A deep neural network-based method for solving obstacle problems

Article · August 2023

DOI: 10.1016/j.nonrwa.2023.103864

Shen Xing
Zhejiang University
7 PUBLICATIONS **114** CITATIONS

Nonlinear Analysis: Real World Applications 72 (2023) 103864

Contents lists available at ScienceDirect

Nonlinear Analysis: Real World Applications

www.elsevier.com/locate/nonrwa

A deep neural network-based method for solving obstacle problems

Xiaoliang Cheng, Xing Shen, Xilu Wang, Kewei Liang [*]

School of Mathematical Sciences, Zhejiang University, Hangzhou, Zhejiang, PR China

ARTICLE INFO

Article history:
Received 2 October 2022
Received in revised form 14 February 2023
Accepted 16 February 2023
Available online xxxx

Keywords:
Obstacle problems
Deep neural networks
Error analysis
Minimization optimization

ABSTRACT

In this paper, we propose a method based on deep neural networks to solve obstacle problems. By introducing penalty terms, we reformulate the obstacle problem as a minimization optimization problem and utilize a deep neural network to approximate its solution. The convergence analysis is established by decomposing the error into three parts: approximation error, statistical error and optimization error. The approximate error is bounded by the depth and width of the network, the statistical error is estimated by the number of samples, and the optimization error is reflected in the empirical loss term. Due to its unsupervised and meshless advantages, the proposed method has wide applicability. Numerical experiments illustrate the effectiveness and robustness of the proposed method and verify the theoretical proof.

1. Introduction

Obstacle problems are typical variational inequalities of the first kind and have received much attention due to the wide range of applications. Fig. 1 illustrates the unilateral obstacle problem for an elastic membrane. Let $\Omega \subset \mathbb{R}^d$ be a bounded Lipschitz domain with the boundary $\partial\Omega$. The obstacle problem is to find the equilibrium position u of an elastic membrane under the action of the vertical force f. The membrane is fixed on the boundary $\partial\Omega$ with the function h and must lie over the obstacle g with $g \le h$ on $\partial\Omega$. The differential form of the obstacle problem is

$$\begin{cases} -\Delta u \ge f & \text{in } \Omega, \\ u \ge g & \text{in } \Omega, \\ (-\Delta u - f)(u-g) = 0 & \text{in } \Omega, \\ u = h & \text{on } \partial\Omega. \end{cases} \tag{1}$$

[*] Corresponding author.
E-mail addresses: xiaoliangcheng@zju.edu.cn (X. Cheng), shenxingsx@zju.edu.cn (X. Shen), wangxilu95@163.com (X. Wang), matlkw@zju.edu.cn (K. Liang).

https://doi.org/10.1016/j.nonrwa.2023.103864
1468-1218/© 2023 Elsevier Ltd. All rights reserved.

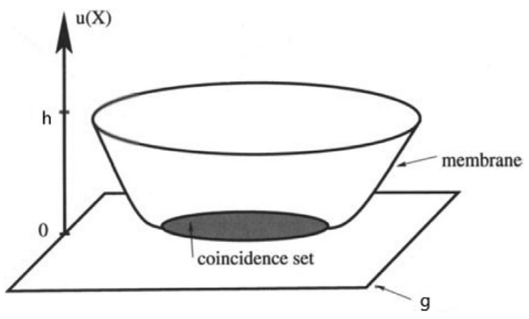


Fig. 1. Membrane over a plate obstacle.

Various numerical methods have been proposed for solving obstacle problems, the vast majority of which focus on approximation solutions to the weak variational inequality, such as Galerkin least squares finite element method [1], multigrid algorithm [2], piecewise linear iterative algorithm [3], the first-order least-squares method [4], the level set method [5], and dynamical functional particle method [6].

Recently, there has been a growing interest in solving differential equations and inverse problems by deep learning [7–18]. For variational and semi-variational inequalities, deep learning-based solution methods are less available. In [19], the author converts the original obstacle problem into an equivalent energy functional minimum problem. The initial and boundary value conditions are transformed into the loss function of the deep neural network through Lagrange multipliers. In [20], the authors proposed a deep learning-based method to solve elliptic hemivariational inequalities based on equivalent variational forms and compared the numerical performance of three different parameter update training strategies. However, these works mainly focus on computational methods and lack theoretical analysis.

In this work, we propose a deep learning-based approach to solving the obstacle problem. First, a minimal optimization problem is constructed based on a variational form of the obstacle problem and then solved by deep learning methods. In particular, we provide a theoretical analysis of the proposed deep learning method and establish the rate of convergence for a deep neural network with $ReLU^3$ activation functions in the H_1 norm. The error of the deep learning method is decomposed into three parts: the approximation error, the statistical error, and the optimization error. The approximate error is related to the depth and width of the network, the statistical error is estimated by the sample size of the Rademacher complexity tool, and the optimization error reflects the experience loss term in numerical experiments. We established upper bounds on the number of training samples, the depth, and the width of the network to achieve the desired accuracy. Moreover, numerical experiments illustrate the effectiveness and robustness of the proposed method and verify the theoretical proof.

This paper is organized as follows. In Section 2, we decompose the error into three components. The bounds of the approximation error and the statistical error are discussed. The results of the error analysis are established in the main theorem. Moreover, we also present the pipeline of the optimization algorithm. In Section 3, three numerical examples are provided to demonstrate the proposed method. In Section 4, we summarize our work with a short conclusion.

2. Error analysis of obstacle problems

Let $V = H^1(\Omega)$ and $U = \{v \in V : v = h \text{ on } \partial\Omega\}$. We denote by $K = \{v \in V : v \geq g \text{ in } \Omega, v = h \text{ on } \partial\Omega\}$ as the set of admissible displacements. Assume that $f \in L_\infty(\Omega)$, $g \in W^2_\infty(\Omega)$, $h \in W^2_\infty(\partial\Omega)$. Problem (1) can be rewritten in the form of energy minimization,

$$\text{Find } u \in K : \quad J(u) \leq J(v), \quad \forall v \in K, \tag{2}$$

where $J(v) = a(v,v) - (f,v)$ with the quadratic form $a(v,v) = \frac{1}{2}\int_\Omega |\nabla v|^2 \, dx$.

The solution of (2) is characterized by the elliptic variational inequality

$$u \in K : \quad \int_{\Omega} \nabla u \cdot \nabla(v-u)\mathrm{d}x \geq \int_{\Omega} f(v-u)\mathrm{d}x, \quad \forall v \in K. \tag{3}$$

Let

$$\tilde{\mathcal{L}}(v) = J(v) + \alpha \int_{\Omega} [g(x) - v(x)]_+^2 \mathrm{d}x, \tag{4}$$

where α is a positive constant and $[t]_+ = \max\{0, t\}$. From [21], the unique solution u^* of (4) is sufficiently close to the solution of (2) with sufficiently large α. We relax the constraint of the boundary condition $v \in U$ with a penalty term and have the following minimization problem:

$$\min_{v \in V} \mathcal{L}(v), \tag{5}$$

where

$$\mathcal{L}(v) = \frac{1}{2}\int_{\Omega} |\nabla v|^2 dx - \int_{\Omega} fv dx + \alpha \int_{\Omega} [g-v]_+^2 dx + \beta \int_{\partial\Omega} (v-h)^2 dx,$$

α and β are positive constants. In numerical computations, the above $\mathcal{L}$ is replaced by the discrete form

$$\begin{aligned}\widehat{\mathcal{L}}(v) = {} & \frac{|\Omega|}{N}\sum_{i=1}^{N}\left[\frac{1}{2}\|\nabla v(X_i)\|_2^2 - f(X_i)v(X_i) + \alpha[g(X_i) - v(X_i)]_+^2\right] \\ & + \frac{\beta|\partial\Omega|}{M}\sum_{j=1}^{M}[v(Y_j) - h(Y_j)]^2,\end{aligned} \tag{6}$$

where $\{X_k\}_{k=1}^N$ and $\{Y_k\}_{k=1}^M$ are i.i.d. random variables drawn from the uniform distributions $U(\Omega)$ and $U(\partial\Omega)$, respectively.

2.1. Deep neural network approximation

To approximate the solution, we consider a fully connected feedforward neural network $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^{N_{\mathcal{D}}}$, which is defined as

$$\begin{aligned}&\mathbf{f}_0(\mathbf{x}) = \mathbf{x},\\ &\mathbf{f}_k(\mathbf{x}) = \boldsymbol{\sigma}^{(k)}\left(\mathbf{W}_k\mathbf{f}_{k-1} + \mathbf{b}_k\right) = \boldsymbol{\sigma}_i^{(k)}\left(\left(\mathbf{W}_k\mathbf{f}_{k-1} + \mathbf{b}_k\right)_i\right), \quad \text{for} \quad k = 1, \ldots, \mathcal{D}-1,\\ &\mathbf{f} := \mathbf{f}_{\mathcal{D}}(\mathbf{x}) = \mathbf{W}_{\mathcal{D}}\mathbf{f}_{\mathcal{D}-1} + \mathbf{b}_{\mathcal{D}},\end{aligned}$$

where the weights matrix $\mathbf{W}_k = \left(w_{ij}^{(k)}\right) \in \mathbb{R}^{N_k \times N_{k-1}}$, the bias term $\mathbf{b}_k = \left(b_i^{(k)}\right) \in \mathbb{R}^{N_k}$ and the activation function $\boldsymbol{\sigma}^{(k)} = \left(\sigma_i^{(k)}\right) \in \mathbb{R}^{N_k}$. The trainable parameters $\mathbf{W}_k$ and $\mathbf{b}_k$ can be updated during training by the backpropagation algorithm. The activation function introduces nonlinearity, which makes the deep neural network to be a universal function approximator. We denote by $\mathcal{D}$ the depth of the network, by $\mathcal{W} := \max_{k=1,\ldots,\mathcal{D}}\{N_k\}$ the width of the network, and by Φ the set of activation functions. Neural networks with $\mathcal{D}, \mathcal{W}, \Phi$ are defined as $\mathcal{N}(\mathcal{D}, \mathcal{W}, \Phi) := \{\mathbf{f} :$ the fully connected feedforward neural network with the depth $\mathcal{D}$, the width $\mathcal{W}$, and the activation function set $\Phi\}$.

The set $\mathcal{P}$ is all neural networks, that is,

$$\mathcal{P} = \bigcup_{\mathcal{D}, \mathcal{W}, \Phi} \mathcal{N}(\mathcal{D}, \mathcal{W}, \Phi).$$

Thus, the obstacle problem amounts to finding u_θ such that

$$u_\theta = \arg\min_{v_\theta \in \mathcal{P}} \mathcal{L}(v_\theta), \tag{7}$$

where $\theta = \{\mathbf{W}_k, \mathbf{b}_k\}_{k=1}^{\mathcal{D}}$ is the collection of trainable parameters.

2.2. Error decomposition

In this section, we consider the error between the approximation $u_{\theta_{\mathcal{A}}}$ and the solution u^*. The subscript of $u_{\theta_{\mathcal{A}}}$ refers to the optimization algorithm $\mathcal{A}$ used in training networks. The following lemma divides the error into three parts [22,23].

Lemma 1. *If the fully connected feedforward neural networks are adopted to solve the obstacle problem, then*

$$\begin{aligned}&\left\|u_{\theta_{\mathcal{A}}}-u^*\right\|_{H^1(\Omega)}^2\\ &\le\frac{2}{c_2}\left[\underbrace{(\alpha+\beta c_1)\cdot\inf_{\bar u\in\mathcal{P}}\|\bar u-u^*\|_{H^1(\Omega)}^2}_{\mathcal{E}_{app}}+\underbrace{2\sup_{u\in\mathcal{P}}|\mathcal{L}(u)-\widehat{\mathcal{L}}(u)|}_{\mathcal{E}_{sta}}+\underbrace{\widehat{\mathcal{L}}\left(u_{\theta_{\mathcal{A}}}\right)-\widehat{\mathcal{L}}\left(\widehat{u}_\theta\right)}_{\mathcal{E}_{opt}}\right],\end{aligned}$$

where c_1, c_2 are constants, $\widehat{\mathcal{L}}$ is defined in (6) and $\widehat{u}_\theta$ is the optimal solution of $\widehat{\mathcal{L}}$.

Proof. For any $\bar u\in\mathcal{P}$, we have

$$\begin{aligned}&\mathcal{L}\left(u_{\theta_{\mathcal{A}}}\right)-\mathcal{L}\left(u^*\right)\\ &=\mathcal{L}\left(u_{\theta_{\mathcal{A}}}\right)-\widehat{\mathcal{L}}\left(u_{\theta_{\mathcal{A}}}\right)+\widehat{\mathcal{L}}\left(u_{\theta_{\mathcal{A}}}\right)-\widehat{\mathcal{L}}\left(\widehat{u}_\theta\right)+\widehat{\mathcal{L}}\left(\widehat{u}_\theta\right)-\widehat{\mathcal{L}}(\bar u)\\ &\quad+\widehat{\mathcal{L}}(\bar u)-\mathcal{L}(\bar u)+\mathcal{L}(\bar u)-\mathcal{L}\left(u^*\right)\\ &\le\left[\mathcal{L}(\bar u)-\mathcal{L}\left(u^*\right)\right]+2\sup_{u\in\mathcal{P}}|\mathcal{L}(u)-\widehat{\mathcal{L}}(u)|+\left[\widehat{\mathcal{L}}\left(u_{\theta_{\mathcal{A}}}\right)-\widehat{\mathcal{L}}\left(\widehat{u}_\theta\right)\right],\end{aligned}$$

where we use that $\widehat{\mathcal{L}}\left(\widehat{u}_\theta\right)-\widehat{\mathcal{L}}(\bar u)\le 0$ in the last step. Since $\bar u$ can be any element in $\mathcal{P}$, we take the infimum of $\bar u$:

$$\begin{aligned}\mathcal{L}\left(u_{\theta_{\mathcal{A}}}\right)-\mathcal{L}\left(u^*\right)\le&\inf_{\bar u\in\mathcal{P}}\left[\mathcal{L}(\bar u)-\mathcal{L}\left(u^*\right)\right]+2\sup_{u\in\mathcal{P}}|\mathcal{L}(u)-\widehat{\mathcal{L}}(u)|\\ &+\left[\widehat{\mathcal{L}}\left(u_{\theta_{\mathcal{A}}}\right)-\widehat{\mathcal{L}}\left(\widehat{u}_\theta\right)\right].\end{aligned}$$

Now for any $u\in\mathcal{P}$, we set $v=u-u^*$, and have

$$\begin{aligned}\mathcal{L}(u^*+v)=&\frac12\int_\Omega(\nabla(u^*+v))^2dx+\alpha\int_\Omega[g-(u^*+v)]_+^2dx-\int_\Omega f(u^*+v)dx\\ &+\beta\int_{\partial\Omega}(u^*+v-h)^2dx\\ =&\frac12\int_\Omega(\nabla u^*)^2dx+\frac12\int_\Omega(\nabla v)^2dx+\int_\Omega\nabla u^*\cdot\nabla vdx\\ &+\alpha\int_\Omega[g-(u^*+v)]_+^2dx-\int_\Omega fu^*dx-\int_\Omega fvdx+\beta\int_{\partial\Omega}(v^2)dx\\ =&\mathcal{L}(u^*)+\frac12|v|_1^2+\beta\|Tv\|_{L^2(\partial\Omega)}^2-\alpha\int_\Omega[g-u^*]_+^2dx\\ &+2\alpha\int_\Omega(g-u^*)_+vdx+\alpha\int_\Omega[g-(u^*+v)]_+^2dx,\end{aligned}$$

where the operator T refers to the trace operator. The last step holds since $u^*\in U$. According to $g-u^*$ and $g-(u^*+v)$, it can be divided into four cases.

(1) When $g-u^*\ge 0$ and $g-(u^*+v)\ge 0$,

$$\mathcal{L}(u^*+v)=\mathcal{L}(u^*)+\frac12|v|_1^2+\beta\|Tv\|_{L^2(\partial\Omega)}^2+\alpha\int_\Omega v^2dx.$$

(2) When $g - u^* \geq 0$ and $g - (u^* + v) < 0$,

$$\begin{aligned}\mathcal{L}(u^* + v) = \mathcal{L}(u^*) + \frac{1}{2}|v|_1^2 + \beta\|Tv\|^2_{L^2(\partial\Omega)} \\ - \alpha \int_\Omega [g - (u^* + v)]^2 dx + \alpha \int_\Omega v^2 dx.\end{aligned}$$

Because of $-\alpha \int_\Omega [g - (u^* + v)]^2 dx + \alpha \int_\Omega v^2 dx > 0$, we obtain

$$\begin{aligned}\mathcal{L}(u^* + v) - \mathcal{L}(u^*) &< \alpha \int_\Omega v^2 dx + \frac{1}{2}|v|_1^2 + \beta\|Tv\|^2_{L^2(\partial\Omega)}, \\ \mathcal{L}(u^* + v) - \mathcal{L}(u^*) &> \frac{1}{2}|v|_1^2 + \beta\|Tv\|^2_{L^2(\partial\Omega)}.\end{aligned}$$

(3) When $g - u^* < 0$ and $g - (u^* + v) \geq 0$,

$$\mathcal{L}(u^* + v) = \mathcal{L}(u^*) + \frac{1}{2}|v|_1^2 + \beta\|Tv\|^2_{L^2(\partial\Omega)} + \alpha \int_\Omega [g - (u^* + v)]^2 dx.$$

Hence,

$$\begin{aligned}\mathcal{L}(u^* + v) - \mathcal{L}(u^*) &< \alpha \int_\Omega v^2 dx + \frac{1}{2}|v|_1^2 + \beta\|Tv\|^2_{L^2(\partial\Omega)}, \\ \mathcal{L}(u^* + v) - \mathcal{L}(u^*) &\geq \frac{1}{2}|v|_1^2 + \beta\|Tv\|^2_{L^2(\partial\Omega)}.\end{aligned}$$

(4) When $u^* - g < 0$ and $u^* + v - g < 0$,

$$\mathcal{L}(u^* + v) = \mathcal{L}(u^*) + \frac{1}{2}|v|_1^2 + \beta\|Tv\|^2_{L^2(\partial\Omega)}.$$

By the above results of four cases, we deduce that, for any $u \in \mathcal{P}$,

$$\mathcal{L}(u^* + v) - \mathcal{L}(u^*) < \alpha \int_\Omega v^2 dx + \frac{1}{2}|v|_1^2 + \beta\|Tv\|^2_{L^2(\partial\Omega)}.$$

From the Trace theorem and the fact $u = u^* + v$, we have

$$\mathcal{L}(u) - \mathcal{L}(u^*) < (\alpha + c_1\beta)\|u - u^*\|^2_{H_1},$$

where c_1 is a contant and $\alpha \gg 1$. On the other hand, we obtain, from Poincaré–Friedrichs-inequality,

$$\mathcal{L}(u) - \mathcal{L}(u^*) \geq \frac{c_2}{2}\|u - u^*\|^2_{H_1},$$

where c_2 is also a constant. In summary,

$$\mathcal{L}\left(u_{\theta_\mathcal{A}}\right) - \mathcal{L}\left(u^*\right) \geq \frac{c_2}{2}\|u_{\theta_\mathcal{A}} - u^*\|^2_{H_1},$$

and

$$\mathcal{L}(\bar{u}) - \mathcal{L}\left(u^*\right) < (\alpha + \beta c_1)\|\bar{u} - u^*\|^2_{H_1}.$$

From

$$\mathcal{L}\left(u_{\theta_A}\right) - \mathcal{L}\left(u^*\right) \leq \inf_{\bar{u}\in\mathcal{P}}\left[\mathcal{L}(\bar{u}) - \mathcal{L}\left(u^*\right)\right] + 2\sup_{u\in\mathcal{P}}|\mathcal{L}(u) - \widehat{\mathcal{L}}(u)| + \left[\widehat{\mathcal{L}}\left(u_{\theta_\mathcal{A}}\right) - \widehat{\mathcal{L}}\left(\widehat{u}_\theta\right)\right],$$

we can have the conclusion of the lemma. □

2.3. Approximation error

We denote the dyadic partition of $[0,1]$ by π_l,

$$\pi_l : t_0^{(l)} = 0 < t_1^{(l)} < \cdots < t_{2^l-1}^{(l)} < t_{2^l}^{(l)} = 1,$$

where $t_i^{(l)} = i \cdot 2^{-l} \left(0 \le i \le 2^l\right)$. The cardinal B-spline of order 3 over π_l can be written as

$$N_{l,i}^{(3)}(x) = 2^{2l-1} \sum_{j=0}^{3} (-1)^j \left(\begin{array}{c} 3 \\ j \end{array} \right) \left(x - i2^{-l} - j2^{-l}\right)_+^2, \tag{8}$$

for $i = -2, \ldots, 2^l - 1$. The multivariate case is defined by the product of several univariate cardinal B-splines, i.e.,

$$N_{l,\mathbf{i}}^{(3)}(\mathbf{x}) = \prod_{j=1}^{d} N_{l,i_j}^{(3)}(x_j), \quad \mathbf{i} = (i_1, \ldots, i_d), -3 < i_j < 2^l. \tag{9}$$

Here, we consider a special neural network (ReLU^3 network), which has the activation function $\sigma(x) =$ ReLU^3, i.e.,

$$\sigma(x) = \begin{cases} x^3, & x \ge 0, \\ 0, & \text{else}. \end{cases}$$

To present the upper bound of $\mathcal{E}_{app}$, we use the same approach as Y. Jiao et al. did in proving Theorem 4.2 of [23]. We omit the proof details of the following lemmas for space reasons (See [24] and Theorem 3.4 of [25]).

Lemma 2. *Assume $u^* \in H^2$, there exists $\{\alpha_j\}_{j=1}^{\left(2^l-4\right)^d}$ $(\alpha_j \in \mathbb{R})$ with $l > 2$ such that*

$$\left\| u^* - \sum_{j=1}^{\left(2^l-4\right)^d} \alpha_j N_{l,\mathbf{i}_j}^{(3)} \right\|_{H^1(\Omega)} \le \frac{C}{2^l} \|u^*\|_{H^1(\Omega)},$$

where C is a constant only depend on d.

Lemma 3. *The multivariate cardinal B-spline $N_{l,\mathbf{i}}^{(3)}(\mathbf{x})$ can be implemented exactly by a ReLU^3 network with the depth of $\lceil \log_2 d \rceil + 2$ and the width of $12d$.*

Theorem 1. *Assume $\|u^*\|_{H^1(\Omega)} \le c_3$. Then for any given $\epsilon > 0$, there exist an* ReLU^3 *network $u \in \mathcal{P}$ with*

$$\mathcal{D} \le \lceil \log_2 d \rceil + 3, \quad \mathcal{W} \le 12d \left[\frac{Cc_3}{\epsilon} - 4\right]^d,$$

such that

$$\|u^* - u\|_{H^1(\Omega)}^2 \le \epsilon,$$

where C is a constant depending only on the dimension d.

Proof. Combining Lemmas 2 and 3, we have, for $\epsilon > 0$ and $\frac{1}{2^l} \le \left\lceil \frac{\epsilon}{C\|u^*\|_{H^1}} \right\rceil$, there exists $u \in \mathcal{P}$ such that

$$\|u^* - u\|_{H^1(\Omega)} \le \epsilon.$$

The depth $\mathcal{D}$ and the width $\mathcal{W}$ of u satisfy $\mathcal{D} \le \lceil \log_2 d \rceil + 3$ and $\mathcal{W} \le 12d \left[\frac{C\|u^*\|_{H^1}}{\epsilon} - 4\right]^d$, respectively. □

2.4. Statistical error

In this section, we introduce the Rademacher complexity to bound $\mathcal{E}_{sta}$. Let $\{X_k\}_{k=1}^{N}$ and $\{Y_k\}_{k=1}^{M}$ be i.i.d. random variables drawn from the distributions $U(\Omega)$ and $U(\partial\Omega)$, respectively. By applying triangle inequality, we have

$$\mathbb{E}_{\{X_k,Y_k\}_{k=1}^{M}} \sup_{u\in\mathcal{P}} |\mathcal{L}(u)-\widehat{\mathcal{L}}(u)| \le \sum_{j=1}^{8} \mathbb{E}_{\{X_k,Y_k\}_{k=1}^{M}} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_j(u)-\widehat{\mathcal{L}}_j(u)\right|,$$

where

$$\mathcal{L}_1(u) = 1/2|\Omega|\mathbb{E}_{X\sim \mathrm{U}(\Omega)}\left[\|\nabla u\|_2^2(X)\right],$$

$$\mathcal{L}_2(u) = |\Omega|\mathbb{E}_{X\sim \mathrm{U}(\Omega)}[u(X)f(X)],$$

$$\mathcal{L}_3(u) = \beta|\partial\Omega|\mathbb{E}_{Y\sim U(\partial\Omega)}u(Y)^2$$

$$\mathcal{L}_4(u) = \beta|\partial\Omega|\mathbb{E}_{Y\sim U(\partial\Omega)}(h(Y))^2$$

$$\mathcal{L}_5(u) = -2\beta|\partial\Omega|\mathbb{E}_{Y\sim U(\partial\Omega)}u(Y)h(Y)$$

$$\mathcal{L}_6(u) = \alpha|\Gamma|\mathbb{E}_{X\sim \mathrm{U}(\Gamma)}[u(X)^2],$$

$$\mathcal{L}_7(u) = -2\alpha|\Gamma|\mathbb{E}_{X\sim \mathrm{U}(\Gamma)}[u(X)g(X)],$$

$$\mathcal{L}_8(u) = \alpha|\Gamma|\mathbb{E}_{X\sim \mathrm{U}(\Gamma)}[g(X)^2],$$

where $\Gamma \subset \Omega$ with $u > g$ and $\widehat{\mathcal{L}}_j(u)$ is the discretization corresponding to $\mathcal{L}_j(u)$.

Definition 1. The Rademacher complexity of a set $A \subseteq \mathbb{R}^N$ is defined as

$$\Re(A) = \mathbb{E}_{\{\xi_i\}_{k=1}^{N}}\left[\sup_{a_1,\dots,a_N\in A} \frac{1}{N}\sum_{k=1}^{N}\xi_k a_k\right],$$

where $\{\xi_k\}_{k=1}^{N}$ are N i.i.d Rademacher variables with $\mathbb{P}(\xi_k=1) = \mathbb{P}(\xi_k=-1) = \frac{1}{2}$. The Rademacher complexity of function class $\mathcal{F}$ associated with random samples $\{X_k\}_{k=1}^{N}$ is defined as

$$\Re(\mathcal{F}) = \mathbb{E}_{\{X_k,\xi_k\}_{k=1}^{N}}\left[\sup_{u\in\mathcal{F}} \frac{1}{N}\sum_{k=1}^{N}\xi_k u(X_k)\right].$$

By introducing Rademacher complexity, we get the following lemma and then derive the bound of $\mathcal{E}_{sta}$ in Theorem 2.

Lemma 4. *Assume that*

$$\max\left(\|f\|_{L^\infty(\Omega)}, \|h\|_{L^\infty(\partial\Omega)}, \|g\|_{L^\infty(\Omega)}, \|u\|_{L^\infty(\Omega)}, \|\nabla u\|^2_{L^\infty(\Omega)}\right) \le \mathcal{B} < \infty,$$

where $u \in \mathcal{P}$ and $\mathcal{B}$ is a constant. Then

$$\mathbb{E}_{\{X_k\}_{k=1}^{N}} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_1(u)-\widehat{\mathcal{L}}_1(u)\right| \le 2\mathcal{B}^2|\Omega|\Re(\mathcal{F}_1),$$

$$\mathbb{E}_{\{X_k\}_{k=1}^{N}} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_2(u)-\widehat{\mathcal{L}}_2(u)\right| \le 2\mathcal{B}|\Omega|\Re(\mathcal{F}_2),$$

$$\mathbb{E}_{\{X_k\}_{k=1}^N} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_3(u) - \widehat{\mathcal{L}}_3(u)\right| \le 2\beta\mathcal{B}^2|\partial\Omega|\Re(\mathcal{F}_3),$$
$$\mathbb{E}_{\{X_k\}_{k=1}^N} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_4(u) - \widehat{\mathcal{L}}_4(u)\right| \le 2\beta\mathcal{B}^2|\partial\Omega|\Re(\mathcal{F}_4),$$
$$\mathbb{E}_{\{X_k\}_{k=1}^N} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_5(u) - \widehat{\mathcal{L}}_5(u)\right| \le 4\beta\mathcal{B}|\partial\Omega|\Re(\mathcal{F}_5),$$
$$\mathbb{E}_{\{X_k\}_{k=1}^N} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_6(u) - \widehat{\mathcal{L}}_6(u)\right| \le 2\alpha\mathcal{B}^2|\Gamma|\Re(\mathcal{F}_6),$$
$$\mathbb{E}_{\{X_k\}_{k=1}^N} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_7(u) - \widehat{\mathcal{L}}_7(u)\right| \le 4\alpha\mathcal{B}|\Gamma|\Re(\mathcal{F}_7),$$
$$\mathbb{E}_{\{X_k\}_{k=1}^N} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_8(u) - \widehat{\mathcal{L}}_8(u)\right| \le 2\alpha\mathcal{B}^2|\Gamma|\Re(\mathcal{F}_8),$$

where

$$\mathcal{F}_1 = \left\{f:\Omega\to\mathbb{R}, \mid \exists u\in\mathcal{P}\ f(x) = \|\nabla u(x)\|^2\right\}, \quad \mathcal{F}_2 = \mathcal{P},$$
$$\mathcal{F}_3 = \left\{f:\partial\Omega\to\mathbb{R}\,|\exists u\in\mathcal{P}|_{\partial\Omega}\,,\ s.t.\ f(x)=u(x)^2\right\}, \quad \mathcal{F}_4 = \{f:\partial\Omega\to\mathbb{R}\mid -1,0,1\},$$
$$\mathcal{F}_5 = \mathcal{P}|_{\partial\Omega}\,, \quad \mathcal{F}_6(u) = \left\{f:\Omega\to\mathbb{R}, \exists u\in\mathcal{P},\ s.t.\ f(x)=u(x)^2\right\},$$
$$\mathcal{F}_7(u) = \left\{f:\Omega\to\mathbb{R}, \exists u\in\mathcal{P},\ s.t.\ f(x)=u(x)\right\}, \quad \mathcal{F}_8 = \{f:\partial\Omega\to\mathbb{R}\mid -1,0,1\}.$$

Proof. We prove only one of these inequalities. The other inequalities can be proved in a similar way.

$$\begin{aligned}
&\mathbb{E}_{\{X_k\}_{k=1}^N} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_2(u) - \widehat{\mathcal{L}}_2(u)\right| \\
&= |\Omega|\mathbb{E}_{\{X_k\}_{k=1}^N} \sup_{u\in\mathcal{P}} \left|\mathbb{E}_{X\sim U(\Omega)}\left(u(X)f(X)\right) - \frac{1}{N}\sum_{k=1}^N (u(X_k)f(X_k))\right| \\
&= |\Omega|\mathbb{E}_{\{X_k\}_{k=1}^N} \sup_{u\in\mathcal{P}} \left|\mathbb{E}_{\{\hat{X}_k\}_{k=1}^N}\frac{1}{N}\sum_{k=1}^N\left(u(\hat{X}_k)f(\hat{X}_k)\right) - \frac{1}{N}\sum_{k=1}^N (u(X_k)f(X_k))\right| \\
&\le \frac{|\Omega|}{N}\mathbb{E}_{\{X_k,\hat{X}_k\}_{k=1}^N}\left[\sup_{u\in\mathcal{P}}\left|\sum_{k=1}^N\left(u(\hat{X}_k)f(\hat{X}_k) - u(X_k)f(X_k)\right)\right|\right] &(10)\\
&= \frac{|\Omega|}{N}\mathbb{E}_{\{X_k,\hat{X}_k,\xi_k\}_{k=1}^N}\left[\sup_{u\in\mathcal{P}}\left|\sum_{k=1}^N\xi_k\left(u(\hat{X}_k)f(\hat{X}_k) - u(X_k)f(X_k)\right)\right|\right] &(11)\\
&\le \frac{|\Omega|}{N}\mathbb{E}_{\{\hat{X}_k,\xi_k\}_{k=1}^N}\left[\sup_{u\in\mathcal{P}}\left|\sum_{k=1}^N\xi_k\left(u(\hat{X}_k)f(\hat{X}_k)\right)\right|\right] \\
&\qquad + \frac{|\Omega|}{N}\mathbb{E}_{\{X_k,\xi_k\}_{k=1}^N}\left[\sup_{u\in\mathcal{P}}\left|\sum_{i=1}^N\xi_k\left(u(X_k)f(X_k)\right)\right|\right] \\
&\le 2\mathcal{B}|\Omega|\Re\left(\mathcal{F}_2\right). &(12)
\end{aligned}$$

In the previous proof, we used Jensen's inequality to obtain (10). The Eq. (11) can be deduced from the definition of ξ_k in Definition 1. And (12) holds because the distributions of the two terms are the same. □

Theorem 2. *Let N and M be the number of samples in Ω and $\partial\Omega$, respectively. And both the depth $\mathcal{D}$ and width $\mathcal{W}$ of the network are positive integers. For any $\epsilon > 0$, if*

$$N, M = C\mathcal{D}^6\mathcal{W}^2(\mathcal{D}+\log\mathcal{W})(\log\mathcal{D}+\log\mathcal{W})\left(\frac{1}{\epsilon}\right)^2\log\frac{1}{\epsilon},$$

then we have

$$\mathbb{E}_{\{X_k\}_{k=1}^N,\{Y_k\}_{k=1}^M} \sup_{u\in\mathcal{P}} |\mathcal{L}(u) - \widehat{\mathcal{L}}(u)| \le \epsilon$$

where $\mathcal{P} = \mathcal{N}\left(\mathcal{D}, \mathcal{W}, \left\{\mathrm{ReLU}^3\right\}\right)$, *and* C *is a constant.*

Proof. From (C.7) in [23], we have

$$\mathfrak{R}\left(\mathcal{F}_i\right) \le 28\sqrt{\frac{3}{2}} \max\left\{\mathcal{B}, \mathcal{B}^2\right\} \left(\frac{\mathcal{H}}{N}\right)^{1/2} \sqrt{\log\left(\frac{eN}{\mathcal{H}}\right)}, \quad i = 1,2,6,7,8,$$

and

$$\mathfrak{R}\left(\mathcal{F}_i\right) \le 28\sqrt{\frac{3}{2}} \max\left\{\mathcal{B}, \mathcal{B}^2\right\} \left(\frac{\mathcal{H}}{M}\right)^{1/2} \sqrt{\log\left(\frac{eM}{\mathcal{H}}\right)}, \quad i = 3,4,5,$$

where $\mathcal{H} = c_4(\mathcal{D}+3)^4\mathcal{W}^2(\mathcal{D}+3+log((\mathcal{D}+3\mathcal{W})))$ (c_4 is a constant). Then,

$$\begin{aligned}&\mathbb{E}_{\{X_k\}_{k=1}^N,\{Y_k\}_{k=1}^M} \sup_{u\in\mathcal{P}} |\mathcal{L}(u) - \widehat{\mathcal{L}}(u)| \\ \le &\sum_{j=1}^{8} \mathbb{E}_{\{X_k\}_{k=1}^N,\{Y_k\}_{k=1}^M} \sup_{u\in\mathcal{P}} \left|\mathcal{L}_j(u) - \widehat{\mathcal{L}}_j(u)\right| \\ \le &28\sqrt{\frac{3}{2}} c_5 \max\left\{\mathcal{B}, \mathcal{B}^2\right\} \left((4+8\alpha)|\Omega| \left(\frac{\mathcal{H}}{N}\right)^{1/2} \sqrt{\log\left(\frac{eN}{\mathcal{H}}\right)}\right. \\ &\left. + (8\beta)|\partial\Omega| \left(\frac{\mathcal{H}}{M}\right)^{1/2} \sqrt{\log\left(\frac{eM}{\mathcal{H}}\right)}\right)\end{aligned}$$

where c_5 is a constant. Therefore, the conclusion of the theorem holds. □

2.5. Main theorem

We present the main result of this paper.

Theorem 3. *Assume* $\mathcal{E}_{opt} = 0$ *and* $\|u^*\|_{H^1(\Omega)} \le c_3$. *For any* $\epsilon > 0$, *if*

$$\mathcal{D} \le \lceil \log_2 d \rceil + 3, \quad \mathcal{W} \le 12d\left[\frac{Cc_3}{\epsilon} - 4\right]^d,$$

and u_θ *is the minimizer of* (7) *with numbers of samples*

$$M, N = C\left(\frac{1}{\epsilon}\right)^{2d+2} \log\frac{1}{\epsilon},$$

then we have

$$\mathbb{E}_{\{X_k\}_{k=1}^N,\{Y_k\}_{k=1}^M} \|u_\theta - u^*\|_{H^1(\Omega)} \le \epsilon.$$

Proof. For any $\epsilon > 0$, by Theorem 1, there exists a neural network function $\bar{u}$ with depth $\lceil \log_2 d \rceil + 3$ and width $12d$ such that

$$\mathcal{E}_{app} = \|u^* - \bar{u}\|_{H^1} \le \epsilon.$$

From Theorem 2, when the number of samples

$$M, N = C_2\mathcal{D}^4\mathcal{W}^2(\mathcal{D} + \log\mathcal{W})(\log\mathcal{D} + \log\mathcal{W})\left(\frac{1}{\epsilon}\right)^2 \log\frac{1}{\epsilon} = C_3\left(\frac{1}{\epsilon}\right)^{2d+2} \log\frac{1}{\epsilon},$$

we have

$$\mathcal{E}_{sta} = \mathbb{E}_{\{X_k\}_{k=1}^N, \{Y_k\}_{k=1}^M} \sup_{u \in \mathcal{P}} |\mathcal{L}(u) - \widehat{\mathcal{L}}(u)| \leq \epsilon.$$

Applying the results of Lemma 1, we complete the proof. □

Remark. Although we have established the non-asymptotic convergence rate of the deep learning method, we do not discuss $\mathcal{E}_{\text{opt}}$. In practical calculations, the stochastic gradient descent algorithm (SGD) with mini-batch is adopted, which gives the deep learning method the advantage of being meshless and unsupervised. The optimization algorithm is shown in Algorithm 1.

Algorithm 1 Optimization algorithm

```
Require: The network depth $\mathcal{D}$ and width $\mathcal{W}$, the penalty parameters $\alpha$ and $\beta$, the number of samples $N$
  and $M$, the initial guess of the parameter $\theta_1$, the learning rate $\eta$, and the total number of iterations K.
  for k=1,2,...K do
    Randomly sample a batch $\{X_i\}_{i=1}^N \sim \mathrm{U}(\Omega)$, $\{Y_j\}_{i=1}^M \sim \mathrm{U}(\partial\Omega)$.
    Compute $\text{loss}_1 = \frac{1}{N}\sum_{i=1}^N \left[\frac{1}{2}\|\nabla u_\theta(X_i)\|_2^2 - f(X_i)\,u_\theta(X_i)\right]$,
              $\text{loss}_2 = \frac{1}{N}\sum_{i=1}^N [g(X_i) - u_\theta(X_i)]_+^2$,
              $\text{loss}_3 = \frac{1}{M}\sum_{j=1}^M [u_\theta(Y_j) - h(Y_j)]^2$.
    Compute $\mathcal{L}(\theta) = \text{loss}_1 + \alpha\,\text{loss}_2 + \beta\,\text{loss}_3$.
    Update $\theta_{k+1} = \theta_k - \eta\nabla_\theta\mathcal{L}(\theta_k)$.
  end for
```

3. Numerical examples

In this section, three examples (including a 2D bilateral obstacle problem) are considered to examine the robustness and effectiveness of our proposed method. We will compare the solutions of deep neural networks with reference (analytical or numerical) solutions. In each example, a neural network with 8 (or more) hidden layers and 80 neurons is employed with the activation function of $ReLU^3$. We incorporate layer normalization into the network and use the Adam optimizer version of the SGD with a learning rate of 5×10^{-4} to optimize the loss function. Furthermore, all networks are generated and trained with the Pytorch library [26]. The whole source code is available at https://github.com/Xingbaji/Obstacle_problem.

Example 1. Consider a 1D obstacle problem

$$\begin{cases} -u'' \geq 0 & \text{in } \Omega, \\ u \geq g & \text{in } \Omega, \\ u''(u-g) = 0 & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{13}$$

where $\Omega = [0,1]$ and

$$g(x) = \begin{cases} 100x^2 & \text{for} \quad 0 \leq x \leq 0.25, \\ 100x(1-x) - 12.5 & \text{for} \quad 0.25 \leq x \leq 0.5, \\ g(1-x) & \text{for} \quad 0.5 \leq x \leq 1.0. \end{cases}$$

The exact solution is

$$u_{\text{exact}}(x) = \begin{cases} (100 - 50\sqrt{2})x & \text{for} \quad 0 \leq x \leq \frac{1}{2\sqrt{2}}, \\ 100x(1-x) - 12.5 & \text{for} \quad \frac{1}{2\sqrt{2}} \leq x \leq 0.5, \\ u_{\text{exact}}(1-x) & \text{for} \quad 0.5 \leq x \leq 1.0. \end{cases}$$

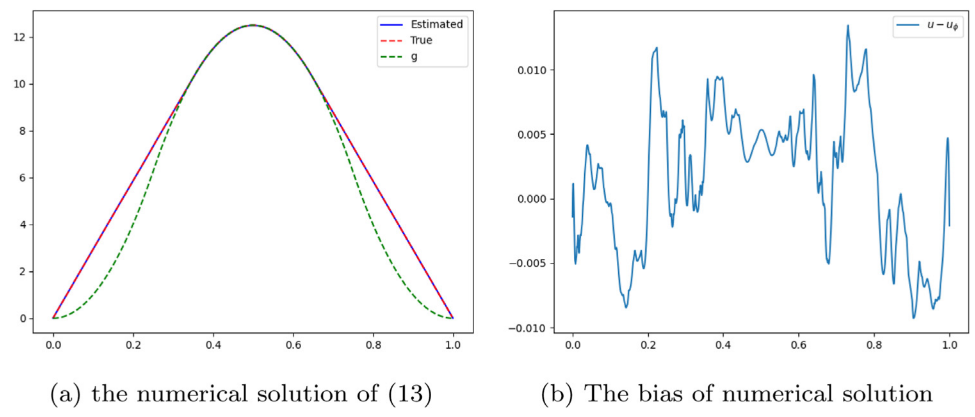


Fig. 2. The numerical results of Example 1.

Table 1
The relative error with respect to different sampling number. The sampled points are fixed on the uniform mesh.

Sampling number	Relative error
20	0.47
50	0.35
100	0.14
200	0.067
500	0.035
1000	0.019
10000	0.0042

The loss function is defined as

$$\hat{\mathcal{L}}(u_\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{2}\|\nabla u_\theta(X_i)\|_2^2 + \alpha[g(X_i) - u_\theta(X_i)]_+^2\right] + \frac{\beta}{2}\left[u_\theta(0)^2 + u_\theta(1)^2\right],$$

where $\{X_k\}_{k=1}^N$ are randomly sampled from $\mathcal{U}(0,1)$.

In this example, the training data set is divided into mini-batches of size 2000. We typically train the neural network with 3000 iterations and set the penalty parameters $\alpha = \beta = 5000$. We evaluate the performance of our method on a uniform mesh with a grid size of 10^{-3}.

Fig. 2(a) shows the numerical result, the exact solution, and the obstacle function. Fig. 2(b) shows the difference between the numerical solution and the exact solution. In Fig. 3, we plot the evolutions of $loss_1$, $loss_2$, and $loss_3$, which are defined in Algorithm 1. The L_1 error of solution during the training process with *log* scale is also reported. All results show that our method converges within 1000 iterations.

In Theorem 3, we give the estimate of statistical error with respect to the number of samples, which needs 10^4 samples to ensure that the statistical error is less than 10^{-1}. In practice, the numerical results perform better than the theoretical analysis. Table 1 shows the relationship between relative error and the number of samples. To ensure the approximation error and the optimization error of these experiments are close, we use the same neural network architecture with 8 hidden layers and 80 neurons and take 5000 iterations for all the models. In Fig. 4, we observe that the statistical error is inversely proportional to the number of samples on the $log-log$ scale. This property is basically consistent with the theoretical analysis. In addition, our method is unsupervised, which means we can always generate a sufficient number of samples.

Theorem 1 gives the upper bound of the approximation error with respect to the depth and width of the neural networks. In this experiment, given $d = 1$ and the exact solution u^*, we obtain that the approximation error is less than 0.01 when $\mathcal{D} \le 3, \mathcal{W} \le 395$. We use 10^4 samples and take 5000 iterations to ensure sufficient

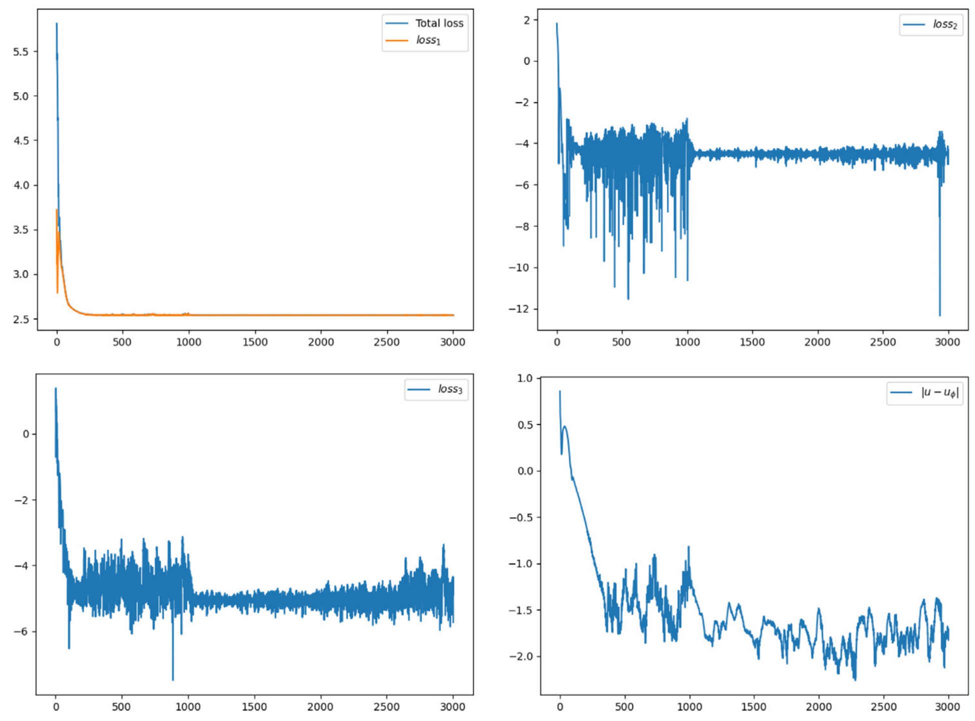


Fig. 3. The evolutions of $loss_1$, $loss_2$, and $loss_3$. The L_1 error during the training process with the *log* scale is plotted in the right-bottom subfigure.

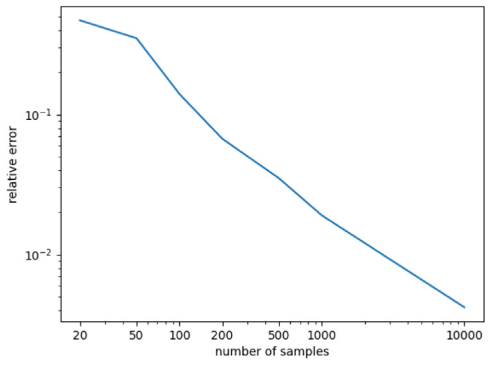


Fig. 4. Relative error vs. number of samples in the *log*-*log* scale.

small optimization errors and statistical errors. Table 2 shows the effect of the neural network architecture (number of layers and neurons). The numerical result with 3 × 320 validates our theoretical analysis. In practice, the neural network with too many neurons in each layer is usually slower, so we use the neural network with 8 layers and 80 neurons in each layer as the default model. Furthermore, we can see that our method is robust with respect to the neural network architecture.

In Table 3, we investigate the robustness of the penalty parameters of our method. We can observe that our method obtains accurate solutions as long as $\alpha, \beta \geq 1000$, which means our method does not require fine-tuning of penalty parameters, but only needs to select a sufficiently large number.

Table 2
The relative error with different neural network architectures.

Layers & neurons	Relative error
3 × 40	0.022
3 × 80	0.028
3 × 160	0.010
3 × 320	0.0096
8 × 40	0.009
8 × 80	0.005

Table 3
The relative error with different penalty parameters.

α	β	Relative error
100	100	0.10
500	500	0.018
1000	1000	0.0074
1000	5000	0.0087
5000	1000	0.0081
5000	5000	0.0051
10000	1000	0.0069
1000	10000	0.0091
10000	10000	0.0065

Example 2. Consider a 2D obstacle problem

$$\begin{cases} -\Delta u \geq f & \text{in } \Omega, \\ u \geq g & \text{in } \Omega, \\ (-\Delta u - f)(u-g) = 0 & \text{in } \Omega, \\ u = h & \text{on } \partial\Omega, \end{cases} \tag{14}$$

where $\Omega := [-2,2]^2$, $f = 0$ and the obstacle function

$$g(x,y) = \begin{cases} \sqrt{1-r^2}, & r = \sqrt{x^2+y^2} \leq 1, \\ -1, & \text{elsewhere} . \end{cases}$$

The Dirichlet boundary condition is determined from the exact solution

$$u^*(x,y) = \begin{cases} \sqrt{1-r^2}, & r \leq r^*, \\ -\left(r^*\right)^2 \ln(r/2)/\sqrt{1-\left(r^*\right)^2}, & r \geq r^*, \end{cases}$$

where $r^* \approx 0.6979651482$ satisfies $\left(r^*\right)^2\left(1 - \ln\left(r^*/2\right)\right) = 1$. And the loss function is

$$\hat{\mathcal{L}}(u_\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{2}\|\nabla u_\theta(X_i)\|_2^2 - f(X_i)u_\theta(X_i) + \alpha[g(X_i) - u_\theta(X_i)]_+^2\right] + \frac{\beta}{M}\sum_{j=1}^{M}[u_\theta(Y_j) - h(Y_j)]^2.$$

In this example, we set $\alpha = \beta = 5000$, and train the neural network with 5000 iterations. The batch size of interior and boundary points are 40000 and 800, In Fig. 5, we present the numerical solution and its difference from the exact solution. We visualize $loss_1, loss_2$, and $loss_3$ in Fig. 6. It can be noted that the proposed method performs well on the boundary and the contact parts. In Fig. 6(a), the $loss_1$ is close to 0 except for the contact part, which is consistent with the definition of $loss_1$.

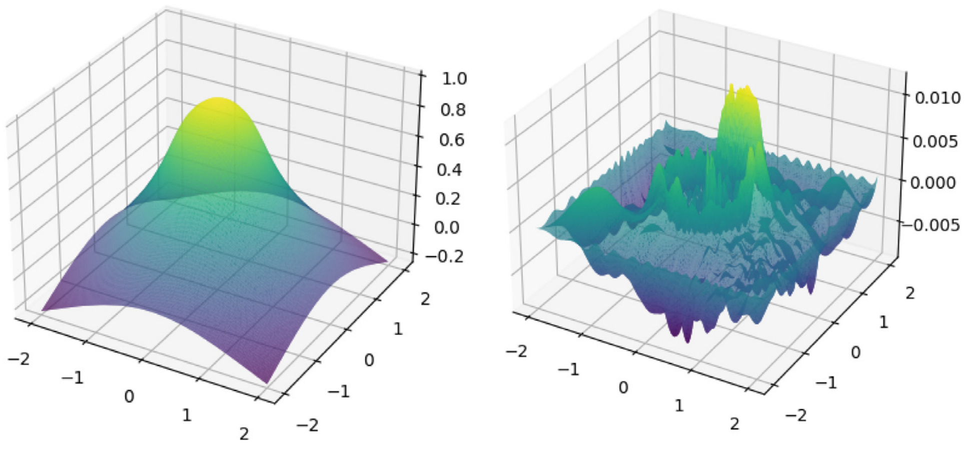


Fig. 5. Left: The profile of the neural network solution. Right: The difference between the neural network solution and the exact solution.

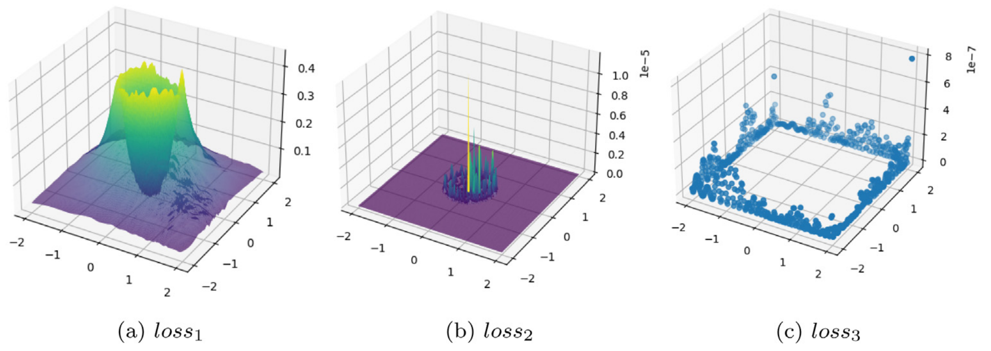


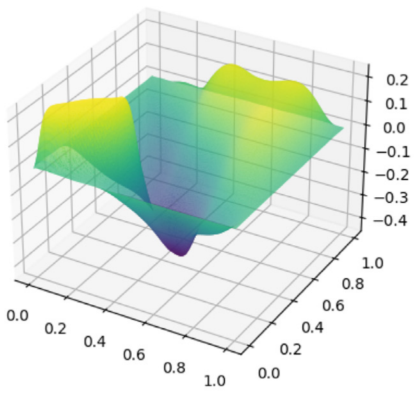
Fig. 6. The profiles of $loss_1$(Left), $loss_2$(Middle) and $loss_3$(Right).

Example 3. Our method can be extended to the bilateral obstacle problem. We consider a two-dimensional case that has been reported in [27,28], i.e., $\Omega = (0,1) \times (0,1)$. Let the lower obstacle $g_1(x,y) = -\operatorname{dist}((x,y), \partial\Omega)$, the upper obstacle $g_2(x,y) = 0.2$, $h = 0$, and

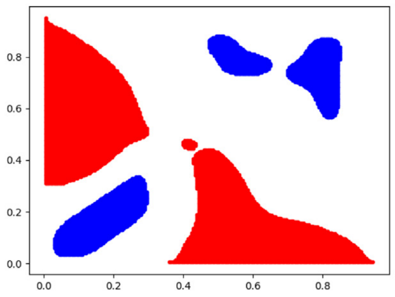
$$f(x,y) = \begin{cases} 300, & \text{if } (x,y) \in S, \\ -70\exp(y)k(x), & \text{if } x \le 1-y \text{ and } (x,y) \notin S, \\ 15\exp(y)k(x), & \text{if } x > 1-y \text{ and } (x,y) \notin S, \end{cases}$$

where $S = \{(x,y) \in \Omega : |x-y| \le 0.1 \text{ and } x \le 0.3\}$,

$$k(x) = \begin{cases} 6x, & \text{if } 0 < x \le 1/6, \\ 2(1-3x), & \text{if } 1/6 < x \le 1/3, \\ 6(x-1/3), & \text{if } 1/3 < x \le 1/2, \\ 2(1-3(x-1/3)), & \text{if } 1/2 < x \le 2/3, \\ 6(x-2/3), & \text{if } 2/3 < x \le 5/6, \\ 2(1-3(x-2/3)), & \text{if } 5/6 < x \le 1. \end{cases}$$



(a) The numerical solution of example 3.



(b) The lower(red) and upper(blue) coincidence sets.

Fig. 7. The numerical results and the coincidence sets of Example 3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The corresponding loss function is

$$\hat{\mathcal{L}}(u) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{2}\|\nabla u(X_i)\|_2^2 - f(X_i)u(X_i) + \alpha_1[g_1(X_i) - u(X_i)]_+^2\right.$$
$$\left. + \alpha_2[u(X_i) - g_2(X_i)]_+^2\right] + \frac{\beta}{M}\sum_{j=1}^{M}(u(Y_j) - h(Y_j))^2.$$

We take $\alpha_1 = \alpha_2 = \beta = 50000$ and train the neural network for 5000 iterations. The batch sizes for interior and boundary points are 40000 and 800, respectively. Fig. 7(a) shows the approximated solution. There is no analytical solution for this example, so we display the lower(red) and upper(blue) coincidence sets in Fig. 7(b), which closely resembles Figure 3 of Example 5.4 in [27].

4. Conclusion

In this work, we propose a method based on deep learning to solve the obstacle problem. By introducing penalty terms, the original obstacle problem is transformed into a minimizing optimization problem. It can solve the optimization problem in the framework of neural networks naturally and get an approximate solution to the obstacle problem. The theoretical analysis of the unilateral obstacle problem is given. The non-asymptotic convergence rate is constructed by estimating the upper bounds of the depth and width of the network and the number of training samples required to achieve the expected accuracy. Numerical examples show that the proposed method is robust with respect to parameters and network structure and has the advantages of being easy to implement, meshless, and unsupervised. Especially without tedious fine-tuning, the proposed method can be applied to bilateral obstacle problems, such as Example 3. This shows that the proposed method can be applied to more complex obstacle problems. In future work, we will explore the application of deep learning methods to other variational inequalities.

Acknowledgments

The authors thank the anonymous reviewers for valuable comments and suggestions. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under the Marie Sklodowska-Curie Grand Agreement No. 823731 CONMECH.

References

[1] E. Burman, P. Hansbo, M.G. Larson, R. Stenberg, Galerkin least squares finite element method for the obstacle problem, Comput. Methods Appl. Mech. Engrg. 313 (2017) 362–374.
[2] R. Kornhuber, Monotone multigrid methods for elliptic variational inequalities II, Numer. Math. 72 (1996) 481–499.
[3] D.M. Yuan, X.L. Cheng, An iterative algorithm based on the piecewise linear system for solving bilateral obstacle problems, Int. J. Comput. Math. 89 (16–18) (2012) 2374–2384.
[4] T. Fuhrer, First-order least-squares method for the obstacle problem, Numer. Math. (4) (2020) 55–88.
[5] K. Majava, X.-C. Tai, A level set method for solving free boundary problems associated with obstacles, Int. J. Numer. Anal. Model. 1 (2) (2004) 157–171.
[6] Q. Ran, X. Cheng, S. Abide, A dynamical method for solving the obstacle problem, Numer. Math. Theory Methods Appl. 2 (13) (2020) 353–371.
[7] S.H. Rudy, J.N. Kutz, S.L. Brunton, Deep learning of dynamics and signal-noise decomposition with time-stepping constraints, J. Comput. Phys. 396 (2019) 483–506.
[8] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[9] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[10] Y. Sun, L. Zhang, H. Schaeffer, Neupde: Neural network based ordinary and partial differential equations for modeling time-dependent data, 2019, arXiv preprint arXiv:1908.03190.
[11] A.B. Farimani, J. Gomes, V.S. Pande, Deep learning the physics of transport phenomena, 2017, arXiv preprint arXiv:1709.02432.
[12] G.-J. Both, S. Choudhury, P. Sens, R. Kusters, DeepMoD: Deep learning for model discovery in noisy data, 2019, arXiv preprint arXiv:1904.09406.
[13] Y. Khoo, J. Lu, L. Ying, Solving parametric PDE problems with artificial neural networks, 2017, arXiv preprint arXiv:1707.03351.
[14] Y. Khoo, L. Ying, SwitchNet: a neural network model for forward and inverse scattering problems, SIAM J. Sci. Comput. 41 (5) (2019) A3182–A3201.
[15] J. Han, A. Jentzen, E. Weinan, Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning, 2017, pp. 1–13, arXiv preprint arXiv:1707.02568.
[16] M. Hutzenthaler, A. Jentzen, T. Kruse, T.A. Nguyen, P. von Wurstemberger, Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations, 2018, arXiv preprint arXiv:1807.01212.
[17] R.K. Tripathy, I. Bilionis, Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification, J. Comput. Phys. 375 (2018) 565–588.
[18] N. Winovich, K. Ramani, G. Lin, ConvPDE-UQ: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains, J. Comput. Phys. 394 (2019) 263–279.
[19] Z. Wu, A Deep Neural Network Method for Solving a Class of Contact Problems, Zhejiang Unviersity, 2021.
[20] J. Huang, C. Wang, H. Wang, Adaptive learning on the grids for elliptic hemivariational inequalities, 2021, arXiv preprint arXiv:2104.04881.
[21] R. Scholz, Numerical solution of the obstacle problem by the penalty method, Computing 32 (4) (1984) 297–306.
[22] C. Duan, Y. Jiao, Y. Lai, X. Lu, Z. Yang, Convergence rate analysis for deep Ritz method, 2021, arXiv preprint arXiv:2103.13330.
[23] Y. Jiao, Y. Lai, D. Li, X. Lu, Y. Wang, J.Z. Yang, Convergence analysis for the PINNs, 2021, arXiv preprint arXiv:2109.01780.
[24] X. Shen, Deep Learning-based Numerical Methods for Solving Differential Equations (Ph.D. thesis), Zhejiang University, 2022.
[25] M.H. Schultz, Approximation theory of multivariate spline functions in Sobolev spaces, SIAM J. Numer. Anal. 6 (4) (1969) 570–582.
[26] A. Paszke, S. Gross, F. Massa, et al., PyTorch: An imperative style, high-performance deep learning library, 2019.
[27] T. Kärkkäinen, K. Kunisch, P. Tarvainen, Augmented Lagrangian active set methods for obstacle problems, J. Optim. Theory Appl. 119 (3) (2003) 499–533.
[28] F. Wang, X.-L. Cheng, An algorithm for solving the double obstacle problems, Appl. Math. Comput. 201 (1–2) (2008) 221–228.