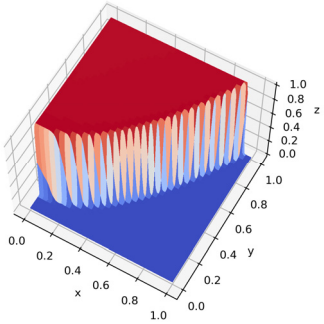
(a) The interface



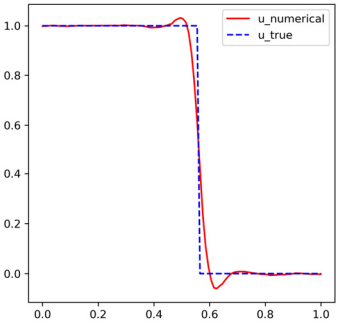
(b) The exact solution



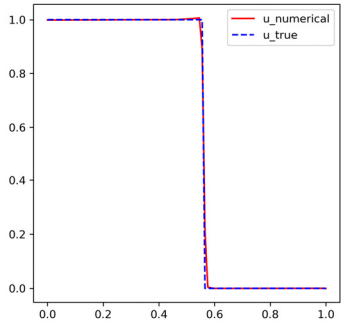
(c) A 2–3000–1 ReLU NN function approximation



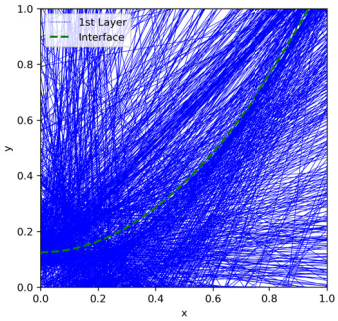
(d) A 2–60–60–1 ReLU NN function approximation



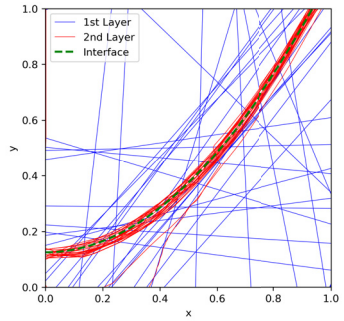
(e) The trace of Figure 6(c) on $y = 1 - x$



(f) The trace of Figure 6(d) on $y = 1 - x$



(g) The breaking hyperplanes of the approximation in Figure 6(c)



(h) The breaking hyperplanes of the approximation in Figure 6(d)

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

TABLE 4
Relative errors of the problem in subsection 5.1.3.

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

The same PDE was solved in [28, 29, 41], and in the experiments, the layer cannot be fully resolved and should be viewed as discontinuous (see Figure 8(b)). The L^2 errors of the least-squares finite element methods in [28, 29] are approximately between 4×10^{-2} and 6×10^{-2} with 10^4 to 10^6 degrees of freedom, whereas the error by the LSNN method is approximately 3×10^{-2} with 5251 parameters (Table 6). The discontinuous Galerkin finite element methods (DGFEMs) in [41] give similar results with a P0-DGFEM solution having no overshoot and a P1-DGFEM solution having a nontrivial overshoot. There is no overshoot from the LSNN method (Figures 8(a) and 8(c)).

5.2. Three-dimensional problems. We present numerical results for three three-dimensional test problems with piecewise constant or variable advection velocity fields whose solutions are piecewise constant along a connected series of planes or a surface. All three test problems are defined on the domain $\Omega=(0,1)^3$, approximation results are depicted on $z=0.505$ (except for the last test problem on $z=0.205$), and again, as in the experiments for $d=2$, we have

$$|||u-u_N|||_{\boldsymbol{\beta}} \le C\,|\alpha_1-\alpha_2|\,\sqrt{\varepsilon}.$$

5.2.1. A problem with a 2-plane segment interface. Let $\gamma=f=0$, $\bar\Omega=\bar\Upsilon_1\cup\bar\Upsilon_2$, and

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

The advective velocity field is a piecewise constant field given by

$$\boldsymbol{\beta}(x,y,z)=\begin{cases}(1-\sqrt{2},1,0)^T, & (x,y,z)\in\Upsilon_1,\\ (-1,\sqrt{2}-1,0)^T, & (x,y,z)\in\Upsilon_2.\end{cases} \tag{5.11}$$

The inflow boundary and the inflow boundary condition are given by

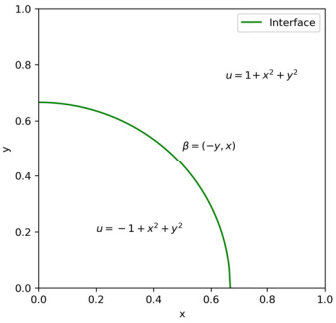
$$\Gamma_-=\{(x,0,z): x,z\in(0,1)\}\cup\{(1,y,z): y,z\in(0,1)\}$$
$$\text{and}\quad g(x,y,z)=\begin{cases}0, & (x,y,z)\in\Gamma_-^1\equiv\{(x,0,z): x\in(0,0.7),\ z\in(0,1)\},\\ 1, & (x,y,z)\in\Gamma_-^2=\Gamma_-\setminus\Gamma_-^1,\end{cases}$$

respectively. Let

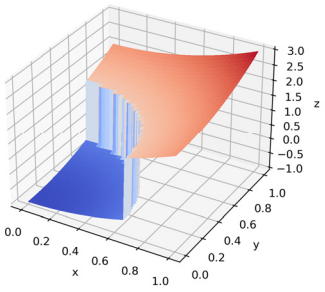
$$\Omega_1=\left\{(x,y,z)\in\Omega: y<(1-\sqrt{2})x+0.7,\ y<\frac{1}{1-\sqrt{2}}(x-0.7)\right\}.$$

The exact solution is a unit step function in three dimensions (see Figure 9(b)),

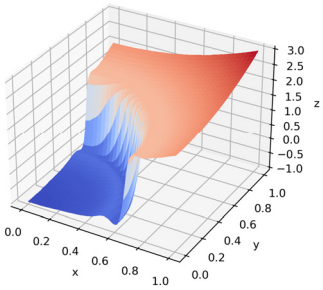
$$u(x,y,z)=\begin{cases}0, & (x,y,z)\in\Omega_1,\\ 1, & (x,y,z)\in\Omega_2=\Omega\setminus\Omega_1.\end{cases} \tag{5.12}$$



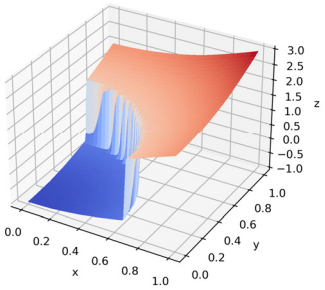
(a) The interface



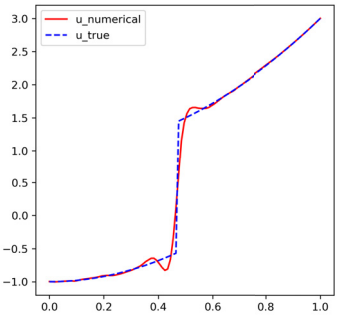
(b) The exact solution



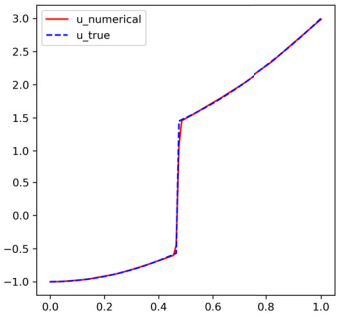
(c) A 2–4000–1 ReLU NN function approximation



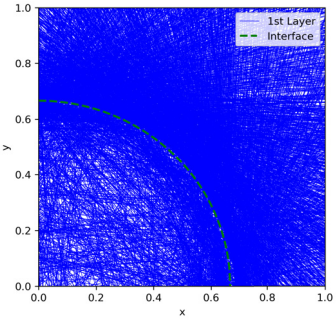
(d) A 2–65–65–1 ReLU NN function approximation



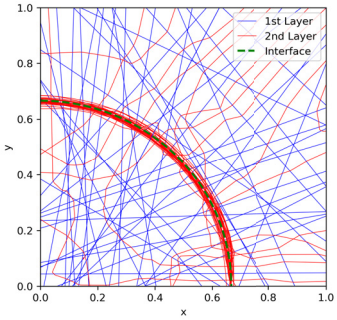
(e) The trace of Figure 7(c) on $y = x$



(f) The trace of Figure 7(d) on $y = x$

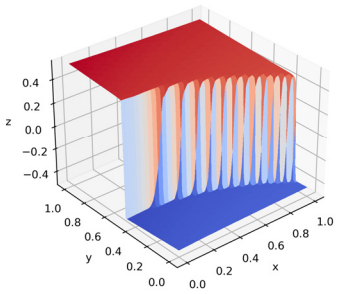


(g) The breaking hyperplanes of the approximation in Figure 7(c)

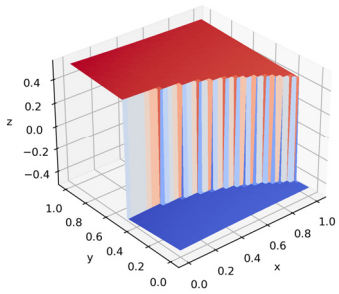


(h) The breaking hyperplanes of the approximation in Figure 7(d)

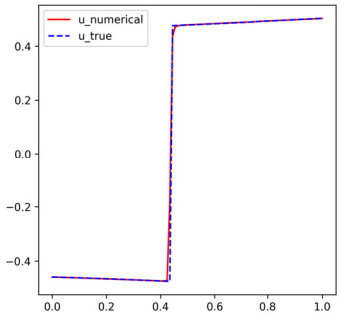
FIG. 7. *Approximation results of the problem in subsection* 5.1.4.



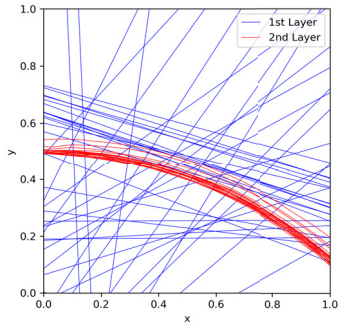
(a) A 2–70–70–1 ReLU NN function approximation



(b) The exact solution with $\varepsilon = 10^{-10}$ in (5.10)



(c) The trace of Figure 8(a) on $y = x$



(d) The breaking hyperplanes of the approximation in Figure 8(a)

FIG. 8. *Approximation results of the problem in subsection* 5.1.5.

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

TABLE 6
Relative errors of the problem in subsection 5.1.5.

Network structure	$\|u-u_{\mathcal{T}}^N\|_0$	$\frac{\|u-u_{\mathcal{T}}^N\|_0}{\|u\|_0}$	$\frac{\|\|\|u-u_{\mathcal{T}}^N\|\|\|_{\boldsymbol{\beta}}}{\|\|\|u\|\|\|_{\boldsymbol{\beta}}}$	$\frac{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{f})}{\mathcal{L}^{1/2}(u_{\mathcal{T}}^N,\mathbf{0})}$	Parameters
2–70–70–1	0.032229	0.066337	0.069926	0.023664	5251

100000 iterations were implemented with 3–300–1 and 3–5–5–1 ReLU NN functions (depth $\lceil \log_2(d+1) \rceil + 1 = 3$ for $d = 3$). The numerical results are presented in Figure 9 and Table 7. For three dimensions again, the 2-layer network structure with a large number of parameters generated a transition layer along the discontinuity interface (Figures 9(a) and 9(g)) but had trouble approximating the solution (Figure 9(b)) accurately pointwise (Figures 9(c) and 9(e)). The 3-layer network structure with 4% of the number of parameters of the 2-layer network approximated the solution accurately (Table 7). As explained in Example 5.1.1, Figures 9(d), 9(f), and 9(h) also

TABLE 7
Relative errors of the problem in subsection 5.2.1.

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

indicate that the function p in Lemma 4.3 appears to be the approximation and in this example to be contained in $\mathcal{M}(3,10)$.

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

$$\boldsymbol{\beta}(x,y,z) = (-y,x,0)^T,\ (x,y,z) \in \Omega. \tag{5.13}$$

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

$$f(x,y,z) = \begin{cases} 0, & (x,y,z) \in \Omega_1, \\ 1, & (x,y,z) \in \Omega_2 = \Omega \setminus \Omega_1. \end{cases} \tag{5.14}$$

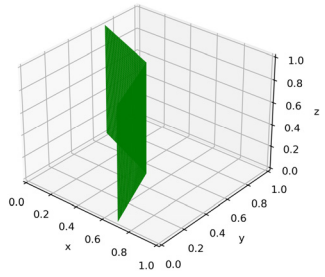
The exact solution is (see Figure 10(b))

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

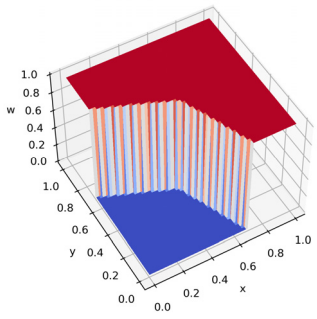
150000 iterations were implemented with 3–1500–1 and 3–50–50–1 ReLU NN functions. The numerical results are presented in Figure 10 and Table 8. Again to minimize the loss function over a larger subset of CPWL functions, we increased the number of hidden neurons. Even though the 2-layer ReLU NN function approximation provides an approximate location of the discontinuity interface (Figures 10(a) and 10(g)), the structure failed to approximate the solution (Figure 10(b)) around the interface accurately (Figures 10(c) and 10(e)). The 3-layer network structure with less than 40% of the number of parameters of the 2-layer network approximated the solution well, locating the discontinuity interface (Figures 10(d), 10(f), and 10(h) and Table 8).

5.2.3. A problem with a spherical interface. Let $\gamma = 1$. The advective velocity field is a variable field given by

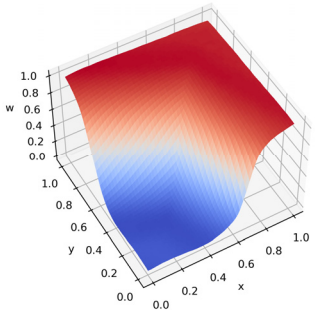
$$\boldsymbol{\beta}(x,y,z) = (-y-z,x,x)^T,\ (x,y,z) \in \Omega. \tag{5.15}$$



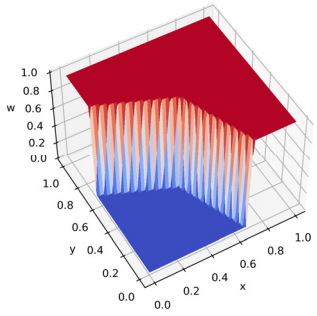
(a) The interface



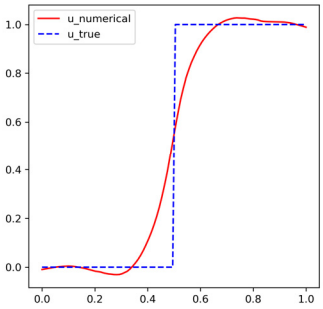
(b) The exact solution



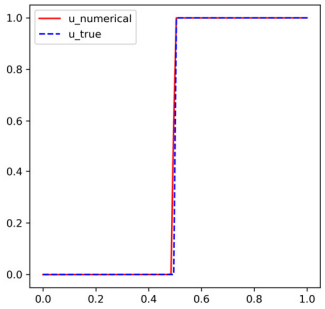
(c) A 3–300–1 ReLU NN function approximation



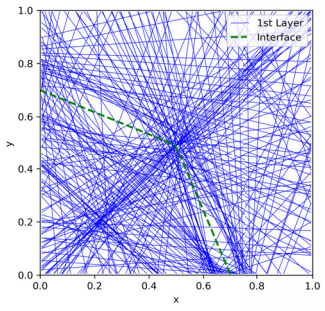
(d) A 3–5–5–1 ReLU NN function approximation



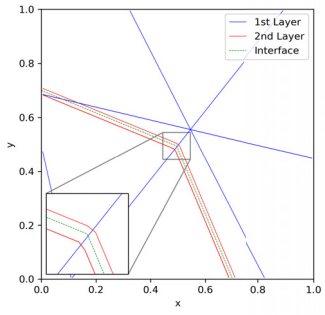
(e) The trace of Figure 9(c) on $y = x$



(f) The trace of Figure 9(d) on $y = x$

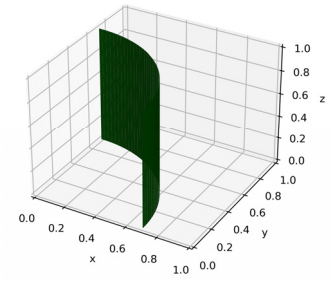


(g) The breaking hyperplanes of the approximation in Figure 9(c)

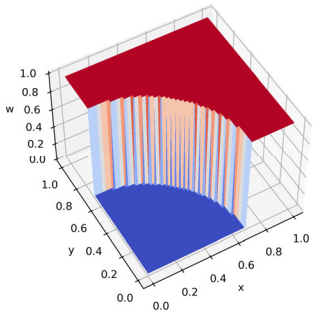


(h) The breaking hyperplanes of the approximation in Figure 9(d)

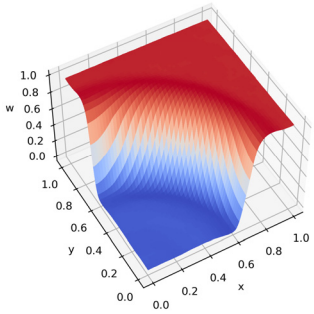
FIG. 9. *Approximation results of the problem in subsection* 5.2.1.



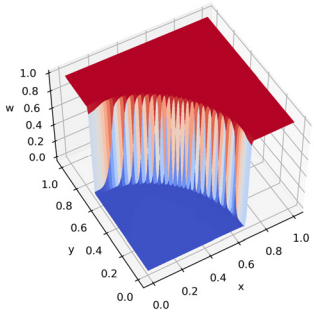
(a) The interface



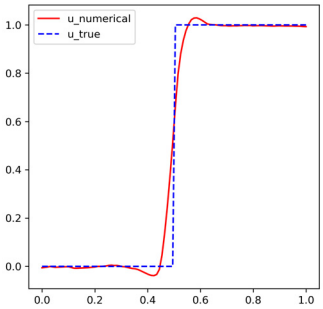
(b) The exact solution



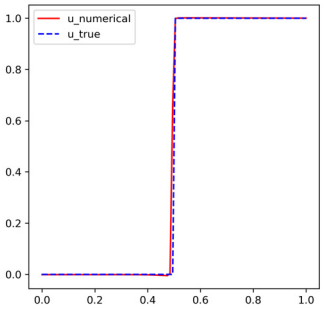
(c) A 3–1500–1 ReLU NN function approximation



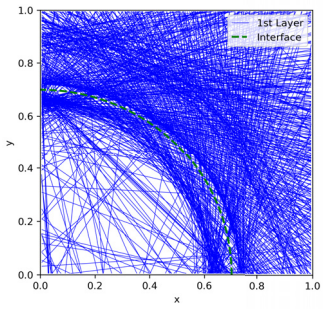
(d) A 3–50–50–1 ReLU NN function approximation



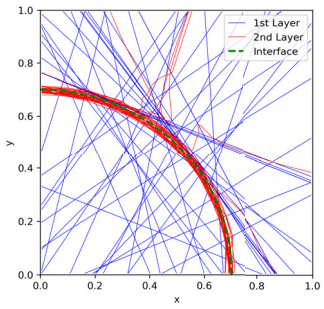
(e) The trace of Figure 10(c) on $y = x$



(f) The trace of Figure 10(d) on $y = x$



(g) The breaking hyperplanes of the approximation in Figure 10(c)



(h) The breaking hyperplanes of the approximation in Figure 10(d)

FIG. 10. *Approximation results of the problem in subsection* 5.2.2.

TABLE 8
Relative errors of the problem in subsection 5.2.2.

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

TABLE 9
Relative errors of the problem in subsection 5.2.3.

Network structure	$\frac{\|u-u_T^N\|_0}{\|u\|_0}$	$\frac{\|\|\|u-u_T^N\|\|\|_{\boldsymbol{\beta}}}{\|\|\|u\|\|\|_{\boldsymbol{\beta}}}$	$\frac{\mathcal{L}^{1/2}(u_T^N,\mathbf{f})}{\mathcal{L}^{1/2}(u_T^N,\mathbf{0})}$	Parameters
3–1376–1	0.113045	0.144105	0.106094	6881
3–80–80–1	0.042233	0.064332	0.041935	6881

The inflow boundary and the inflow boundary condition are given by

$$\begin{aligned}\Gamma_- &= \{(x,0,z) : x,z \in (0,1)\} \cup \{(1,y,z) : y,z \in (0,1)\} \\ &\quad \cup \{(x,y,0) : x,y \in (0,1)\}\end{aligned}$$

$$\text{and} \quad g(x,y,z) = \begin{cases} 0, & (x,y,z) \in \Gamma_-^1 \equiv \left\{(x,0,z) : 0 < z < \sqrt{0.7^2 - x^2},\ x \in (0,0.7)\right\}, \\ 0, & (x,y,z) \in \Gamma_-^2 \equiv \left\{(x,y,0) : 0 < y < \sqrt{0.7^2 - x^2},\ x \in (0,0.7)\right\}, \\ 1, & (x,y,z) \in \Gamma_-^3 = \Gamma_- \setminus (\Gamma_-^1 \cup \Gamma_-^2), \end{cases}$$

respectively. Let

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

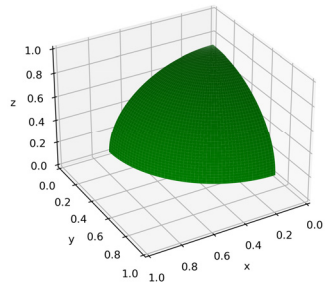
The following right-hand-side function is

$$f(x,y,z) = \begin{cases} 0, & (x,y,z) \in \Omega_1, \\ 1, & (x,y,z) \in \Omega_2 = \Omega \setminus \Omega_1. \end{cases} \tag{5.16}$$

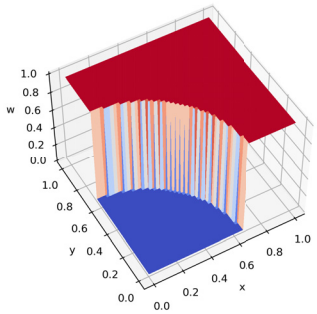
The exact solution is (see Figure 11(b))

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

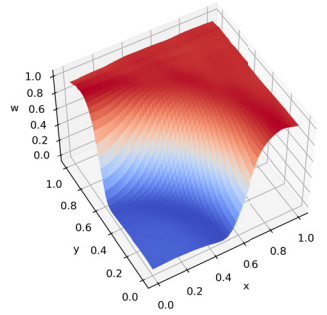
200000 iterations were implemented with 3–1376–1 and 3–80–80–1 ReLU NN functions. The numerical results are presented in Figure 11 and Table 9. We increased the number of hidden neurons and ρ was set to $h/12$ in the finite difference quotient in (3.11) because of the jump along the more curved interface (Figure 11(a)). Unlike the other two three-dimensional test problems, in this test problem, the third dimension actually plays a role as we can see from the advective velocity field (5.15). The approximation results are similar to those of the previous examples. Moreover, we note that the two-hidden-layer NN outperforms the one-hidden-layer NN with the same number of parameters, which together with the previous examples suggests that the one-hidden-layer NN needs more neurons.



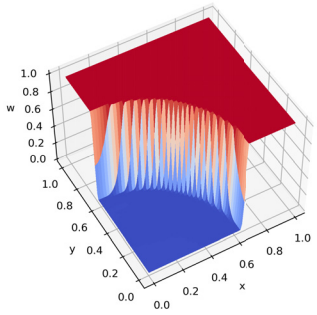
(a) The interface



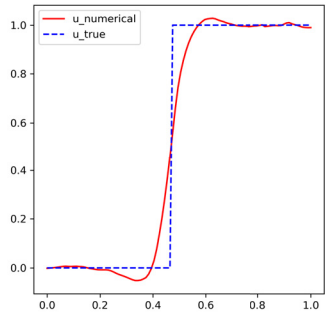
(b) The exact solution



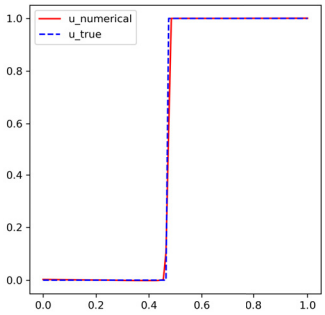
(c) A 3–1376–1 ReLU NN function approximation



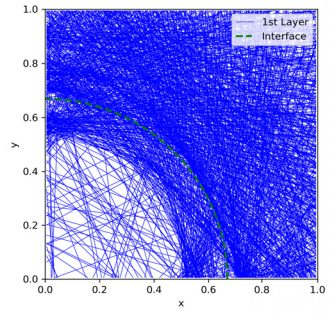
(d) A 3–80–80–1 ReLU NN function approximation



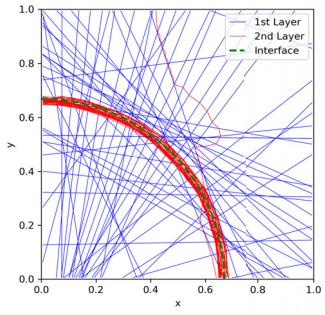
(e) The trace of Figure 11(c) on $y = x$



(f) The trace of Figure 11(d) on $y = x$



(g) The breaking hyperplanes of the approximation in Figure 11(c)



(h) The breaking hyperplanes of the approximation in Figure 11(d)

FIG. 11. *Approximation results of the problem in subsection* 5.2.3.

REFERENCES

[1] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, *Understanding deep neural networks with rectified linear units*, in International Conference on Learning Representations, 2018.

[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 (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. 17, Elsevier, Amsterdam, 2016, pp. 289–317, https://doi.org/10.1016/bs.hna.2016.07.002.

[5] 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., 433 (2023), 115298.

[6] 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.

[7] 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.

[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), 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*, Ann. Appl. Probab., 33 (2023), pp. 3039–3079, https://doi.org/10.1214/22-AAP1884.

[11] J. Chen, *Least-Squares ReLU Neural Network Method for Scalar Hyperbolic Conservation Law*, Ph.D. thesis, Purdue University, 2021.

[12] 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.

[13] 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.

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

[15] R. A. DeVore, K. I. Oskolkov, and P. P. Petrushev, *Approximation by feed-forward neural networks*, Ann. Numer. Math., 4 (1996), pp. 261–288.

[16] M. W. M. G. Dissanayake and N. Phan-Thien, *Neural network based approximations for solving partial differential equations*, Commun. Numer. Methods Eng., 10 (1994), pp. 195–201, https://doi.org/10.1002/cnm.1640100303.

[17] 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.

[18] O. Fuks and H. A. Tchelepi, *Limitations of physics informed machine learning for nonlinear two-phase transport in porous media*, J. Mach. Learn. Model. Comput., 1 (2020), pp. 19–37, https://doi.org/10.1615/JMachLearnModelComput.2020033905.

[19] 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.

[20] M. Imaizumi and K. Fukumizu, *Deep neural networks learn non-smooth functions effectively*, in Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, PMLR, 2019, pp. 869–878.

[21] M. Imaizumi and K. Fukumizu, *Advantage of deep neural networks for estimating functions with singularity on hypersurfaces*, J. Mach. Learn. Res., 23 (2022), pp. 1–53.

[22] 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.

[23] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in International Conference on Learning Representations, 2015.

[24] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Trans. 941 Neural Networks, 9 (1998), pp. 987–1000, https://doi.org/10.1109/72.712178.

[25] M. Liu and Z. Cai, *Adaptive two-layer ReLU neural network:* II. *Ritz approximation to elliptic PDEs*, Comput. Math. Appl., 113 (2022), pp. 103–116, https://doi.org/10.1016/j.camwa.2022.03.010.

[26] 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.

[27] M. Liu, Z. Cai, and K. Ramani, *Deep Ritz method with adaptive quadrature for linear elasticity*, Comput. Methods Appl. Mech. Engrg., 415 (2023), 116229, https://doi.org/10.1016/j.cma.2023.116229.

[28] 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.

[29] Q. Liu and S. Zhang, *Adaptive flux-only least-squares finite element methods for linear transport equations*, J. Sci. Comput., 84 (2020), 26, https://doi.org/10.1007/s10915-020-01269-y.

[30] T. Mao and D.-X. Zhou, *Rates of approximation by relu shallow neural networks*, J. Complexity, 79 (2023), 101784, https://doi.org/10.1016/j.jco.2023.101784.

[31] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, *On the number of linear regions of deep neural networks*, Adv. Neural Inf. Process. Syst., 27 (2014).

[32] R. Pascanu, G. Montufar, and Y. Bengio, *On the number of response regions of deep feed forward networks with piece-wise linear activations*, in International Conference on Learning Representations, 2014.

[33] 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.

[34] 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.

[35] A. Pinkus, *Approximation theory of the MLP model in neural networks*, Acta Numer., 8 (1999), pp. 143–195, https://doi.org/10.1017/S0962492900002919.

[36] 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.

[37] Z. Shen, H. Yang, and S. Zhang, *Deep network approximation characterized by number of neurons*, Commun. Comput. Phys., 28 (2020), pp. 1768–1811, https://doi.org/10.4208/cicp.OA-2020-0149.

[38] 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.

[39] J. Tarela and M. Martinez, *Region configurations for realizability of lattice piecewise-linear models*, Math. Comput. Model., 30 (1999), pp. 17–27, https://doi.org/10.1016/S0895-7177(99)00195-8.

[40] 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.

[41] S. Zhang, *Battling Gibbs phenomenon: On finite element approximations of discontinuous solutions of PDEs*, Comput. Math. Appl., 122 (2022), pp. 35–47, https://doi.org/10.1016/j.camwa.2022.07.014.