FRIEDRICHS LEARNING: WEAK SOLUTIONS OF PARTIAL DIFFERENTIAL EQUATIONS VIA DEEP LEARNING

FAN CHEN AND JIANGUO HUANG

SCHOOL OF MATHEMATICAL SCIENCES, AND MOE-LSC,
SHANGHAI JIAO TONG UNIVERSITY, SHANGHAI 200240, CHINA.

CHUNMEI WANG

DEPARTMENT OF MATHEMATICS & STATISTICS, TEXAS TECH UNIVERSITY, 1108 MEMORIAL CIRCLE, LUBBOCK, TX 79409, USA

HAIZHAO YANG

DEPARTMENT OF MATHEMATICS, PURDUE UNIVERSITY, WEST LAFAYETTE, IN 47907, USA

Abstract. This paper proposes Friedrichs learning as a novel deep learning methodology that can learn the weak solutions of PDEs via Friedrichs' seminal minimax formulation, which transforms the PDE problem into a minimax optimization problem to identify weak solutions. The name "Friedrichs learning" is for Friedrichs' contribution to the minimax framework for PDEs in a weak form. The weak solution and the test function in the weak formulation are parameterized as deep neural networks in a mesh-free manner, which are alternately updated to approach the optimal solution networks approximating the weak solution and the optimal test function, respectively. Extensive numerical results indicate that our mesh-free method can provide reasonably good solutions to a wide range of PDEs defined on regular and irregular domains in various dimensions, where classical numerical methods such as finite difference methods and finite element methods may be tedious or difficult to be applied.

Key words. Partial Differential Equation; Friedrichs' System; Minimax Optimization; Weak Solution; Deep Neural Network; High Dimension; Complex Domain.

AMS subject classifications. 65M75; 65N75; 62M45;

1. Introduction. High-dimensional PDEs and PDEs defined on complex domains are important tools in physical, financial, and biological models [45, 18, 58, 24, 57]. Generally speaking, they do not have closed-form solutions making numerical solutions of such equations indispensable in real applications. First, developing numerical methods for high-dimensional PDEs has been a challenging task due to the curse of dimensionality in conventional discretization. Second, conventional numerical methods rely on mesh generation that requires profound expertise and programming skills. Especially, for problems defined in complicated domains, it is challenging and time-consuming to implement conventional methods. As an efficient parametrization tool for high-dimensional functions [7, 16, 49, 48, 35, 38, 53, 54] with user-friendly software (e.g., TensorFlow and PyTorch), neural networks have been applied to solve PDEs via various approaches recently. The idea of using neural networks to solve PDEs dates back to 1990s [44, 25, 14, 43] and was revisited and popularized recently [15, 28, 17, 41, 55, 9, 46, 8, 37, 36, 12, 52, 47].

Most network-based PDE solvers are concerned with the strong solutions that are differentiable and satisfy PDEs in common sense. Unlike strong solutions, weak solutions are functions for which the derivatives may not all exist but which are nonetheless deemed to satisfy the PDE in some precisely defined sense. These solutions are crucial because many PDEs in modeling real-world phenomena do not have sufficiently smooth solutions. Though there has been extensive research on network-based PDE solvers in the strong form, to the best of our knowledge, there is only one solver reported for weak solutions [59, 6], which was designed for elliptic PDEs with solutions in $H^1(\Omega)$. This paper proposes Friedrichs learning as an alternative method that can learn the weak solutions of elliptic and hyperbolic PDEs in $L^2(\Omega)$ via Friedrichs' seminal minimax formulation, which transforms the PDE problem into a minimax optimization problem to identify

weak solutions. Note that no regularity for the solution is required in Friedrichs learning, which is the main advantage of the proposed method, making it applicable to a wider range of PDE problems. Besides, Friedrichs learning is capable of solving PDEs with discontinuous solutions without a priori knowledge of the location of the discontinuity. Although the current version of Friedrichs learning may not be able to provide highly accurate solutions, a coarse solution without a priori knowledge of the discontinuity could provide a rough estimation of the discontinuity and could be a good initial guess of conventional computation approaches following the Int-Deep framework in [34].

The main philosophy of Friedrichs learning is to reformulate the PDE problem into a minimax optimization, the solution of which is a test deep neural network (DNN) that maximizes the loss and a solution DNN that minimizes the loss. For a high order PDE, we first reformulate it into a first-order PDE system by introducing auxiliary functions, the weak form of which naturally leads to a minimax optimization by integration by parts according to the theory of Friedrichs system [23]. These two main steps are also the key difference from [59] making our Friedrichs learning a more general framework for a wider range of PDEs than existing deep learning methods for weak solutions. Let us introduce the formulation of Friedrichs learning using first-order boundary value problems (BVPs) with homogeneous boundary conditions without loss of generality, since initial value problems (IVPs) can be treated as BVPs, where the time variable is considered to be one more spatial variable and the inhomogeneous boundary condition can be simply transferred to a homogenous one by subtracting the boundary function from the solution.

In the seminal results by Friedrichs in [23] and other investigations in [5, 22], an abstract framework of the boundary value problem of the first-order system was established, which is referred to as the Friedrichs' system in the literature. Let us introduce this concept using a concrete and simple example here to illustrate the main idea and intuition. A more abstract framework will be discussed later in Section 2. Let $r \in \mathbb{N}$ and $\Omega \subset \mathbb{R}^d$ be an open and bounded set with Lipschitz boundary $\partial\Omega$. If real matrix functions $\boldsymbol{A}_k \in W^{1,\infty}(\Omega; M_r)$[1] for $k = 1, \ldots, d$, and a matrix function $\boldsymbol{C} \in L^\infty(\Omega; M_r)$ satisfy: $\boldsymbol{A}_k = \boldsymbol{A}_k^T$ and $\boldsymbol{C} + \boldsymbol{C}^T + \sum_{k=1}^d \partial_k \boldsymbol{A}_k \geq 2\mu_0 \boldsymbol{I}$ for some $\mu_0 > 0$ and the identity matrix $\boldsymbol{I}$, then the first-order differential operator $T : \mathcal{D} \subset L \to L$ with $L := L^2(\Omega; \mathbb{R}^r)$ defined by $T\boldsymbol{u} := \sum_{k=1}^d \partial_k(\boldsymbol{A}_k\boldsymbol{u}) + \boldsymbol{C}\boldsymbol{u}$ is called the Friedrichs operator, while the first-order system of PDE $T\boldsymbol{u} = \boldsymbol{f}$ is called the Friedrichs system, where $\mathcal{D}$ is a dense subspace of L and $\boldsymbol{f}$ is a given data function in L.

Friedrichs [23] also introduced an abstract framework for representing boundary conditions via matrix valued boundary fields. First, define $\boldsymbol{A}_{\boldsymbol{v}} := \sum_{k=1}^d v_k \boldsymbol{A}_k \in L^\infty(\partial\Omega; M_r)$, where $\boldsymbol{v} \in \mathbb{R}^d$ is the unit outward normal direction on $\partial\Omega$, and let $\boldsymbol{M} : \partial\Omega \to M_r$ be a matrix field on the boundary. Then a homogeneous Dirichlet boundary condition of the Friedrichs' system is prescribed by $(\boldsymbol{A}_{\boldsymbol{v}} - \boldsymbol{M})\boldsymbol{u} = \boldsymbol{0}$ on $\partial\Omega$ by choosing an appropriate $\boldsymbol{M}$ to ensure the well-poseness of the Friedrichs' system. In real applications, $\boldsymbol{M}$ is given by physical knowledge. The graph space $W := \{\boldsymbol{u} \in L : T\boldsymbol{u} \in L\}$ is an inner product space with the inner product $(\cdot,\cdot)_T := (\cdot,\cdot)_L + (T\cdot, T\cdot)_L$, which induces the graph norm $\|\cdot\|_W$. Then an indefinite inner product on W is introduced as $\langle \boldsymbol{u}, \boldsymbol{v} \rangle := (T\boldsymbol{u}, \boldsymbol{v})_L - (\boldsymbol{u}, \tilde{T}\boldsymbol{v})_L$ for $\boldsymbol{u}, \boldsymbol{v} \in W$, where $\tilde{T}$ means the formal adjoint operator of T. The indefinite inner product arises from the integration by parts intuitively and induces a linear boundary operator $\mathcal{B}$ on W defined by $(\mathcal{B}\boldsymbol{u}, \boldsymbol{v})_L := \langle \boldsymbol{u}, \boldsymbol{v} \rangle$.

Once the matrix field $\boldsymbol{M}$ and the boundary operator $\mathcal{B}$ are specified, let $V := \ker(\mathcal{B} - \boldsymbol{M})$ and $V^* := \ker(\mathcal{B} + \boldsymbol{M}^*)$, where M^* is the adjoint of M. Then it can be proved that $\boldsymbol{u}$ solves the boundary value problem (BVP)

$$T\boldsymbol{u} = \boldsymbol{f} \text{ in } \Omega, \quad \text{and} \quad (\boldsymbol{A}_{\boldsymbol{v}} - \boldsymbol{M})\boldsymbol{u} = \boldsymbol{0} \text{ on } \partial\Omega \tag{1.1}$$

[1] M_r means the set of matrix functions of size $r \times r$.

if and only if $\boldsymbol{u}$ also solves the minimax problem

$$\min_{\boldsymbol{u}\in V}\max_{\boldsymbol{v}\in V^*}\mathcal{L}(\boldsymbol{u},\boldsymbol{v})=\frac{(\boldsymbol{u},\tilde{T}\boldsymbol{v})_L-(\boldsymbol{f},\boldsymbol{v})_L}{\|\tilde{T}\boldsymbol{v}\|_L}.$$

Hence, in Friedrichs learning, DNNs are applied to parametrize $\boldsymbol{u}$ and $\boldsymbol{v}$ to solve the above minimax problem to obtain the solution of the BVP in (1.1). Friedrichs learning also works for other kinds of boundary conditions. For simplicity, we will focus on the Dirichlet boundary condition in this paper.

This paper is organized as follows. In Section 2, we introduce Friedrichs' minimax formulation for weak solutions and its variant for time-dependent PDEs. In Section 3, several concrete examples of PDEs and their minimax formulation are provided. In Section 4, network-based optimization is introduced to solve the minimax problem in Friedrichs' formulation to complete our Friedrichs learning. In Section 5, several numerical examples are provided to demonstrate the proposed Friedrichs learning. Finally, we conclude this paper in Section 6.

2. Friedrichs' Minimax Formulation for Weak Solutions. In this section, we shall briefly review Friedrichs' system in a Hilbert space setting following the structure in [1, 10]. Then we will introduce Friedrichs' minimax formulation for weak solutions as the foundation of Friedrichs learning.

2.1. An Abstract Framework of Friedrichs' System. Firstly, we recall some basic results on Friedrichs' system for later uses [10, 1]. We assume L is a real Hilbert space associated with the inner product $(\cdot,\cdot)_L$ and the induced norm $\|\cdot\|_L$. The dual space of L, denoted by L', can be identified naturally with L by the Riesz representation theorem. For a dense subspace $\mathcal{D}$ of L, we consider two linear operators $T:\mathcal{D}\to L$ and $\tilde{T}:\mathcal{D}\to L$ satisfying the following properties: for any $u,v\in\mathcal{D}$, there exists a positive constant C such that

$$(Tu,v)_L=(u,\tilde{T}v)_L, \tag{2.1}$$

$$\|(T+\tilde{T})u\|_L\le C\|u\|_L. \tag{2.2}$$

Usually, $\tilde{T}$ is called the formal adjoint of T. Then, as shown in [1, Lemma 2.1], we can define a graph space W by

$$W=\{u\in L: Tu\in L\}, \tag{2.3}$$

which is a Hilbert space with respect to the graph norm $\|\cdot\|_T=(\cdot,\cdot)_T^{1/2}$ which induced by the inner product $(\cdot,\cdot)_T=(\cdot,\cdot)_L+(T\cdot,T\cdot)_L$. In addition, owing to (2.2),

$$W=\{u\in L:\tilde{T}u\in L\}.$$

In other words, W is also a graph space associated with $\tilde{T}$.

The abstract framework of Friedrichs' system concerns the solvability of the problem

$$Tu=f\in L, \tag{2.4}$$

and its solution falls in the graph space W. Obviously, the problem (2.4) may not be well-posed since its solution in W may not be unique. We are interested in constructing a subspace $V\subseteq W$ such that $T:V\to L$ is an isomorphism. A standard way is carried out as follows. We first define a self-adjoint boundary operator $B\in\mathcal{L}(W,W')$ as follows (cf. [1]). For any $u,v\in W$,

$$\langle Bu,v\rangle_{W'\times W}=(Tu,v)_L-(u,\tilde{T}v)_L. \tag{2.5}$$

We assume that there exists an operator $M \in \mathcal{L}(W, W')$ such that

$$\langle Mw, w\rangle \geq 0, \forall w \in W, \tag{2.6}$$

$$W = \mathcal{N}(B - M) + \mathcal{N}(B + M), \tag{2.7}$$

where $\mathcal{N}$ is the nullspace of its argument. Moreover, let $M^* \in \mathcal{L}(W, W')$ denote the adjoint operator of M defined as follows.

$$\langle M^* u, v\rangle_{W'\times W} = \langle Mv, u\rangle_{W'\times W}, \quad \forall\, (u, v) \in W \times W.$$

To find V such that the problem (2.4) is well-posed, we should make an additional assumption for L as follows.

$$((T + \tilde{T})v, v)_L \geq 2\mu_0 \|v\|_L^2, \quad \forall\, v \in L, \tag{2.8}$$

where μ_0 is a positive constant. Then we we choose

$$V = \mathcal{N}(B - M), \quad V^* = \mathcal{N}(B + M^*). \tag{2.9}$$

According to [1, Lemma 3.2 and Theorem 3.1], we have the following important result for Friedrichs' system.

THEOREM 2.1. *Assume* (2.2) *and* (2.8) *and choose M such that* (2.6) *and* (2.7) *hold. Let V and V^* are given as in* (2.9)*. Then the following statements hold:*

1. *For all $v \in W$, there hold*

$$\mu_0 \|v\|_L \leq \|Tv\|_L, \quad \mu_0 \|v\|_L \leq \|\tilde{T}v\|_L. \tag{2.10}$$

2. *For all $f \in L$, problem* (2.4) *have a unique solution in V. In other words, T is an isomorphism from V onto L. Moreover, $\tilde{T}$ is an isomorphism from V^* onto L.*

2.2. First Order PDEs of Friedrichs' Type. As a typical application of the above framework, which is enough for the study in this paper, we restrict L to be the space of square integral (vector-valued) functions over an open and bounded domain $\Omega \subset \mathbb{R}^d$ with Lipschitz boundary, $\mathcal{D}$ to be the space of test functions, and T to be a first-order differential operator with its formal adjoint $\tilde{T}$. In particular, we take $L = [L^2(\Omega)]^m$, $m \in \mathbb{N}$ and $\mathcal{D} = [D(\Omega)]^m$, where $D(\Omega) = C_0^\infty(\Omega)$. $\mathcal{D}$ is thus dense in L.

Consider $T : \mathcal{D} \to L$ as

$$T\boldsymbol{u} = \sum_{k=1}^{d} \boldsymbol{A}_k \partial_k \boldsymbol{u} + \boldsymbol{C}\boldsymbol{u} = \boldsymbol{f}, \qquad \forall u \in \mathcal{D}, \tag{2.11}$$

with the standard assumptions being imposed on $\boldsymbol{A}_k$ and $\boldsymbol{C}$ for Friedrichs' system [19, 20, 23]:

$$\boldsymbol{C} \in [L^\infty(\Omega)]^{m,m}, \tag{2.12}$$

$$\boldsymbol{A}_k \in [L^\infty(\Omega)]^{m,m}, k = 1, \cdots, d \quad \text{and} \quad \sum_{k=1}^{d} \partial_k \boldsymbol{A}_k \in [L^\infty(\Omega)]^{m,m} \tag{2.13}$$

$$\boldsymbol{A}_k = \boldsymbol{A}_k^T, \quad \text{a. e. in } \Omega, k = 1, \cdots, d. \tag{2.14}$$

The formal adjoint $\tilde{T}:\mathcal{D}\to L$ of T is thus as follows

$$\tilde{T}\boldsymbol{u}=-\sum_{k=1}^{d}\boldsymbol{A}_k\partial_k\boldsymbol{u}+(\boldsymbol{C}^*-\sum_{k=1}^{d}\partial_k\boldsymbol{A}_k)\boldsymbol{u},\qquad\forall\boldsymbol{u}\in\mathcal{D}. \tag{2.15}$$

It is easy to see that T and $\tilde{T}$ satisfy (2.1)-(2.2). All the results in this section hold true for Friedrichs' system satisfying (2.12)-(2.14).

Regarding to the abstract Friedrichs' system, an explicit representation of B could be found; however, it is impossible for M on the abstract level. Assume $\mathcal{B}=\sum_{k=1}^{d}n_k\boldsymbol{A}_k$ is well-defined a.e. on $\partial\Omega$ where $\boldsymbol{n}=(n_1,\cdots,n_d)$ is the unit outward normal vector of $\partial\Omega$. For simplicity of notations, we set $\mathcal{H}=[H^s]^m$ with H^s being the usual Sobolev space of order s, and $\mathcal{C}^1=[C^1]^m$ with C^1 being the space of continuously differentiable functions.

LEMMA 2.2. *[3, 39] For $\boldsymbol{u},\boldsymbol{v}\in\mathcal{H}^1(\Omega)\subset W(\Omega)$, there holds*

$$\langle B\boldsymbol{u},\boldsymbol{v}\rangle_{W'(\Omega)\times W(\Omega)}=\langle\mathcal{B}\boldsymbol{u},\boldsymbol{v}\rangle_{\mathcal{H}^{-\frac{1}{2}}(\partial\Omega)\times\mathcal{H}^{\frac{1}{2}}(\partial\Omega)}.$$

Specifically, $\langle B\boldsymbol{u},\boldsymbol{v}\rangle_{W'(\Omega)\times W(\Omega)}=\int_{\partial\Omega}\boldsymbol{v}^T\mathcal{B}\boldsymbol{u}ds$, for any $\boldsymbol{u},\boldsymbol{v}\in C_0^\infty(\mathbb{R}^d)$.

If Ω has segment property [4], $\mathcal{C}^1$ is thus dense in $\mathcal{H}^1(\Omega)$ and further is dense in W. Therefore, the representation could be uniquely extended to the whole space W in the sense that for any $u\in W(\Omega)$ and $v\in\mathcal{H}^1(\Omega)$,

$$\langle Bu,v\rangle_{W'(\Omega)\times W(\Omega)}=\langle\mathcal{B}u,v\rangle_{\mathcal{H}^{-\frac{1}{2}}(\partial\Omega)\times\mathcal{H}^{-\frac{1}{2}}(\partial\Omega)}. \tag{2.16}$$

The coercivity condition on T dictated by the positiveness condition on the coefficients $\boldsymbol{A}_k$ and $\boldsymbol{C}$ [19, 20, 21] is needed to show the well-posedness of PDEs of Friedrichs' type. After some direct manipulation, the abstract coercivity condition (2.8) is equivalent to the following full coercivity for Friedrichs' PDEs:

$$\boldsymbol{C}+\boldsymbol{C}^*-\sum_{k=1}^{d}\partial_k\boldsymbol{A}_k\geq 2\mu_0\boldsymbol{I}_m,\quad\text{a.e., in }\Omega, \tag{2.17}$$

where μ_0 is a positive constant and I_m is the $m\times m$ identity matrix. If a system does not satisfies the above condition we can often introduce a feasible transformation so that the modified system satisfies this condition. In [10], the authors introduced the so-called partial coercivity condition to study the mathematical theory of the corresponding system. We refer to [10] for more details.

2.3. Friedrichs' Minimax Formulation. Throughout this subsection, we assume all the conditions given in Theorem 2.1 hold. Recall that $V=\mathcal{N}(B-M)$ and $V^*=\mathcal{N}(B+M^*)$ with $M\in\mathcal{L}(W,W')$ satisfying conditions (2.6)-(2.7). For a given $f\in L$, the problem under study is find the solution $u\in V$ such that

$$Tu=f. \tag{2.18}$$

The solution $u\in V$ of (2.18) also satisfies

$$(Tu,v)_L=(f,v)_L,\forall v\in L. \tag{2.19}$$

We restrict $v\in V^*\subset L$. From (2.5),

$$\begin{aligned}(Tu,v)_L=&(u,\tilde{T}v)_L+\langle Bu,v\rangle_{W'\times W}\\=&(u,\tilde{T}v)_L+\langle(B-M)/2\,u,v\rangle_{W'\times W}+\langle(B+M)/2\,u,v\rangle_{W'\times W}\\=&(u,\tilde{T}v)_L+\langle u,(B+M^*)/2\,v\rangle_{W'\times W}\\=&(u,\tilde{T}v)_L,\end{aligned} \tag{2.20}$$

where we used $u \in V = \mathcal{N}(B - M)$ and $v \in V^* = \mathcal{N}(B + M^*)$. This combined with (2.19) gives

$$(u, \tilde{T}v)_L = (f, v)_L, \forall v \in V^*.$$

For $u \in V$, $v \in V^*$, we define

$$L(u, v) = \frac{|(u, \tilde{T}v)_L - (f, v)_L|}{\|\tilde{T}v\|_L}. \tag{2.21}$$

Furthermore, according to the estimate (2.10), we have

$$|(u, \tilde{T}v)_L - (f, v)_L| \leq \|u\|_L \|\tilde{T}v\|_L + \|f\|_L \|v\|_L \leq (\|u\|_L + \frac{1}{\mu_0}\|f\|_L)\|\tilde{T}v\|_L,$$

where μ_0 is defined in (2.8), therefore L is bounded. Then the functional $L(u, v)$ is bounded with respect to $v \in V^*$ for a fixed $u \in L$.

Thus we can reformulate the problem (2.18) as the following minimax problem formally:

$$\min_{u \in V} \max_{v \in V^*} L(u, v) = \frac{|(u, \tilde{T}v)_L - (f, v)_L|}{\|\tilde{T}v\|_L}. \tag{2.22}$$

THEOREM 2.3. *u is a solution of* (2.18) *if and only if u solves the minimax problem* (2.22).

Proof. On one hand, if $u \in V$ is a solution of (2.18), as shown above we know $L(u, v) = 0$, for any $v \in V^*$. Thus, u is a solution of the minimax problem (2.22). One the other hand, we will claim that u is the only solution of problem (2.22). To this end, we assume $u' \neq u$ is another solution of the minimax problem (2.22), we have

$$\max_{v \in V^*} L(u', v) = \max_{v \in V^*} \frac{|(u', \tilde{T}v)_L - (f, v)_L|}{\|\tilde{T}v\|_L} = 0.$$

Thus, we have $L(u', v) = 0$ for all $v \in V^*$. This implies

$$(u', \tilde{T}v)_L - (f, v)_L = 0, \quad v \in V^*,$$

which gives

$$(Tu' - f, v)_L = 0, \quad v \in V^*.$$

Due to $\mathbf{C}_c^\infty(\Omega) \subset V^*$ we know $Tu' = f$ in the sense of distribution. □

We remark that in view of the equivalence between the weak solution and the strong solution given by Friedrichs (cf. [23]), we can also show (2.18) is equivalent to the following minimax problem

$$\min_{u \in L} \max_{v \in V^*} L(u, v) = \frac{|(u, \tilde{T}v)_L - (f, v)_L|}{\|\tilde{T}v\|_L}. \tag{2.23}$$

3. Examples of PDEs and the Corresponding Minimax Formulation. Using the abstract framework and the minimax formulation developed in Section 2, we can obtain some minimax formulations for several typical PDEs, which are given one by one below.

3.1. Advection-Reaction Equation. The advection-reaction equation takes the form:

$$\mu u + \boldsymbol{\beta} \cdot \nabla u = f. \tag{3.1}$$

In the equation above, $\boldsymbol{\beta}$ is a vector field in $\mathbb{R}^d$ such that $\boldsymbol{\beta} \in [L^\infty(\Omega)]^d$, $\nabla \cdot \boldsymbol{\beta} \in L^\infty(\Omega)$, $\mu \in L^\infty(\Omega)$ and $f \in L^2(\Omega)$ is given. Compared with Equation (2.11), Equation (3.1) is a Friedrichs system by setting $\boldsymbol{A}_k = \beta_k$ for $k = 1, 2, ..., d$ and $\boldsymbol{C} = \mu$.

The function μ satisfies that there exists $\mu_0 > 0$ such that

$$\mu(\boldsymbol{x}) - \frac{1}{2}\nabla \cdot \boldsymbol{\beta}(\boldsymbol{x}) \geq \mu_0 > 0, \quad \text{a.e.in } \Omega, \tag{3.2}$$

so that the full coercivity condition in (2.17) holds. The graph space W given in (2.3) is

$$W = \{w \in L^2(\Omega); \beta \cdot \nabla w \in L^2(\Omega)\}.$$

We define the inflow and outflow boundary in Equation (3.1):

$$\partial\Omega^- = \{\boldsymbol{x} \in \partial\Omega; \boldsymbol{\beta}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) < 0\}, \quad \partial\Omega^+ = \{\boldsymbol{x} \in \partial\Omega; \boldsymbol{\beta}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) > 0\}. \tag{3.3}$$

To enforce boundary conditions for well-poseness, we apply the *cone formalism* to obtain

$$V = \{v \in W; v|_{\partial\Omega^-} = 0\}, \tag{3.4}$$

$$V^* = \{v \in W; v|_{\partial\Omega^+} = 0\}. \tag{3.5}$$

In the case of boundary conditions given by (2.9), we can set M such that

$$Mu|_{\partial\Omega} = |\boldsymbol{\beta} \cdot \boldsymbol{n}|\, u|_{\partial\Omega}, \tag{3.6}$$

which is equivalent to (3.4). By Equation (2.11),

$$\tilde{T}v = -\sum_{i=1}^{d} \left(\beta_i \frac{\partial}{\partial x_i}(v) + \frac{\partial}{\partial x_i}\beta_i v\right) + \boldsymbol{C}^T v = -\boldsymbol{\beta} \cdot \nabla v - (\nabla \cdot \boldsymbol{\beta})v + \mu v. \tag{3.7}$$

Hence, the loss function in the minimax problem (2.22) or (2.23) is

$$L(u, v) = \frac{|\big(u, -(\boldsymbol{\beta} \cdot \nabla v + (\nabla \cdot \boldsymbol{\beta})v) + \mu v\big)_\Omega - (f, v)_\Omega|}{\|\boldsymbol{\beta} \cdot \nabla v + (\nabla \cdot \boldsymbol{\beta})v\|_\Omega}, \quad u \in V, v \in V^*. \tag{3.8}$$

We note that if the coercivity condition (3.2) does not hold, we can introduce a transformation $u = e^{\lambda_0 t}\tilde{u}$, so that the equation in $\tilde{u}$ will satisfies (3.2) whenever the positive constant λ_0 is taken large enough.

3.2. Scalar Elliptic PDE's. Let $\mu \in L^\infty(\Omega)$ be positive and uniformly bounded away from zero. Consider the second-order PDE defined on $\Omega \subset \mathbb{R}^d$:

$$-\Delta u + \mu u = f, \tag{3.9}$$

where $f \in L^2(\Omega)$. This PDE can be written into a first-order PDE system

$$\begin{cases} v + \nabla u = 0, \\ \mu u + \nabla \cdot v = f, \end{cases} \tag{3.10}$$

which can be formulated into a Friedrichs' system with $m = d + 1$. The Hilbert space L is chosen as $L = [L^2(\Omega)]^m$. Let $\tilde{u} = (v, u) \in L$ and, for $k = 1, 2, ..., d$,

$$\boldsymbol{A}_k = \begin{bmatrix} 0 & \boldsymbol{e}^k \\ (\boldsymbol{e}^k)^T & 0 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_d & 0 \\ 0 & \mu \end{bmatrix},$$

where e^k is the k-th canonical basis of $\mathbb{R}^d$. Since $\mu > 0$ and has a lower bound away from zero, the full coercivity condition is satisfied. The graph space is

$$W = H(\mathrm{div}; \Omega) \times H^1(\Omega).$$

One possible choice of the boundary condition is

$$V = V^* = H(\mathrm{div}; \Omega) \times H_0^1(\Omega) = \{(v, u) \in W; u|_{\partial\Omega} = 0\}, \tag{3.11}$$

which is a Dirichlet boundary condition. The choices of boundary conditions are not unique obviously. By introducing auxiliary variables, the second-order linear PDE can be reformulated into a first-order PDE system. Finally, the weak solution of (3.9) can be found by solving the equivalent minimax problem in (2.22).

Denote the test function by $\psi = (\psi_u, \psi_v)^\perp$ in the space V^*. The minimax problem can be presented as

$$L(\tilde{u}, \psi) = \frac{|\big(-v, \psi_v - \nabla\psi_u\big)_\Omega + (u, \mu\psi_u - \nabla \cdot \psi_v)_\Omega - (f, v)_\Omega|}{\|(\psi_v - \nabla\psi_u, \mu\psi_u - \nabla \cdot \psi_v)^\intercal\|_\Omega}, \quad \tilde{u} \in V, \psi \in V^*. \tag{3.12}$$

By the integration by part, (3.12) can be simplified as

$$L(\tilde{u}, \psi) = \frac{|(u, \mu\psi_u - \Delta\psi_u)_\Omega - (f, v)_\Omega|}{\|(0, \mu\psi_u - \Delta\psi_u)^\intercal\|_\Omega}, \quad \tilde{u} \in V, \psi \in V^*. \tag{3.13}$$

3.3. Maxwell's Equation in the Diffusion Regime. The Maxwell's equations in $\mathbb{R}^3$ in the diffusive regime could be considered as

$$\begin{cases} \mu\boldsymbol{H} + \nabla \times \boldsymbol{E} = \boldsymbol{f}, \\ \sigma\boldsymbol{E} - \nabla \times \boldsymbol{H} = \boldsymbol{g}, \end{cases} \tag{3.14}$$

with μ and σ being two positive functions in $L^\infty(\Omega)$ and uniformly bounded away from zero. Three-dimensional functions $(\boldsymbol{f}, \boldsymbol{g})$ lie in the space $[L^2(\Omega)]^3 \times [L^2(\Omega)]^3$ and the solution functions $(\boldsymbol{H}, \boldsymbol{E})$ are in the Hilbert space $L = [L^2(\Omega)]^3 \times [L^2(\Omega)]^3$. In Equation (2.11), set $m = 6$ and let $\boldsymbol{A}_k \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{C}$ be

$$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{\mathcal{R}^k} \\ (\boldsymbol{\mathcal{R}^k})^T & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \mu \cdot \boldsymbol{I}_3 & \boldsymbol{0} \\ \boldsymbol{0} & \sigma \cdot \boldsymbol{I_3} \end{bmatrix},$$

for $k = 1, 2, 3$. Here, the entries of $\mathcal{R}^k_{ij} = \mathrm{sign(i - j)}$ if $i = k + 1(mod\ 3)$ and $\mathcal{R}^k_{ij} = 0$ otherwise. The graph space is defined as

$$W = H(\mathrm{curl}; \Omega) \times H(\mathrm{curl}; \Omega).$$

One example of the boundary condition is

$$V = V^* = H(\mathrm{curl}; \Omega) \times H_0(\mathrm{curl}; \Omega). \tag{3.15}$$

The function pair $\boldsymbol{u} := (\boldsymbol{H}, \boldsymbol{E}) \in W$ is in V whenever $\boldsymbol{E} \times \boldsymbol{n}|_{\partial\Omega} = 0$. Let $\boldsymbol{\psi} = (\boldsymbol{\psi_H}, \boldsymbol{\psi_E})^\perp$ be the test function in V^*. Then the minimax problem in (2.22) becomes

$$L(\tilde{\boldsymbol{u}}, \boldsymbol{\psi}) = \frac{|\big(\boldsymbol{H}, -\nabla \times \boldsymbol{\psi_E} + \mu\boldsymbol{\psi_H}\big)_\Omega + \big(\boldsymbol{E}, \nabla \times \boldsymbol{\psi_H} + \sigma\boldsymbol{\psi_E}\big)_\Omega - (\boldsymbol{f}, \boldsymbol{\psi_H})_\Omega - (\boldsymbol{g}, \boldsymbol{\psi_E})_\Omega|}{\|(-\nabla \times \boldsymbol{\psi_E} + \mu\boldsymbol{\psi_H}, \nabla \times \boldsymbol{\psi_H} + \sigma\boldsymbol{\psi_E})^\intercal\|_\Omega}. \tag{3.16}$$

4. Deep Learning-Based Solver. To complete the introduction of Friedrichs learning, we introduce a deep learning-based method to solve the minimax optimization in (2.22) to find the weak solution of (2.18) in this section.

4.1. Overview. In the deep learning-based method, one solution DNN, $\phi_s(\boldsymbol{x};\theta_s)$, is applied to parametrize the weak solution u in (2.22) and another test DNN, $\phi_t(\boldsymbol{x};\theta_t)$, is used to parametrize the test function ψ in (2.22). Here, θ_s and θ_t are the parameters to be identified such that

$$(\bar{\theta}_s,\bar{\theta}_t)=\arg\min_{\theta_s}\max_{\theta_t} L(\phi_s(\boldsymbol{x};\theta_s),\phi_t(\boldsymbol{x};\theta_t)) := \frac{|(\phi_s(\boldsymbol{x};\theta_s),\tilde{T}\phi_t(\boldsymbol{x};\theta_t))_\Omega-(f,\phi_t(\boldsymbol{x};\theta_t))_\Omega|}{\left\|\tilde{T}\phi_t(\boldsymbol{x};\theta_t)\right\|_\Omega}, \tag{4.1}$$

under the constraints

$$\phi_s(\boldsymbol{x};\theta_s)\in V \text{ and } \phi_t(\boldsymbol{x};\theta_t)\in V^*.$$

For simplicity, we use $L(\theta_s,\theta_t)$ for short to represent $L(\phi_s(\boldsymbol{x};\theta_s),\phi_t(\boldsymbol{x};\theta_t))$ from now on.

To solve the constrained minimax optimization above, we apply the penalty method. For this purpose, we shall introduce a distance to quantify how good the solution DNN and test DNN satisfy their constraints. Such a distance is specified according to the boundary conditions. We denote $\mathrm{dist}(\phi(\boldsymbol{x};\theta),V)$ as the distance from a DNN $\phi(\boldsymbol{x};\theta)$ to a space V. Therefore, the penalty term of boundary conditions is

$$L_b(\theta_s,\theta_t) := \lambda_1\log\left(\mathrm{dist}(\phi_s(\boldsymbol{x};\theta_s),V)^2\right)-\lambda_2\log\left(\mathrm{dist}(\phi_t(\boldsymbol{x};\theta_t),V^*)^2\right), \tag{4.2}$$

where λ_1 and λ_2 are two positive hyper-parameters. For instance, when V is the space of functions satisfying a Dirichlet boundary condition, i.e., these functions are equal to a given function $g_d(\boldsymbol{x})$ on the boundary $\partial\Omega'\subset\partial\Omega$, the distance to V is defined as

$$\mathrm{dist}(\phi_s(\boldsymbol{x};\theta_s),V)=\int_{\partial\Omega'}\left(\phi_s(\boldsymbol{x};\theta_s)-g_d(\boldsymbol{x})\right)^2 dx.$$

Finally, the following unconstrained minimax problem

$$(\bar{\theta}_s,\bar{\theta}_t)=\arg\min_{\theta_s}\max_{\theta_t}\left(L(\theta_s,\theta_t)+L_b(\theta_s,\theta_t)\right) \tag{4.3}$$

is solved to obtain the solution DNN $\phi_s(\boldsymbol{x};\bar{\theta}_s)$ as the weak solution of the given PDE in (2.18) by Friedrichs learning.

4.2. Network Implementation. Now, we will introduce the network structure of the solution and test DNNs used in the previous section. In this paper, all DNNs are chosen as ResNet [30] defined as follows. Let $\phi(\boldsymbol{x};\theta)$ denote such a network with an input $\boldsymbol{x}$ and parameters θ, then it is defined recursively using a nonlinear activation function σ as follows:

$$\begin{aligned}
\boldsymbol{h}_0 &= \boldsymbol{V}\boldsymbol{x},\\
\boldsymbol{g}_\ell &= \sigma(\boldsymbol{W}_\ell\boldsymbol{h}_{\ell-1}+\boldsymbol{b}_\ell), \qquad \ell=1,2,\ldots,L,\\
\boldsymbol{h}_\ell &= \bar{\boldsymbol{U}}_\ell\boldsymbol{h}_{\ell-2}+\boldsymbol{U}_\ell\boldsymbol{g}_\ell, \quad \ell=1,2,\ldots,L,\\
\phi(\boldsymbol{x};\theta) &= \boldsymbol{a}^T\boldsymbol{h}_L,
\end{aligned} \tag{4.4}$$

where $\boldsymbol{V}\in\mathbb{R}^{m\times d}$, $\boldsymbol{W}_\ell\in\mathbb{R}^{m\times m}$, $\bar{\boldsymbol{U}}_\ell\in\mathbb{R}^{m\times m}$, $\boldsymbol{U}_\ell\in\mathbb{R}^{m\times m}$, $\boldsymbol{b}_\ell\in\mathbb{R}^m$ for $\ell=1,\ldots,L$, $\boldsymbol{a}\in\mathbb{R}^m$, $\boldsymbol{h}_{-1}=\boldsymbol{0}$. Throughout this paper, $\boldsymbol{U}_\ell$ is set as the identity matrix in the numerical implementation

of ResNets for the purpose of simplicity. Furthermore, as used in [17], we set $\bar{\boldsymbol{U}}_\ell$ as the identify matrix when ℓ is even and set $\bar{\boldsymbol{U}}_\ell = \mathbf{0}$ when ℓ is odd. θ consists of all the weights and biases $\{\boldsymbol{W}^l, \boldsymbol{b}^l\}_{l=0}^L$. The number m is called the width and L is called the depth of the network. The activation function σ is problem-dependent. For example, the activation function is chosen to satisfy a certain smoothness condition. For a network $\phi(\boldsymbol{x};\theta)$, if it is desired to have $\phi(\boldsymbol{x};\theta)$ in the H^k space for an $k \in \mathbb{N}$, the activation function $\mathrm{ReLU}^{k+1}(x)$ is used, where $\mathrm{ReLU}(x) := \max\{0, x\}$.

4.3. Special Network for Different Boundary Conditions. As discussed in [27, 26], it is possible to build special networks to satisfy various boundary conditions automatically, which can reduce the constrained optimization in the previous section to unconstrained optimization, making the optimization easier to solve. When the PDE domain is regular, e.g., a hypercube or a ball, it is simple to construct such special networks for Dirichlet boundary conditions, Neumann boundary conditions, Robin boundary conditions, etc.

Let us take the case of a homogeneous Dirichlet boundary condition for a boundary value problem as an example. For other cases, the reader can refer to [27, 26]. Assume a Dirichlet boundary condition for a function is desired:

$$\psi(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \text{on } \partial\Omega. \tag{4.5}$$

Then a DNN automatically satisfying the condition above can be constructed by

$$\phi(\boldsymbol{x};\theta) = h(\boldsymbol{x})\hat{\phi}(\boldsymbol{x};\theta) + b(\boldsymbol{x}),$$

where $\hat{\phi}$ is a generic network as in (4.4), and h is a specifically chosen function such that $h = 0$ on $\partial\Omega$, and b is chosen such that $b = g$ on $\partial\Omega$.

For example, if Ω is a d-dimensional unit ball, then $\phi(\boldsymbol{x};\theta)$ can take the form

$$\phi(\boldsymbol{x};\theta) = (|\boldsymbol{x}|^2 - 1)\hat{\phi}(\boldsymbol{x};\theta) + b(\boldsymbol{x}). \tag{4.6}$$

For another example, if Ω is the d-dimensional cube $[-1, 1]^d$, then $\phi(\boldsymbol{x};\theta)$ can take the form

$$\phi(\boldsymbol{x};\theta) = \prod_{i=1}^{d}(x_i^2 - 1)\hat{\phi}(\boldsymbol{x};\theta) + b(\boldsymbol{x}). \tag{4.7}$$

If we construct the solution DNN and test DNN satisfying their boundary conditions, then the minimax problem in (4.3) is reduced to the unconstrained minimax problem

$$(\bar{\theta}_s, \bar{\theta}_t) = \arg\min_{\theta_s}\max_{\theta_t} L_i(\theta_s, \theta_t)$$

to identify the solution DNN $\phi_s(\boldsymbol{x};\bar{\theta}_s)$.

4.4. Network Training. Once the solution and test DNNs have been set up, the rest is to train them to solve the minimax problem in (4.3). The stochastic gradient descent (SGD) method or its variants (e.g., RMSprop [31] and Adam [42]) is an efficient tool to solve this problem numerically. Although the convergence of SGD for the minimax problem is still an active research topic [51, 13, 56], empirical success shows that SGD can provide a good approximate solution. The training algorithm and main numerical setup are summarized in Algorithm 1 below.

There is an inner iteration loop and an outer iteration loop in Algorithm 1. The outer iteration loop has n iterations. The inner iteration loop contains n_s steps of θ_s updates and n_t steps of θ_t updates.

In each inner iteration for updating θ_s, we generate two new sets of random samples $\{\boldsymbol{x}_i^1\}_{i=1}^{N_1} \subset \Omega$ and $\{\boldsymbol{x}_i^2\}_{i=1}^{N_2} \subset \partial\Omega_D$ following uniform distributions. Then we define the empirical loss of these training points for the Friedrichs' system (2.11) as

$$L_t(\theta_s, \theta_t) := \hat{L}(\theta_s, \theta_t) - \hat{L}_b(\theta_s, \theta_t), \tag{4.8}$$

where

$$\hat{L}(\theta_s, \theta_t) := \frac{|\hat{L}_n(\theta_s, \theta_t)|}{\hat{L}_d(\theta_s, \theta_t)}$$

with

$$\begin{aligned}\hat{L}_n(\theta_s, \theta_t) := &\frac{1}{N_1}\sum_{i=1}^{N_1}\big(\sum_{j=1}^{d}\frac{\partial}{\partial x_j}(-\boldsymbol{A}_j\phi_t(\boldsymbol{x}_i^1;\theta_t)), \phi_s(\boldsymbol{x}_i^1;\theta_s)\big) + \frac{1}{N_1}\sum_{i=1}^{N_1}\big(\boldsymbol{C}^\intercal\phi_t(\boldsymbol{x}_i^1;\theta_t), \phi_s(\boldsymbol{x}_i^1;\theta_s)\big)\\ &-\frac{1}{N_1}\sum_{i=1}^{N_1}\big(f(\boldsymbol{x}_i^1), \phi_t(\boldsymbol{x}_i^1;\theta_t)\big) + \frac{1}{N_2}\sum_{i=1}^{N_2}\big((\sum_{j=1}^{d}\boldsymbol{A}_j n_j)\phi_s(\boldsymbol{x}_i^2;\theta_s), \phi_t(\boldsymbol{x}_i^2;\theta_t)\big),\end{aligned}$$

and

$$\hat{L}_d(\theta_s, \theta_t) := \frac{1}{N_1}\sum_{i=1}^{N_1}\left\|\big(\sum_{j=1}^{d}\frac{\partial}{\partial x_j}(-\boldsymbol{A}_j\phi_t(\boldsymbol{x}_i^1;\theta_t)) + \boldsymbol{C}^\intercal\phi_t(\boldsymbol{x}_i^1;\theta_t)\big)\right\|_2^2,$$

where $(\cdot,\cdot)$ denotes the inner product of two vectors and $\|\cdot\|_2$ denotes the 2-norm of a vector. As for the boundary loss, let us take the Dirichlet boundary condition $u(\boldsymbol{x}) = g_d(\boldsymbol{x})$ as an example. In this case, the boundary loss can be formulated as

$$\hat{L}_b(\theta_s, \theta_t) := \frac{1}{N_2}\sum_{i=1}^{N_2}\left\|\phi_s((\boldsymbol{x}_i^2, \theta_s) - g_d(\boldsymbol{x}_i^2)\right\|_2^2.$$

If the solution DNN and test DNN are built to satisfy their boundary conditions automatically, $\hat{L}_b(\theta_s, \theta_t)$ is equal to zero and, hence, it is not necessary to compute this loss.

Next, we compute the gradient of $L_t(\theta_s, \theta_t)$ with respect to θ_s as the gradient descent direction and denote it as g_s. The gradient is evaluated via the autograd in PyTorch, which is essentially equivalent to a sequence of chain rules to compute the gradient since the loss is the composition of several simple functions with explicit formulas. Thus, θ_s can be updated by applying one step of gradient descend with a step size η_s

$$\theta_s \leftarrow \theta_s - \eta_s g_s. \tag{4.9}$$

In each outer iteration of Algorithm 1, we repeatedly sample new training points and update θ_s for n_s steps.

In each inner iteration, θ_t can be updated similarly to maximize the empirical loss $L_t(\theta_s, \theta_t)$. In each inner iteration for updating θ_t, we generate random samples, evaluate the gradient of the empirical loss with respect to θ_t, and denote it as g_t. Then θ_t can be updated via one step of gradient ascend with a step size η_t as follows:

$$\theta_t \leftarrow \theta_t + \eta_t g_t. \tag{4.10}$$

In each outer iteration of Algorithm 1, we repeatedly sample new training points and update θ_t for n_t steps.

We would like to emphasize that minimax optimization problems are in general more challenging to solve than minimization problems arising in network-based PDE solvers in the strong form. Note that, when the test DNN is fixed, the loss function in (4.1) is a convex functional of the solution DNN. Hence, the difficulty of the minimization problem when the test DNN is fixed is the same as the network-based least squares method. No matter what the test function is, the gradient descent update of θ_s can improve the solution DNN as long as the gradient is not zero and the step size is appropriate.

To further facilitate the convergence of Friedrichs learning, a restarting strategy is introduced to obtain the restarted Friedrichs learning in Algorithm 1, in the same spirit of typical restarted iterative solvers in numerical linear algebra, e.g., the restarted GMRES [40], or the restart strategies in optimization [2, 29, 50, 33]. Without loss of generality, the restarted Friedrichs learning is introduced for PDEs with Dirichlet boundary conditions. For other boundary conditions, restarted Friedrichs learning can be designed similarly.

Recall that the following formula is used to construct a solution DNN ϕ_s satisfying the Dirichlet boundary conditions automatically

$$\phi_s(\boldsymbol{x};\theta_s) = h(\boldsymbol{x})\hat{\phi}_s(\boldsymbol{x};\theta_s) + b(\boldsymbol{x}), \tag{4.11}$$

where $b(\boldsymbol{x})$ is a known function satisfying the boundary conditions and $h = 0$ on the boundary. If $b(\boldsymbol{x})$ in (4.11) is closer to the true solution, it is easier to train a generic DNN $\hat{\phi}_s$ such that ϕ_s can approximate the true solution accurately. Therefore, after a few rounds of outer iterations in the original Friedrichs learning, we have obtained a rough solutions DNN, which can serve as a better b function in (4.11) to construct a new solution DNN. Training on the new solution DNN could lead to a better solution. Repeatedly applying this idea completes the restarted Friedrichs learning for the solution DNN.

For a fixed solution DNN, the maximization problem over the test DNN is not convex both in the parameter space and in the DNN space. Therefore, it is challenging to obtain a best test DNN and it is beneficial to restart the test DNN with new random parameters when the solution DNN cannot be improved.

5. Numerical Experiments. In this section, the proposed Friedrich learning algorithm is tested on several PDE examples. In the beginning, we solve two advection equations even when the solution is not continuous to show the ability of Friedrichs learning to deal with low regularity. Second, we solve a high-dimensional parabolic equation to show the capacity of Friedrichs learning for high-dimensional problems. Third, we solve a wave equation to verify the performance of Friedrichs learning on complex domains. Finally, we solve a Maxwell equation system, a typical example of Friedrichs system.

For the experiments below, we have summarized all hyperparameters in Table 5.1. We set the solution DNN $\phi_s(\boldsymbol{x},\theta_s)$ as a fully connected ResNet with ReLU activation functions, depth 7, and width m_s, where m_s is problem dependent. The activation of $\phi_s(\boldsymbol{x},\theta_s)$ is chosen as ReLU because weak solutions might not even be continuous. ReLU DNNs may better approximate discontinuous functions than DNNs with other smooth activation functions. The test DNN $\phi_t(\boldsymbol{x},\theta_t)$ has the same structure with depth 7 and width m_t. To ensure the smoothness of $\phi_t(\boldsymbol{x},\theta_t)$, we adopt the Tanh activation function. The optimizers for updating $\phi_s(\boldsymbol{x},\theta_s)$ and $\phi_t(\boldsymbol{x},\theta_t)$ are chosen as Adam and RMSprop, respectively. All of our experiments share the same setting for network structure and optimizer. The learning rate in the optimization is adjusted in an exponentially decaying scheme. At the k-th iteration, we set the learning rate $\eta_s^{(k)}$ as

$$\eta_s^{(k)} = \eta_s^{(0)}(\frac{1}{10})^{(k/\nu_s)}, \quad \eta_t^{(k)} = \eta_t^{(0)}(\frac{1}{10})^{(k/\nu_t)},$$

Algorithm 1 Restarted Friedrichs Learning for Weak Solutions of PDEs.

Require: The desired PDE.

Ensure: Parameters θ_t and θ_s solving the minimax problem in (4.3).

Set iteration parameters n, n_s, and n_t. Set sample size parameters N_1 and N_2. Set step sizes $\eta_s^{(k)}$ and $\eta_t^{(k)}$ in the k-th outer iteration. Set the restart index set Θ_s and Θ_t.
Initialize $\phi_s(\boldsymbol{x};\theta_s^{0,0})$ and $\phi_t(\boldsymbol{x};\theta_t^{0,0})$.
for $k = 1, \cdots, n$ **do**
  if $k \in \Theta_s$ **then**
    Keep a copy $b(\boldsymbol{x}) = \phi_s(\boldsymbol{x}, \theta_s^{k-1,0})$ and randomly re-initialized $\theta_s^{k-1,0}$.
    if the penalty method for boundary conditions is used **then**
      Set a new DNN $\phi_s(\boldsymbol{x}, \theta_s^{k-1,0}) = \hat{\phi}_s(\boldsymbol{x}, \theta_s^{k-1,0}) + b(\boldsymbol{x})$ with a generic DNN $\hat{\phi}_s(\boldsymbol{x}, \theta_s^{k-1,0})$.
    else
      Set a new DNN $\phi_s(\boldsymbol{x}, \theta_s^{k-1,0}) = h(\boldsymbol{x})\hat{\phi}_s(\boldsymbol{x}, \theta_s^{k-1,0}) + b(\boldsymbol{x})$ with a generic DNN $\hat{\phi}_s(\boldsymbol{x}, \theta_s^{k-1,0})$ and $h(\boldsymbol{x})$ in (4.11).
    end if
  end if
  for $j = 1, \cdots, n_s$ **do**
    Generate uniformly distributed sample points $\{\boldsymbol{x}_i^1\}_{i=1}^{N_1} \subset \Omega$ and $\{\boldsymbol{x}_i^2\}_{i=1}^{N_2} \subset \partial\Omega$.
    Compute the gradient of the loss function in (4.8) at the point $(\theta_s^{k-1,j-1}, \theta_t^{k-1,0})$ with respect to θ_s and denote it as $g(\theta_s^{k-1,j-1}, \theta_t^{k-1,0})$.
    Update $\theta_s^{k-1,j} \leftarrow \theta_s^{k-1,j-1} - \eta_s^{(k)} g(\theta_s^{k-1,j-1}, \theta_t^{k-1,0})$ with a step size $\eta_s^{(k)}$.
  end for
  $\theta_s^{k,0} \leftarrow \theta_s^{k-1,n_s}$.
  If $k \in \Theta_t$, re-initialize $\theta_t^{k-1,0}$ randomly.
  for $j = 1, \cdots, n_t$ **do**
    Generate uniformly distributed sample points $\{\boldsymbol{x}_i^1\}_{i=1}^{N_1} \subset \Omega$ and $\{\boldsymbol{x}_i^2\}_{i=1}^{N_2} \subset \partial\Omega$.
    Compute the gradient of the loss function in (4.8) at $(\theta_s^{k,0}, \theta_t^{k-1,j-1})$ with respect to θ_t and denote it as $g(\theta_s^{k,0}, \theta_t^{k-1,j-1})$.
    Update $\theta_t^{k-1,j} \leftarrow \theta_t^{k-1,j-1} + \eta_t^{(k)} g(\theta_s^{k,0}, \theta_t^{k-1,j-1})$ with a step size $\eta_t^{(k)}$.
  end for
  $\theta_t^{k,0} \leftarrow \theta_t^{k-1,n_t}$.
  if Stopping criteria is satisfied **then**
    Return $\theta_s = \theta_s^{k,0}$ and $\theta_t = \theta_t^{k,0}$.
  end if
end for

where $\eta_s^{(0)}$ is the initial learning rates, ν_s is the decaying rate for the solution DNN. $\eta_t^{(k)}$ is set up similarly. The varying decaying rates are to prevent our learning rates from vanishing too fast, which might stop training before the network is fully trained.

Special networks satisfying boundary conditions automatically are used in all experiments so as to avoid tuning the parameters λ_1 and λ_2 in (4.2). The ε introduced in (4.1) is set as 1 and the inner iteration numbers $n_s = 1$ and $n_t = 1$ in all experiments. The value of other parameters listed in Table 5.1 will be specified later.

To measure the solution accuracy, the following discrete relative ℓ^2 error at uniformly distributed

Notation	Meaning
d	the dimension of the problem
n	the number of outer iterations
$\eta_s^{(0)}$	the initial learning rate for optimizing the solution network
$\eta_t^{(0)}$	the initial learning rate for optimizing the test network
ν_s	the decaying rate for η_s
ν_t	the decaying rate for η_t
m_s	the width of each layer in the solution network
m_t	the width of each layer in the test network
n_s	the number of inner iterations for the solution network
n_t	the number of inner iterations for the test network
N	the number of training points inside the domain
N_b	the number of training points on the domain boundary
ε	the gradient stabilizer introduced in (4.1)
Θ_s	the restart index set of the solution network
Θ_t	the restart index set of the solution network

TABLE 5.1
Parameters in the model and algorithm.

test points in domain is applied:

$$e_{\ell^2}(\theta_s) := \left(\frac{\sum_i \|\phi_s(\boldsymbol{x}_i;\theta_s) - u^*(\boldsymbol{x}_i)\|_2^2}{\sum_i \|u^*(\boldsymbol{x}_i)\|_2^2} \right)^{\frac{1}{2}},$$

where $\|\cdot\|_2$ denotes the 2-norm of a vector. In the case when the true solution is continuous, the following discrete relative ℓ^∞ error at uniformly distributed test points in domain is also applied:

$$e_{\max}(\theta_s) := \frac{\max_i(\|\phi_s(\boldsymbol{x}_i;\theta_s) - u^*(\boldsymbol{x}_i)\|_\infty)}{\max_i(\|u^*(\boldsymbol{x}_i)\|_\infty)},$$

where $\|\cdot\|_\infty$ denotes the ℓ^∞-norm of a vector. In all examples, we choose 10000 testing points for error evaluation.

5.1. Advection-Reaction Equation with Discontinuous Initial Value. In the first example, we identify the weak solution in $L^2(\Omega)$ of the advection-reaction equation in (3.1) with discontinuous initial value. Following Example 2 in [32], we choose the velocity $\beta = (1, 9/10)^\intercal$ and $\mu = 1$ in a domain $\Omega = (-1, 1)^2$. We choose the right-hand-side function f and the boundary function g such that the exact solution is

$$u(x,y) = \begin{cases} \sin(\pi(x+1)^2/4)\sin(\pi(y-\frac{9}{10}x)/2) & \text{for } -1 \le x \le 1,\ \frac{9}{10}x < y \le 1, \\ e^{-5(x^2+(y-\frac{9}{10}x)^2)} & \text{for } -1 \le x \le 1,\ -1 \le y < \frac{9}{10}x. \end{cases} \tag{5.1}$$

The exact solution is visualized in Figure 5.1(b). The discontinuity of the initial value function will propagate along the characteristic line $y = 9x/10$. Hence, the derivative of the exact solution does not exist along the characteristic line. Classical network-based least square algorithms in the strong form will encounter a large residual error near the characteristic line and hence fail to

provide a meaningful solution. This drawback of the classical method in the strong form motivates our Friedrichs learning in the weak form.

As discussed in [32], a priori knowledge of the characteristic line is crucial for conventional finite element methods with adaptive mesh to obtain high accuracy. In [32], the streamline diffusion method (SDFEM) can obtain a solution with $O(10^{-2})$ accuracy using $O(10^4)$ degree of freedom when the mesh is aligned with the discontinuity, i.e., when the priori knowledge of the characteristic line is used in the mesh generation. The discontinuous Galerkin method (DGFEM) in [32] can obtain $O(10^{-8})$ accuracy under the same setting. When the mesh is not aligned with the discontinuity, e.g., when the characteristic line is not used in mesh generation, DGFEM converge as slow as SDFEM and the accuracy is not better than $O(10^{-2})$ with $O(10^4)$ degree of freedom according to the discussion in [32].

Friedrichs learning is a mesh-free method and the weak solution is identified without the priori knowledge of the characteristic line. By the discussion in Section 4.3, a pecial network $\phi_s(\boldsymbol{x}, \theta_s)$ is constructed as follows e to fulfill the boundary condition of the solution:

$$\phi_s(\boldsymbol{x}, \theta_s) = \cos(-\frac{\pi}{4} + \frac{\pi}{4}x)\cos(-\frac{\pi}{4} + \frac{\pi}{4}y)\hat{\phi}_s(\boldsymbol{x}, \theta_s) + b(x, y), \tag{5.2}$$

where $b(x, y)$ is construct directly from the boundary condition as

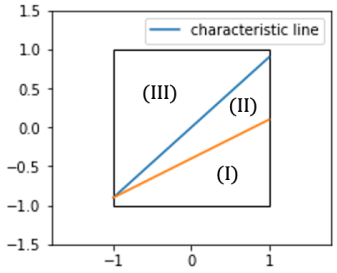
(5.3)
$$b(x,y) = \begin{cases} 0, & \text{for } -1 \le x \le 1,\ -0.4 + x/2 < y \le 1, \\ e^{-5[x^2+(-1-9/10x)^2]} + e^{-5[(-1)^2+(y+9/10)^2]} & \\ \quad -e^{-5[(-1)^2+(-1+9/10)^2]}, & \text{for } -1 \le x \le 1,\ -1 \le y \le -0.4 + x/2, \end{cases}$$

satisfying $b(x, y) = u(x, y)$ on $\partial\Omega^-$. For test function we also set similar structure so that $\phi_t(\boldsymbol{x}, \theta_t) = 0$ on $\partial\Omega^+$.

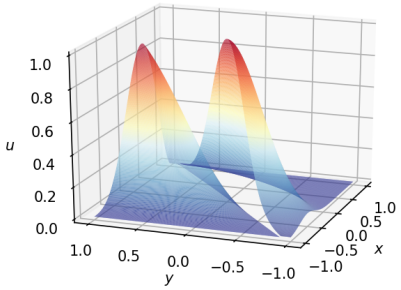
We have three important remarks about the choice of $b(x, y)$ above. First of all, the choice of $b(x, y)$ is to show that $b(x, y)$ is not necessarily close to the true solution. With the restarted Friedrichs learning, the numerical performance is not sensitive to the choice of the initial $b(x, y)$ as long as the boundary conditions are satisfied. We only restart the training of the solution network once to obtain a satisfactory solution. In the first round of training, we train a ResNet of width 50 for $2,000$ outer iterations to get a rough solution with an L^2 relative error $2.013e-1$. As shown in Figure 5.1(c), the rough solution has already captured the characteristic line. After restarting in the second round of training, $30,000$ outer iterations are enough to make the L^2 error of the solution DNN decrease to $1.746e-2$ as shown in Figure 5.1(f) and 5.1(g), which is comparable with the SDFEM in [32]. Note that the result in [32] uses the characteristic line while we do not. All other parameters are shown in Table 5.2.

Second, the choice of $b(x, y)$ above actually makes Friedrichs learning more challenging to train. The true solutions are discontinuous along the characteristic line, the blue line in Figure 5.1(a), and $b(x, y)$ is discontinuous along the orange line in Figure 5.1(a). Hence, to make the solution DNN ϕ_s in (5.2) approximate the true solution well, the DNN $\hat{\phi}_s$ in (5.2) needs to learn these two lines automatically and should be approximately discontinuous along these two lines. As shown by Figure 5.1(d), the solution DNN ϕ_s has a configuration similar to the true solution in Figure 5.1(b), which means that Friedrichs learning has successfully learned these two lines.

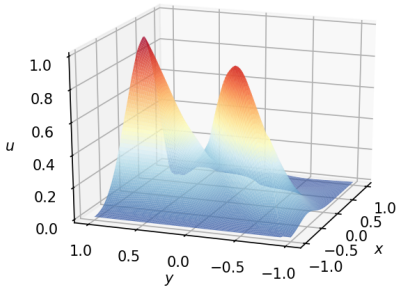
Third, to facilitate the convergence of the solution DNN, the test DNN ϕ_t should be large along these two lines in response to emphasize the error of the solution DNN ϕ_s, which can make the gradient descent change the configuration of ϕ_s near these two lines more dramatically than other places. As shown in Figure 5.1(e), the test DNN has a larger magnitude near these two lines agreeing with the expected functionality of the test DNN.



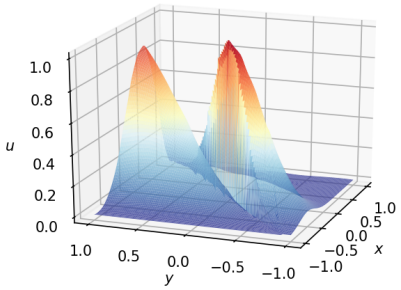
(a) The characteristic line (blue) and the line (orange) along which $b(x, y)$ in (5.3) is discontinuous.



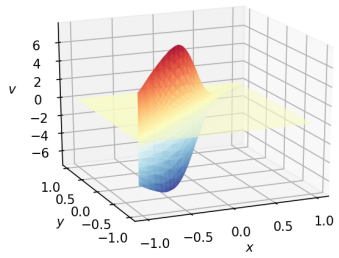
(b) Exact solution.



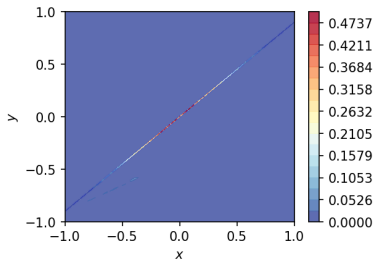
(c) The solution DNN right before restarting.



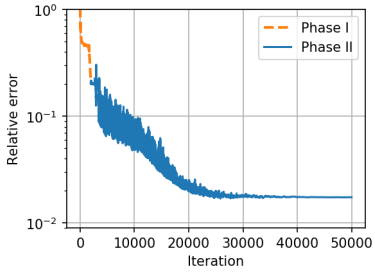
(d) The solution DNN after restarting at epoch 50,000.



(e) The test DNN at epoch 50,000.



(f) $|u^*(x) - \phi_s(\boldsymbol{x}, \theta_s)|$ at epoch 50,000.



(g) The relative error versus iteration.

FIG. 5.1. *Numerical results of Equation* (3.1) *when the exact solution is chosen as* (5.1).

5.2. Advection Equation in a Fan-Shaped Domain. The second experiment is to solve a two-dimensional advection-reaction equation (3.1) in a fan-shaped domain $\Omega = \{(x, y) \in \mathbb{R}^+ \times \mathbb{R}^+ : 0.1 \leq \sqrt{x^2 + y^2} \leq 1\}$, which is the numerical example considered by [11]. In this transport problem, the varying velocity is $\beta = \frac{1}{\sqrt{x^2+y^2}}(y, -x)^\intercal$, the reaction coefficient is $\mu = 0.01$, and the boundary conditions on $\Omega^- = \{(0, y), 0.1 \leq y \leq 1\}$ are given such that the exact solution is

$$u(x, y) = e^{\mu\sqrt{x^2+y^2}\arcsin(\frac{y}{\sqrt{x^2+y^2}})} \arctan\left(\frac{\sqrt{x^2 + y^2} - 0.5}{0.1}\right), \tag{5.4}$$

which is visualized in Figure 5.2(a). The same network structure and size as in Section 5.1 is applied in this example. The parameters used in this experiment are listed in Table 5.3. After 10,000 outer

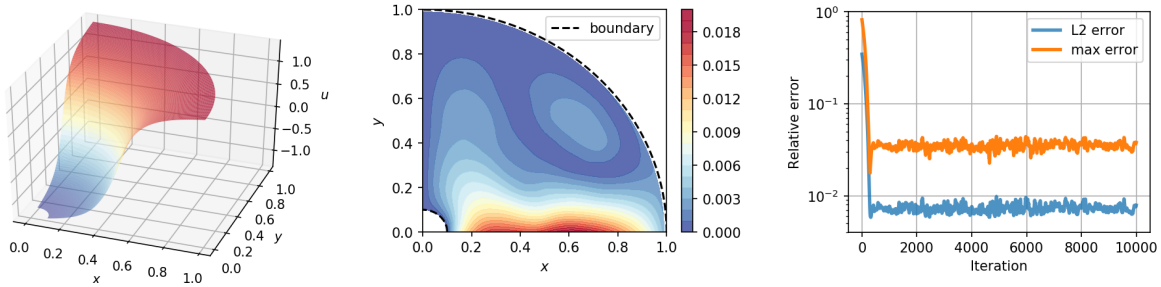
Parameters	N	N_b	$\eta_s^{(0)}$	$\eta_t^{(0)}$	ν_s
Value	$90k$	$10k$	3×10^{-4}	3×10^{-3}	$10k$
Parameters	ν_t	m_s	m_t	Θ_s	Θ_t
Value	$10k$	50 before epoch 2k, 250 after	150	$\{2k\}$	$\varnothing$

TABLE 5.2
The parameters of the experiment in Section 5.1.

Parameters	N	N_b	$\eta_s^{(0)}$	$\eta_t^{(0)}$	ν_s
Value	40000	10000	1×10^{-5}	1×10^{-3}	10000
Parameters	ν_t	m_s	m_t	Θ_s	Θ_t
Value	10000	250	150	$\varnothing$	$\{500 \cdot n, n \in \mathbb{Z}\}$

TABLE 5.3
The parameters of the implementation for experiment in Section 5.2.

iterations, we obtain a solution with an L^2 relative error $7.904e-3$ and an maximum relative error $3.803e-2$. The exact solution, the error contour, and the curve of error versus iteration are presented in Figure 5.2. We only show the results within $10,000$ outer iterations since the accuracy remains roughly the same afterwards. Note that we applied ReLU ResNet to approximate the solution. Our results are compatible with the results in [11], where polynomials of degree 1 are applied for FEM implementation.



(a) Exact solution $u^*(x)$.

(b) $|u^*(x) - \phi_s(\boldsymbol{x}, \theta_s)|$ after $10,000$ outer iterations.

(c) The relative error versus iterations.

FIG. 5.2. *Numerical results of the advection-reaction equation with an exact solution in* (5.4).

5.3. Advection-Reaction Equation with a Winding Characteristic Curve. Now we consider the advection-reaction equation (3.1) in a circle-shaped domain. Let $\Omega = \{(t,x,y)|0 \leq t \leq 1, x^2+y^2 \leq 1\}$, $\beta(t,x,y) = (1, -2\pi y, 2\pi x)$, $\mu = 0$, and $f = 0$. By (3.1) and the change of variable $v(t,x,y) = e^{-\mu' t}u(t,x,y)$, we get

$$(1,-2\pi y, 2\pi x)\nabla u = 0 \iff \mu' v + (1,-2\pi y, 2\pi x)\nabla v = 0.$$

Hence,

$$\beta \cdot \nabla v + \mu' v = 0 \quad \text{for } (t,x,y) \in \Omega. \tag{5.5}$$

In this example, μ' is chosen as 0.1 to enforce the coercive condition of the Friedrichs system.

Parameters	N	N_b	$\eta_s^{(0)}$	$\eta_t^{(0)}$	n_s	n_t
Value	40000	10000	1×10^{-4}	1×10^{-3}	1	1
Parameters	ν_s	ν_t	m_s	m_t	Θ_s	Θ_t
Value	15000	20000	250	150	$\varnothing$	$\varnothing$

TABLE 5.4
The parameters of the implementation for experiment in Section 5.3.

We will solve the above equation for v. For any $(t, x, y) \in \Omega$, the characteristic curve passing through it has a winding shape and centers at the origin. Therefore, the tangent line of the characteristic curve at any point is perpendicular to the corresponding normal vector of $\partial\Omega$. In this case, the inflow boundary $\partial\Omega^-$ is $\{t = 0\} \times \partial\Omega$, which means that the propagation of wave only depends on the initial value on the circle $\partial\Omega^-$. The outflow boundary $\partial\Omega^+$ is $\{t = 1\} \times \partial\Omega$ and we enforce the test DNN to be 0 on it. The initial value is choose as $v(0, x, y) = x^2 + (y - 0.5)^2$ such that the exact solution is

$$v(t, x, y) = e^{-\mu' t} v_0(x \cdot \cos(2\pi t) + y \cdot \sin(2\pi t), y \cdot \cos(2\pi t) - x \cdot \sin(2\pi t)).$$

The Friedrichs learning is applied to solve the above IVP. All parameters are summarized in Table 5.4. After 25,000 outer iterations, an approximation solution $\phi_s(\boldsymbol{x}, \theta_s)$ is identified with an L^2 relative error $2.564e-2$ showed in Figure 5.5(a) in comparison of the exact solution in Figure 5.3. The difference of the approximated solution and the exact solution at $t = 0$ and $t = 0.5$ is plotted in Figure 5.5(a) and the maximum absolute error is smaller than 0.1 as shown by Figure 5.5(b). We stop the iteration after 30,000 outer iterations since the solution is reasonable and the accuracy cannot be improved further.

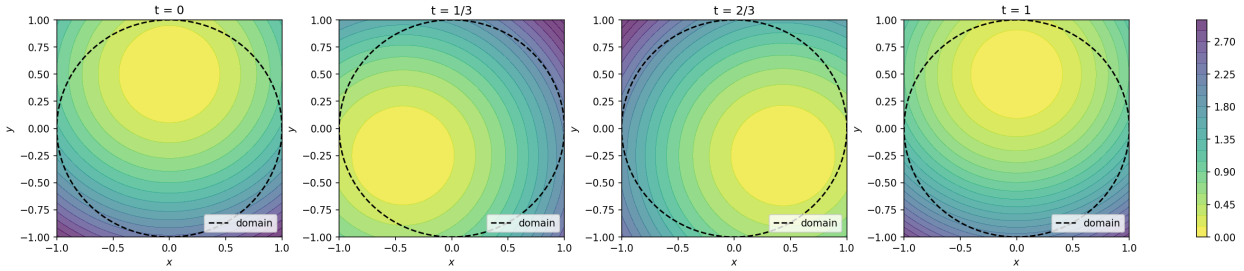


FIG. 5.3. *The contours of the exact solution $u^*(t, x)$ at different time instances.*

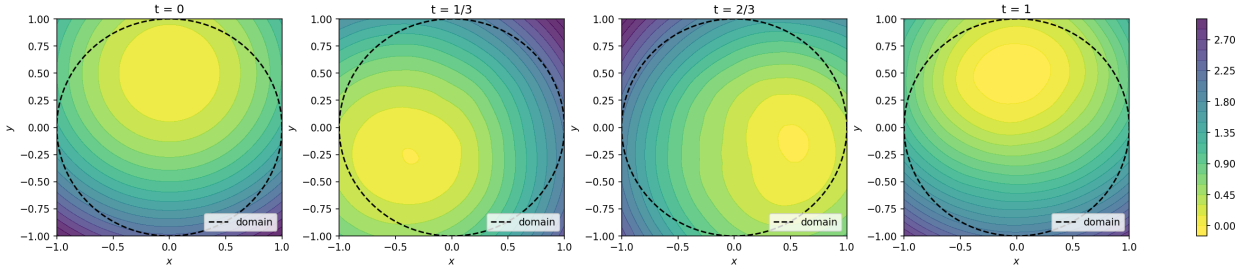
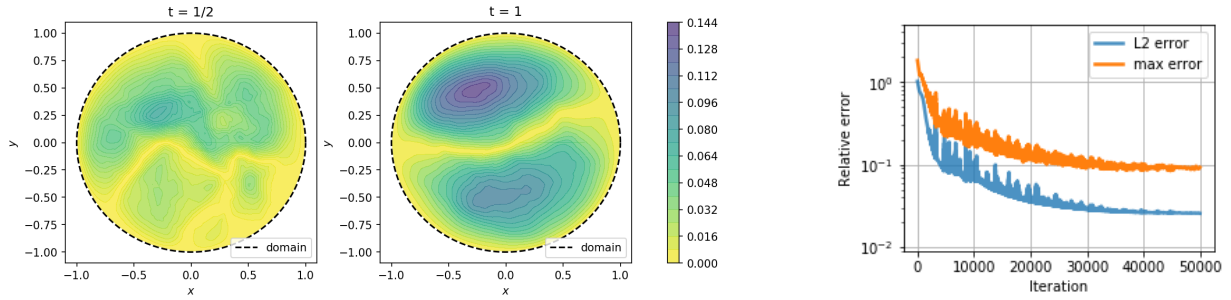


FIG. 5.4. *The contour of the approximate solution $\phi_s(\boldsymbol{x}, \theta_s)$ by Friedrichs learning at different time instances.*



(a) $|u^*(x) - \phi_s(\boldsymbol{x}, \theta_s)|$ after 25,000 outer iterations.

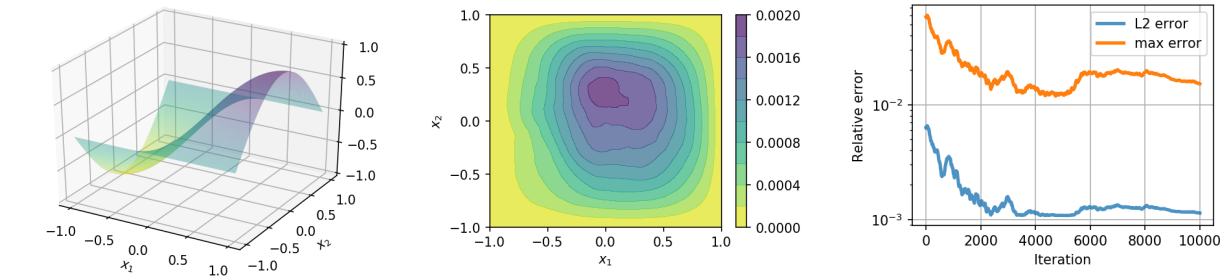
(b) The relative error v.s. iteration.

FIG. 5.5. *Numerical results of Equation* (3.1).

5.4. High-Dimensional Scalar Elliptic Equation. In the fourth experiment, we apply our algorithm to a 15D scalar elliptic equations with Dirichlet boundary conditions in high-dimension. Consider the following five-dimensional problem defined on $\Omega = (-1, 1)^{15}$:

$$\begin{cases} -\nabla \cdot \big(a(x)\nabla u\big) = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{5.6}$$

where $a(x) = 1+|x|^2$, $f(x) = \frac{\pi^2}{2}\cdot(1+|x|^2)\sin(\tilde{x}_1)\cos(\tilde{x}_2)+\pi x_2\sin(\tilde{x}_1)\sin(\tilde{x}_2)-\pi x_1\cos(\tilde{x}_1)\cos(\tilde{x}_2)$, and $g(x) = \sin(\tilde{x}_1)\cos(\tilde{x}_2)$ with $\tilde{x}_i = (\frac{\pi x_i}{2})$ for $i = 1, 2$. The exact solution of the above BVP is $u^*(x) = \sin(\tilde{x}_1)\cos(\tilde{x}_2)$ visualized in Figure 5.6(a). As discussed in Section 3.2, the above problem is first reduced to a system of first-order linear system and then solved by Friedrichs learning. The parameters used in this experiment are listed in Table 5.5. After 10,000 outer iterations, we obtain an L^2 relative error $1.136e-3$ and an L^∞ relative error $1.519e-2$ as shown in Figure 5.6(c). The contour of the difference between the exact solution $u^*(\boldsymbol{x})$ and the approximate solution $\phi_s(\boldsymbol{x}, \theta_s)$ is shown in left panel in Figure 5.6(b).



(a) The exact solution $u^*(x)$.

(b) $|u^*(x) - \phi_s(\boldsymbol{x}, \theta_s)|$ at 10,000 outer iterations.

(c) The relative error versus iteration.

FIG. 5.6. *Numerical results of Equation* (5.6).

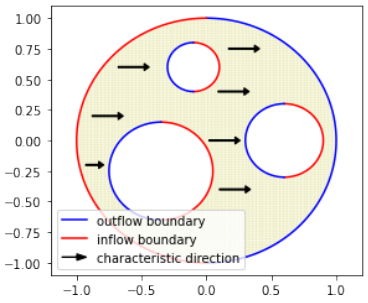
5.5. Wave Equation in a Complex Domain. We solve a three-dimensional wave equation in a complex domain shown in Figure 5.7(a). The boundary condition of the solution is set up on the inflow boundary (red lines) in Figure 5.7(a) and the boundary condition of the test function is on the outflow boundary (blue lines) in Figure 5.7(a). In particular, the wave equation is

$$\begin{cases} u_t + u_x = 0, & \text{for } (t, x, y) \in \Omega, \\ u = \sin(x + y), & \text{for } t = 0, \end{cases} \tag{5.7}$$

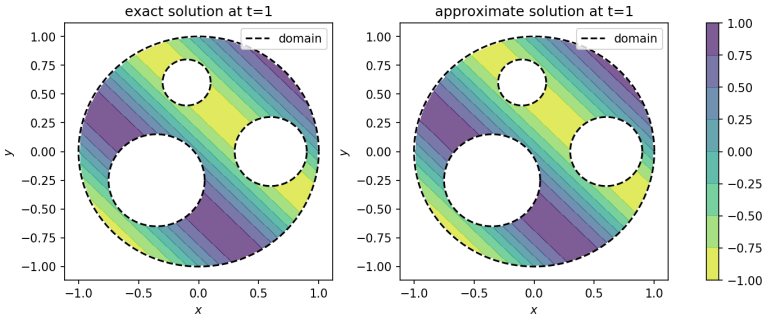
Parameters	N	N_b	$\eta_s^{(0)}$	$\eta_t^{(0)}$	ν_s	ν_t
Value	10000	10000	1×10^{-5}	1×10^{-4}	8000	8000
Parameters	m_s	m_t	Θ_t	n	-	-
Value	150	150	$\{500 \cdot n, n \in \mathbb{Z}\}$	10000	-	-

TABLE 5.5
The parameters of the experiment in Section 5.4.

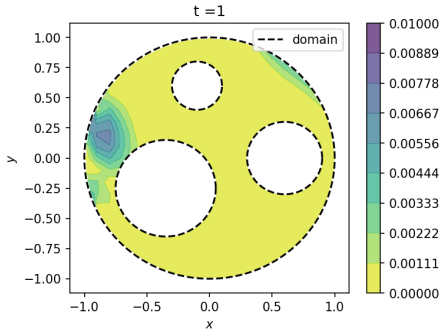
and its exact solution is $u(t,x) = \sin(x+y-t)$. By the change of variable $v = e^{-ct}u$ where $c = 0.1$ here, we can obtain and solve an equation in v satisfying the coercive condition. In this experiment, we choose parameters listed in Table 5.6. After $10,000$ outer iterations in Friedrichs learning, we obtain an approximate solution with an L^2 relative error $1.096e-3$ and an L^∞ relative error $7.690e-3$.



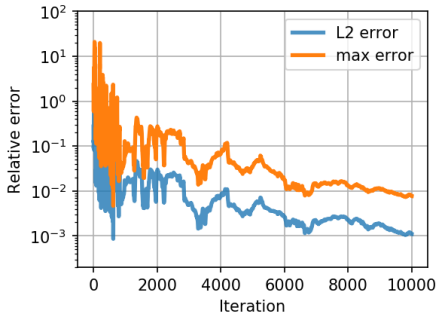
(a) The diagram of domain (yellow), the inflow boundary (red lines), the outflow boundary (blue lines), and the characteristic direction (arrows).



(b) The exact solution v.s the approximate solution at $t = 1$.



(c) The absolute difference between the exact solution and the approximate solution at $10,000$ outer iterations.



(d) The relative error versus iterations.

FIG. 5.7. *Numerical results of Equation* (5.7).

5.6. Maxwell Equations . In the last example, we solve Maxwell equations in (3.14) in $\mathbb{R}^3$. Let $(\boldsymbol{H}, \boldsymbol{E})^\intercal \in \Omega = [0,\pi]^3 \times [0,\pi]^3$ satisfy the following equations:

$$\begin{cases} \mu \boldsymbol{H} + \nabla \times \boldsymbol{E} = \boldsymbol{f}, \\ \sigma \boldsymbol{E} - \nabla \times \boldsymbol{H} = \boldsymbol{g}, \end{cases} \tag{5.8}$$

Parameters	N	N_b	$\eta_s^{(0)}$	$\eta_t^{(0)}$	ν_s
Value	10000	10000	5×10^{-4}	1×10^{-4}	5000
Parameters	ν_t	m_s	m_t	n	d
Value	5000	250	150	10000	15

TABLE 5.6
The parameters of the experiment in Section 5.5.

Parameters	N	N_b	$\eta_s^{(0)}$	$\eta_t^{(0)}$	ν_s	ν_t
Value	50000	0	3×10^{-6}	3×10^{-3}	8000	15000
Parameters	m_s	m_t	n	-	-	-
Value	250	50	20000	-	-	-

TABLE 5.7
The parameters of the experiment in Section 5.6.

where $\mu = \sigma = 1$ and $(\boldsymbol{f}, \boldsymbol{g})^\intercal \in [L^2(\Omega)]^3 \times [L^2(\Omega)]^3$ are set as:

$$\boldsymbol{f} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{g} = \begin{pmatrix} 3\sin y \sin z \\ 3\sin z \sin x \\ 3\sin x \sin y \end{pmatrix}.$$

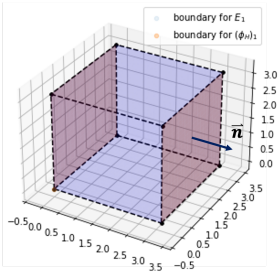
The boundary condition is set as $\boldsymbol{E} \times \boldsymbol{n} = 0$, which is an ideal conductor boundary condition. The solution to these equations are

$$\boldsymbol{H} = \begin{pmatrix} \sin x(\cos z - \cos y) \\ \sin y(\cos x - \cos z) \\ \sin x(\cos y - \cos x) \end{pmatrix} \quad \text{and} \quad \boldsymbol{E} = \begin{pmatrix} \sin y \sin z \\ \sin z \sin x \\ \sin x \sin y \end{pmatrix}.$$

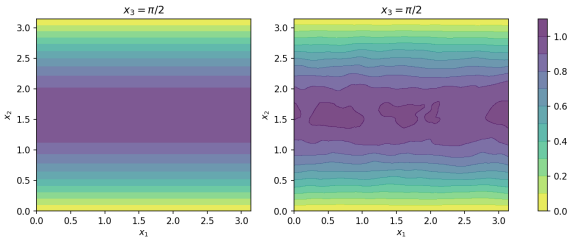
Considering test functions $(\varphi_{\boldsymbol{H}}, \varphi_{\boldsymbol{E}})^\intercal$ in the space $V^* = V$ mentioned in (3.4), we set up DNNs to guarantee the boundary conditions $\varphi_{\boldsymbol{E}} \cdot \boldsymbol{n} = 0$ and $\varphi_{\boldsymbol{H}} \times \boldsymbol{n} = 0$, where $\boldsymbol{n}$ is the normal vector of the boundary. Because the domain is a cube, the normal vector is parallel with one of the unit vectors. The boundary condition above is indeed a Dirichlet's boundary. For example, $S_1 = \{x = \pi\} \cap \partial\Omega$ on the right surface, which implies that $\boldsymbol{E}_2|_{S_1} = \boldsymbol{E}_3|_{S_1} = 0$. It is worth pointing out that the Dirichlet's boundary for $(\boldsymbol{E}_i, (\varphi_{\boldsymbol{H}})_i)$ closes the faces of the cube as shown in Figure 5.8(a).

To solve the Maxwell equations by Friedrich learning, we initialize two ResNets $\phi_s^{(\boldsymbol{H})}(\boldsymbol{x}, \theta_s^{(\boldsymbol{H})})$ and $\phi_s^{(\boldsymbol{E})}(\boldsymbol{x}, \theta_s^{(\boldsymbol{E})})$ with ReLU activation functions and two ResNets $\phi_t^{(\boldsymbol{H})}(\boldsymbol{x}, \theta_t^{(\boldsymbol{H})})$ and $\phi_t^{(\boldsymbol{E})}(\boldsymbol{x}, \theta_t^{(\boldsymbol{E})})$ with Tanh activation functions corresponding to four vector functions $(\boldsymbol{H}, \boldsymbol{E})^\intercal$ and $(\varphi_{\boldsymbol{H}}, \varphi_{\boldsymbol{E}})^\intercal$. Because we have vector functions, we set a ResNet, e.g., $\phi_s^{(\boldsymbol{H})}(\boldsymbol{x}, \theta_s^{(\boldsymbol{H})})$, using three sub-networks with width m_s and depth 7. Each sub-network decide one output value of the vector function. The other three networks are set up similarly. We list all the parameters used in this experiment in Table 5.7. After $20,000$ outer iterations, we obtain an L^2 relative error $1.766e-2$ and an L^∞ relative error $3.467e-2$.

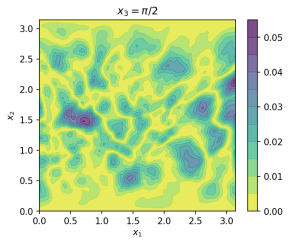
6. Conclusion. This paper proposed Friedrichs learning as a novel deep learning methodology that can learn the weak solutions of PDEs via Friedrichs' seminal minimax formulation, which transforms the PDE problem into a minimax optimization problem to identify weak solutions. The weak solution and the test function in the weak formulation are parameterized as deep neural networks in a mesh-free manner, which are alternately updated to approach the optimal solution



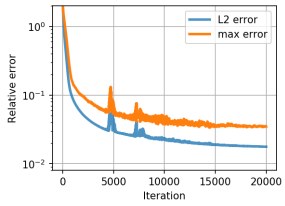
(a) The boundary conditions of $(\boldsymbol{E}_1, (\varphi_{\boldsymbol{H}})_1)$.



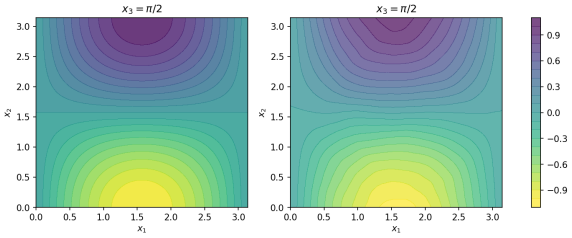
(b) The exact solution of $\boldsymbol{E}_1$ (left) and the corresponding numerical solution $(\phi_{\boldsymbol{E}})_1$ (right).



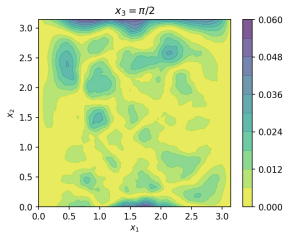
(c) The absolute difference between $\boldsymbol{E}_1$ and $(\phi_{\boldsymbol{E}})_1$ after 5,000 outer iterations.



(d) The relative error versus iterations.



(e) The exact solution of $\boldsymbol{H}_1$ (left) and the corresponding numerical solution $(\phi_{\boldsymbol{H}})_1$ (right).



(f) The absolute difference between $\boldsymbol{H}_1$ and $(\phi_{\boldsymbol{H}})_1$ after 5,000 outer iterations.

FIG. 5.8. *Numerical results of Maxwell quations in* (5.8).

networks approximating the weak solution and the optimal test function, respectively. Extensive numerical results indicate that our mesh-free method can provide reasonably good solutions to a wide range of PDEs defined on regular and irregular domains in various dimensions, where classical numerical methods such as finite difference methods and finite element methods may be tedious or difficult to be applied. Especially in the case when the solution is discontinuous, Friedrichs learning can infer the solution via deep learning without the knowledge of the location of discontinuity.

REFERENCES

[1] J.-L. Guermond A. Ern and G. Caplain. An intrinsic criterion for the bijectivity of hilbert operators related to friedrichs systems. *Comm. Partial Differential Equations*, 32(05):317341, 2007.

[2] Abdullah Al-Dujaili, Shashank Srikant, Erik Hemberg, and Una-May OReilly. On the application of danskins theorem to derivative-free minimax problems. *AIP Conference Proceedings*, 2070(1):020026, 2019.

[3] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[4] N. Antonic and K. Burazin. Graph spaces of first-order linear partial differential operators. *Math. Commun.*, 14:135155, 2009.

[5] Nenad Antoni and Kreimir Burazin. Intrinsic boundary conditions for friedrichs systems. *Communications in Partial Differential Equations*, 35(9):1690–1715, 2010.

[6] Gang Bao, Xiaojing Ye, Yaohua Zang, and Haomin Zhou. Numerical solution of inverse problems by weak adversarial networks. *Inverse Problems*, 36(11):115003, nov 2020.

[7] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.

[8] Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. Deep splitting method for parabolic PDEs. *arXiv e-prints*, arXiv:1907.03452, Jul 2019.

[9] Jens Berg and Kaj Nystrm. A unified deep artificial neural network approach to partial differential equations

in complex geometries. *Neurocomputing*, 317:28 – 41, 2018.

[10] TAN BUI-THANH, LESZEK DEMKOWICZ, and OMAR GHATTAS. A UNIFIED DISCONTINUOUS PETROVGALERKIN METHOD AND ITS ANALYSIS FOR FRIEDRICHS SYSTEMS. *SIAM J. NUMER. ANAL.*, page 19331958, 2013.

[11] Erik Burman, Alfio Quarteroni, and Benjamin Stamm. Interior penalty continuous and discontinuous finite element approximations of hyperbolic equations. *J. Sci. Comput*, pages 10–1007, 2008.

[12] Wei Cai and Zhi-Qin John Xu. Multi-scale Deep Neural Networks for Solving High Dimensional PDEs. *arXiv e-prints*, arXiv:1910.11710, Oct 2019.

[13] Constantinos Daskalakis and Ioannis Panageas. The limit points of (optimistic) gradient descent in min-max optimization. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 9256–9266, USA, 2018. Curran Associates Inc.

[14] M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

[15] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, Dec 2017.

[16] Weinan E, Chao Ma, and Lei Wu. Barron Spaces and the Compositional Function Spaces for Neural Network Models. *arXiv e-prints*, arXiv:1906.08039, Jun 2019.

[17] Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.*, 6:1–12, 2018.

[18] Matthias Ehrhardt and Ronald E. Mickens. A fast, stable and accurate numerical method for the blackscholes equation of american options. *International Journal of Theoretical and Applied Finance*, 11(05):471–501, 2008.

[19] A. Ern and J. L. Guermond. Discontinuous Galerkin methods for Friedrichs systems. Part I. General theory. *SIAM J. Numer. Anal.*, page 753778, 2006.

[20] A. Ern and J. L. Guermond. Discontinuous Galerkin methods for Friedrichs systems. Part II. Second-order elliptic PDEs. *SIAM J. Numer. Anal.*, page 23632388, 2006.

[21] A. Ern and J. L. Guermond. Discontinuous Galerkin methods for Friedrichs systems. Part III. Multifield theories with partial coercivity. *SIAM J. Numer. Anal.*, page 776 804, 2008.

[22] Alexandre Ern, Jean-Luc Guermond, and Gilbert Caplain. An intrinsic criterion for the bijectivity of hilbert operators related to friedrich' systems. *Communications in Partial Differential Equations*, 32(2):317–341, 2007.

[23] K. O. Friedrichs. Symmetric positive linear differential equations. *Communications on Pure and Applied Mathematics*, 11(3):333–418, 1958.

[24] Abhijeet Gaikwad and Ioane Muni Toke. Gpu based sparse grid technique for solving multidimensional options pricing pdes. In *Proceedings of the 2Nd Workshop on High Performance Computational Finance*, WHPCF '09, pages 6:1–6:9, New York, NY, USA, 2009. ACM.

[25] D. Gobovic and M. E. Zaghloul. Analog cellular neural network with application to partial differential equations with variable mesh-size. In *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, volume 6, pages 359–362 vol.6, May 1994.

[26] Yiqi Gu, Chunmei Wang, and Haizhao Yang. Structure probing neural network deflation, 2020.

[27] Yiqi Gu, Haizhao Yang, and Chao Zhou. Selectnet: Self-paced learning for high-dimensional partial differential equations, 2020.

[28] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[29] Kenta Hanada, Takayuki Wada, and Yasumasa Fujisaki. *A Restart Strategy with Time Delay in Distributed Minimax Optimization*, pages 89–100.

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[31] Geoffrey Hinton. Neural Networks for Machine Learning - Lecture 6a - Overview of mini-batch gradient descent., 2012.

[32] Paul Houston, Christoph Schwab, and Endre Sli. Stabilized hp-finite element methods for first-order hyperbolic problems. *SIAM Journal on Numerical Analysis*, 37(5):1618–1643, 2000.

[33] X. Hu, R. Shonkwiler, and M. Spruill. Random restarts in global optimization. 2009.

[34] Jianguo Huang, Haoqin Wang, and Haizhao Yang. Int-deep: A deep learning initialized iterative method for nonlinear problems. *Journal of Computational Physics*, 419:109675, 2020.

[35] M. Hutzenthaler, A. Jentzen, Th. Kruse, and T. A. Nguyen. A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations. Technical report, 2020.

[36] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, and Tuan Anh Nguyen. A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations. *arXiv e-prints*, arXiv:1901.10854, Jan 2019.

[37] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, Tuan Anh Nguyen, and Philippe von Wurstemberger. Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations. *arXiv e-prints*, arXiv:1807.01212, Jul 2018.

[38] Martin Hutzenthaler, Arnulf Jentzen, and von Wurstemberger Wurstemberger. Overcoming the curse of dimensionality in the approximative pricing of financial derivatives with default risks. *Electron. J. Probab.*, 25:73 pp., 2020.

[39] M. Jensen. Discontinuous Galerkin Methods for Friedrichs Systems with Irregular Solutions. *Ph.D. thesis, University of Oxford, Oxford*, 2004.

[40] Wayne Joubert. On the convergence behavior of the restarted gmres algorithm for solving nonsymmetric linear systems. *Numerical Linear Algebra with Applications*, 1(5):427–447, 1994.

[41] YUEHAW KHOO, JIANFENG LU, and LEXING YING. Solving parametric pde problems with artificial neural networks. *European Journal of Applied Mathematics*, page 115, 2020.

[42] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, 2014.

[43] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, Sep. 1998.

[44] Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110 – 131, 1990.

[45] T.T. Lee, F.Y. Wang, and R.B. Newell. Robust model-order reduction of complex biological processes. *Journal of Process Control*, 12(7):807 – 821, 2002.

[46] Ke Li, Kejun Tang, Tianfan Wu, and Qifeng Liao. D3M: A deep domain decomposition method for partial differential equations. *arXiv e-prints*, arXiv:1909.12236, Sep 2019.

[47] Qianxiao Li, Bo Lin, and Weiqing Ren. Computing committor functions for the study of rare events using deep learning. *The Journal of Chemical Physics*, 151(5):054112, 2019.

[48] Hadrien Montanelli and Haizhao Yang. Error bounds for deep ReLU networks using the Kolmogorov–Arnold superposition theorem. *arXiv e-prints*, arXiv:1906.11945, Jun 2019.

[49] Hadrien Montanelli, Haizhao Yang, and Qiang Du. Deep ReLU networks overcome the curse of dimensionality for bandlimited functions. *arXiv e-prints*, arXiv:1903.00735, Mar 2019.

[50] Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D. Lee, and Meisam Razaviyayn. Solving a class of non-convex min-max games using iterative first order methods. pages 14905–14916, 2019.

[51] Hassan Rafique, Mingrui Liu, Qihang Lin, and Tianbao Yang. Non-convex min-max optimization: Provable algorithms and applications in machine learning. *ArXiv*, abs/1810.02060, 2018.

[52] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686 – 707, 2019.

[53] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network with approximation error being reciprocal of width to power of square root of depth. *arXiv:2006.12231*, 2020.

[54] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Neural network approximation: Three hidden layers are enough. *arxiv:2010.14075*, 2020.

[55] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339 – 1364, 2018.

[56] Christopher Srinivasa, Inmar Givoni, Siamak Ravanbakhsh, and Brendan J Frey. Min-max propagation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5565–5573. Curran Associates, Inc., 2017.

[57] David J. Wales and Jonathan P. K. Doye. Stationary points and dynamics in high-dimensional systems. *The Journal of Chemical Physics*, 119(23):12409–12416, 2003.

[58] H. Yserentant. Sparse grid spaces for the numerical solution of the electronic schrödinger equation. *Numerische Mathematik*, 101(2):381–389, Aug 2005.

[59] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak Adversarial Networks for High-dimensional Partial Differential Equations. *arXiv e-prints*, arXiv:1907.08272, Jul 2019.