Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2020-0219

Vol. **29**, No. 5, pp. 1365-1384
May 2021

# Deep Nitsche Method: Deep Ritz Method with Essential Boundary Conditions

Yulei Liao[1,2] and Pingbing Ming[1,2,*]

[1] *LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, Chinese Academy of Sciences, No. 55, East Road Zhong-Guan-Cun, Beijing 100190, China.*
[2] *School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China.*

Received 5 November 2020; Accepted (in revised version) 13 January 2021

**Abstract.** We propose a new method to deal with the essential boundary conditions encountered in the deep learning-based numerical solvers for partial differential equations. The trial functions representing by deep neural networks are non-interpolatory, which makes the enforcement of the essential boundary conditions a nontrivial matter. Our method resorts to Nitsche's variational formulation to deal with this difficulty, which is consistent, and does not require significant extra computational costs. We prove the error estimate in the energy norm and illustrate the method on several representative problems posed in at most 100 dimension.

**AMS subject classifications**: 65N30, 65M12, 41A46, 35J25

**Key words**: Deep Nitsche Method, Deep Ritz Method, neural network approximation, mixed boundary conditions, curse of dimensionality.

## 1 Introduction

Recently there has been a surge of interests in solving partial differential equations by deep learning-based numerical methods [6,11,12,16,17,19,25,29,32,33,40,41,44–46], and we refer to [15] for a review for this direction. These methods allow for the compositional construction of new approximation sets from various neural networks. Such constructions are usually free of a mesh so that they are in essence *meshless methods* [5]. The trial functions in the approximation sets are in general non-interpolatory, which makes the implementation of the essential boundary conditions not an easy task. There are two

*Corresponding author. *Email addresses:* liaoyulei@lsec.cc.ac.cn (Y. L. Liao), mpb@lsec.cc.ac.cn (P. B. Ming)

 

main approaches to handle the essential boundary conditions in deep learning-based numerical methods. One is the conforming method, which exploits a supplementary neural network to make the functions in the trial set satisfy the boundary conditions *exactly*. This is the approach firstly proposed in [30,31] and recently further developed in [6,29]. The conforming method usually involves an accurate evaluation of the distance function or a cut-off function, which is not easy for domain with complicated boundary geometry; see, e.g., [6]. Another one is the penalty method, which is a very general concept and belongs to the so-called nonconforming method [17, 40, 41, 45, 46]. An additional surface term is introduced into the variational formulation to enforce the boundary conditions. However, great care has to be taken to balance the different terms in the functional framework. Otherwise, this may cause problems for the existence and uniqueness of the solution [2, 8]. Moreover, the penalty method usually leads to a sub-optimal rate of convergence as shown in [3] for finite element methods and as shown in [5] for the generalized finite element methods and meshless methods.

Compared to the penalty method, the Lagrange multiplier method treats the essential boundary conditions as a constraint in the minimization. This technique has been used to deal with the essential boundary conditions in finite element method [4] and wavelet method [13]. The optimal rate of convergence may be achieved if the approximation function spaces are chosen properly, which relies on the so-called inf-sup condition [4,13]. The Lagrange multiplier method may also be used to enforce boundary conditions in the neural-network based method provided that the resulting constrained minimization problem can be efficiently solved.

An efficient method for imposing the essential boundary conditions has been proposed by Nitsche in the early 1970's [38] in the finite element method. It was quite unknown for many years, and was revived in [42] by STENBERG. He revealed the interesting relation between Nitsche's method and certain stabilized Lagrangian multiplier methods. More recent efforts on Nitsche's method have been devoted to deal with the elliptic interface problems and the unfitted mesh problems; we refer to [10] for a review of the progress in this direction. In the context of the meshless method, Nitsche's idea has been proved to be an efficient approach to deal with the essential boundary conditions in the framework of a particle partition of unity method [23] as well as the generalized finite element method [35].

In this work, we incorporate the idea of Nitsche into the framework of Deep Ritz Method [17] to deal with the essential boundary conditions. This new algorithm is called Deep Nitsche Method. It also imposes the boundary conditions in a nonconforming way as the penalty method. In contrast to the penalty method, this method is consistent if the exact solution is smooth enough. The method is based on the energy formulation of Nitsche [38], which does not involve a Lagrange multiplier. Hence we need not solve a constrained minimization problem, and the stochastic gradient descent (SGD) method may be used to solve the resulting minimization problem. To analyze the method, we exploit Nitsche's energy formulation instead of the Euler-Lagrange equations associated with the minimization problem, which in general does not exist for the deep Nitsche

method because the trial function set formed by the deep neural network is a manifold instead of a space. We prove the energy error bound of the deep Nitsche method without taking into account the error caused by SGD. The error bound consists of two parts. The first part is the approximation error caused by the underlying deep neural network, and the second part is the estimation error, which comes from the numerically evaluation of the energy functional, equivalently, the loss function. Such error structure bears certain similarity with the *first Lemma of Strang* [7]. We test the method with some mixed boundary value problems in two dimension with smooth solution and singular solution. The variational formulation may be adapted for solving nonlinear problem such as p-Laplace equation. We also apply the method to solve high dimensional problems with Dirichlet boundary condition. In all these cases, the solutions can be well approximated by the proposed method at a relative accuracy of $10^{-2}\sim10^{-3}$, with only $10^3\sim10^4$ parameters for 2d problems and $10^4\sim10^5$ parameters for high dimensional problems.

The rest of the paper is as follows. In the next part, we introduce the energy formulation of the method and an abstract error bound is proved by the aid of the energy formulation. In Section 3, we propose the deep Nitsche method, and then we present the numerical results in Section 4 for solving some mixed boundary value problems with regular and singular solutions in two-dimension and also for problems in high-dimension up to 100. In the last section, we conclude with some remarks.

## 2 Nitsche's variational formulation

We consider the following mixed boundary value problem

$$\begin{cases} -\nabla\cdot(A(x)\nabla u)=f & \text{in } \Omega, \\ u=g_D & \text{on } \Gamma_D, \\ \dfrac{\partial u}{\partial \nu}=g_N & \text{on } \Gamma_N, \end{cases} \tag{2.1}$$

where $\Omega$ is a bounded domain in $\mathbb{R}^d$, and $\Gamma_D\cup\Gamma_N=\partial\Omega$ and $\overline{\Gamma}_D\cap\overline{\Gamma}_N\neq\emptyset$. The conormal derivative of $u$ is defined as $\partial_\nu u=n_iA_{ij}\partial_{x_j}u$ with $n=(n_1,\cdots,n_d)$ the outer normal of $\Omega$. We assume that $A$ is a symmetric matrix with

$$\lambda|\xi|^2\leq A_{ij}(x)\xi_i\xi_j\leq\Lambda|\xi|^2 \qquad \text{a.e.} \quad x\in\Omega \quad \text{and} \quad \xi\in\mathbb{R}^d.$$

The minimization problem is defined as

$$I[u_n]=\min_{v\in\mathcal{H}_n} I[v] \tag{2.2}$$

with

$$\begin{aligned} I[v]=&\frac{1}{2}\int_\Omega A\nabla v\cdot\nabla v\mathrm{d}x+\frac{\beta}{2}\int_{\Gamma_D}(g_D-v)^2\mathrm{d}\sigma(x)+\int_{\Gamma_D}(g_D-v)\partial_\nu v\mathrm{d}\sigma(x)\\ &-\left(\int_\Omega fv\mathrm{d}x+\frac{\beta}{2}\int_{\Gamma_D}g_D^2\mathrm{d}\sigma(x)+\int_{\Gamma_N}g_Nv\mathrm{d}\sigma(x)\right), \end{aligned}$$

where $\beta$ is a positive parameter. Here $\mathcal{H}_n$ is the set with neural network functions with $n$ the size of the set. For example, we define a set for a two-layer shallow network as

$$\mathcal{H}_n := \left\{ v = \sum_{i=1}^{n} a_i \sigma(b_i \cdot x + c_i) \,\middle|\, a_i, c_i \in \mathbb{R},\ b_i \in \mathbb{R}^d \right\}$$

with the activation function $\sigma$. We assume that the activation function is smooth such that $\mathcal{H}_n \subset H^2(\Omega)$. Therefore, $I[v]$ is well-defined for all $v \in \mathcal{H}_n$.

To study the Nitsche's variational problem, one usually resorts to the associated Euler-Lagrange equation as in [35]. Unfortunately, there is no such Euler-Lagrange equation for the minimization problem (2.2) because $\mathcal{H}_n$ is a manifold instead of a subspace. This is one of the main difficulties in analyzing neural network-based numerical method. We overcome this difficulty by exploiting the original energy formulation of Nitsche [38].

**Lemma 2.1.** *The minimization problem* (2.2) *is equivalent to*

$$\widetilde{I}[u - u_n] = \min_{v \in \mathcal{H}_n} \widetilde{I}[u - v], \tag{2.3}$$

*where*

$$\widetilde{I}[v] := \frac{1}{2} \int_\Omega A \nabla v \cdot \nabla v \mathrm{d}x - \int_{\Gamma_D} v \frac{\partial v}{\partial \nu} \mathrm{d}\sigma(x) + \frac{\beta}{2} \int_{\Gamma_D} v^2 \mathrm{d}\sigma(x). \tag{2.4}$$

*Proof.* We start with

$$\begin{aligned} \widetilde{I}[u - v] = \widetilde{I}[u] + \widetilde{I}[v] \\ - \int_\Omega A \nabla u \cdot \nabla v \mathrm{d}x + \int_{\Gamma_D} (u \partial_\nu v + v \partial_\nu u) \mathrm{d}\sigma(x) - \beta \int_{\Gamma_D} uv \mathrm{d}\sigma(x). \end{aligned}$$

Multiplying $(2.1)_1$ by $v$, an integration by parts yields

$$\begin{aligned} \int_\Omega A(x) \nabla u \cdot \nabla v \mathrm{d}x &= \int_\Omega f v \mathrm{d}x + \int_{\partial\Omega} \partial_\nu u v \mathrm{d}\sigma(x) \\ &= \int_\Omega f v \mathrm{d}x + \int_{\Gamma_N} g_N v \mathrm{d}\sigma(x) + \int_{\Gamma_D} \partial_\nu u v \mathrm{d}\sigma(x). \end{aligned}$$

A combination of the above two equation yields

$$\widetilde{I}[u - v] = \widetilde{I}[u] + I[v] \tag{2.5}$$

with $I[v]$ given by (2.2). This proves the equivalence between the minimization problems (2.2) and (2.3). □

The following lemma states that Nitsche's method is consistent if the solution is smooth enough.

**Lemma 2.2.** *Let $\Omega$ be a Lipschitz domain. If $u\in H^2(\Omega)$, then $u$ is a critical point of the minimization problem*

$$\min_{v\in H^2(\Omega)} \widetilde{I}[u-v]. \tag{2.6}$$

The condition in Lemma 2.2 may be replaced by a weaker condition: $u,v\in H^s(\Omega)$ with $s>3/2$.

*Proof.* By the trace theorem [1], the assumption $u\in H^2(\Omega)$ implies that the conormal derivative $\partial_\nu u$ is well-defined and $\partial_\nu u\in L^2(\Omega)$. Therefore, it remains to prove that $w=0$ is a critical point of the minimization problem

$$\min_{w\in H^2(\Omega)} \widetilde{I}[w].$$

The Euler-Lagrange equation associates with the minimization problem $\min_{w\in H^2(\Omega)}\widetilde{I}[w]$ reads as

$$\int_\Omega A\nabla w\cdot\nabla v\mathrm{d}x+\int_{\Gamma_D} w(\beta v-\partial_\nu v)\mathrm{d}\sigma(x)-\int_{\Gamma_D}\partial_\nu w v\mathrm{d}\sigma(x)=0 \quad \text{for all } \ v\in H^2(\Omega).$$

Note that $w,v\in H^2(\Omega)$ guarantees that the Gauss-Green theorem holds, an integration by parts implies that $w$ satisfies (2.1) with $f=g_D=g_N=0$. Therefore, we conclude that $w\equiv 0$ by the uniqueness of the solution of the boundary value problem (2.1). □

In what follows, we assume that the following *inverse trace inequality* is valid: There exists a constant $\gamma$ such that

$$\|\nabla v\|_{L^2(\partial\Omega)}\le\gamma\|\nabla v\|_{L^2(\Omega)} \quad \text{for all } \ v\in\mathcal{H}_n. \tag{2.7}$$

We also make the following approximation assumptions:

$$\begin{aligned}&\inf_{v\in\mathcal{H}_n}\left(\|\nabla(u-v)\|_{L^2(\partial\Omega)}+\gamma\|\nabla(u-v)\|_{L^2(\Omega)}\right)\le\delta,\\ &\inf_{v\in\mathcal{H}_n}\|u-v\|_{L^2(\partial\Omega)}\le\delta_1.\end{aligned} \tag{2.8}$$

We shall derive the error estimate by assuming the existence of the global minimizer $u_n^*$, which in general need not exist. However, for any $\epsilon>0$, there always exists an $\epsilon-$suboptimal global minimizer $u_n^\epsilon\in\mathcal{H}_n$ in the sense that $I[u_n^\epsilon]\le\inf_{v\in\mathcal{H}_n}I[v]+\epsilon$. We refer to [27] for a detailed discussion on the properties of the $\epsilon-$suboptimal global minimizer. All the error estimates remain valid if we replace the global minimizer to the $\epsilon-$suboptimal global minimizer. In what follows, without loss of generality, we assume the existence of at least one global minimizer for the minimization problem (2.2). Given the existence of the minimizer $u_n$, we exploit the minimization problem (2.2) and the identity (2.5) to obtain the error estimate in Theorem 2.1.

**Theorem 2.1.** *If $\beta>8\Lambda^2\gamma^2/\lambda$, the inverse trace inequality* (2.7) *and the approximation assumption* (2.8) *are valid, then the solution $u_n$ satisfies*

$$\|\nabla(u-u_n)\|_{L^2(\Omega)}+\sqrt{\beta}\|u-u_n\|_{L^2(\Gamma_D)}\leq C\left(\delta/\gamma+\delta/\sqrt{\beta}+\sqrt{\beta}\delta_1\right), \tag{2.9}$$

*where $C$ only depends on $\Lambda$ and $\lambda$.*

The above error estimate (2.9) may be written into a more convenient form:

$$\begin{aligned}&\|\nabla(u-u_n)\|_{L^2(\Omega)}+\sqrt{\beta}\|u-u_n\|_{L^2(\Gamma_D)}\\ \leq& C\inf_{v\in\mathcal{H}_n}\left(\left(\frac{1}{\gamma}+\frac{1}{\sqrt{\beta}}\right)\left(\|\nabla(u-v)\|_{L^2(\Gamma_D)}+\gamma\|\nabla(u-v)\|_{L^2(\Omega)}\right)+\sqrt{\beta}\|u-v\|_{L^2(\Gamma_D)}\right).\end{aligned} \tag{2.10}$$

The error estimate is based on the equivalence between the minimization problems (2.2) and (2.3).

*Proof.* Denote $e:=u-u_n$. For any $v\in\mathcal{H}_n$, using the inverse trace inequality (2.7) and the approximation assumption $(2.8)_1$, we obtain

$$\begin{aligned}\left|\int_{\Gamma_D}e\partial_\nu e\mathrm{d}\sigma(x)\right|\leq&\left|\int_{\Gamma_D}e\partial_\nu(u-v)\mathrm{d}\sigma(x)\right|+\left|\int_{\Gamma_D}e\partial_\nu(v-u_n)\mathrm{d}\sigma(x)\right|\\ \leq&\Lambda\|e\|_{L^2(\Gamma_D)}\left(\|\nabla(u-v)\|_{L^2(\Gamma_D)}+\gamma\|\nabla(v-u_n)\|_{L^2(\Omega)}\right)\\ \leq&\Lambda\gamma\|e\|_{L^2(\Gamma_D)}\|\nabla e\|_{L^2(\Omega)}\\ &+\Lambda\|e\|_{L^2(\Gamma_D)}\left(\|\nabla(u-v)\|_{L^2(\Gamma_D)}+\gamma\|\nabla(u-v)\|_{L^2(\Omega)}\right)\\ \leq&\Lambda\gamma\|e\|_{L^2(\Gamma_D)}\|\nabla e\|_{L^2(\Omega)}+\Lambda\delta\|e\|_{L^2(\Gamma_D)}.\end{aligned}$$

Using (2.4) and the above inequality, we obtain

$$2\widetilde{I}[e]\geq\lambda\|\nabla e\|^2_{L^2(\Omega)}+\beta\|e\|^2_{L^2(\Gamma_D)}-2\Lambda\gamma\|e\|_{L^2(\Gamma_D)}\|\nabla e\|_{L^2(\Omega)}-2\Lambda\delta\|e\|_{L^2(\Gamma_D)}.$$

By Cauchy-Schwartz inequality and the fact that $\beta>8\Lambda^2\gamma^2/\lambda$, we obtain

$$\begin{aligned}2\widetilde{I}[e]\geq&\lambda\|\nabla e\|^2_{L^2(\Omega)}+\beta\|e\|^2_{L^2(\Gamma_D)}-\frac{\lambda}{2}\|\nabla e\|^2_{L^2(\Omega)}-\frac{2\Lambda^2\gamma^2}{\lambda}\|e\|^2_{L^2(\Gamma_D)}\\ &-\frac{\beta}{2}\|e\|^2_{L^2(\Gamma_D)}-2\Lambda^2\delta^2/\beta\\ =&\frac{\lambda}{2}\|\nabla e\|^2_{L^2(\Omega)}+\left(\frac{\beta}{2}-\frac{2\Lambda^2\gamma^2}{\lambda}\right)\|e\|^2_{L^2(\Gamma_D)}-2\Lambda^2\delta^2/\beta\\ \geq&\frac{\lambda}{2}\|\nabla e\|^2_{L^2(\Omega)}+\frac{\beta}{4}\|e\|^2_{L^2(\Gamma_D)}-2\Lambda^2\delta^2/\beta.\end{aligned}$$

Next, using (2.3), we obtain that for any $v\in\mathcal{H}_n$, there holds

$$\begin{aligned}2\widetilde{I}[e]\leq 2\widetilde{I}[u-v]\leq\Lambda\|\nabla(u-v)\|^2_{L^2(\Omega)}+2\Lambda\|u-v\|_{L^2(\Gamma_D)}\|\nabla(u-v)\|_{L^2(\Gamma_D)}\\ +\beta\|u-v\|^2_{L^2(\Gamma_D)}\\ \leq\frac{\Lambda\delta^2}{\gamma^2}+2\Lambda\delta_1\delta+\beta\delta_1^2,\end{aligned}$$

where we have used the approximation properties (2.8) in the last step.

A combination of the above two inequalities gives

$$\begin{aligned}\frac{\lambda}{2}\|\nabla e\|^2_{L^2(\Omega)}+\frac{\beta}{4}\|e\|^2_{L^2(\Gamma_D)}&\leq\frac{\Lambda\delta^2}{\gamma^2}+2\Lambda\delta_1\delta+\beta\delta_1^2+2\Lambda^2\delta^2/\beta\\ &\leq\frac{\Lambda\delta^2}{\gamma^2}+3\Lambda^2\delta^2/\beta+2\beta\delta_1^2.\end{aligned}$$

This implies

$$\|\nabla(u-u_n)\|_{L^2(\Omega)}\leq\max(\sqrt{2\Lambda/\lambda},\sqrt{6\Lambda^2/\lambda},2/\sqrt{\lambda})\left(\delta/\gamma+\delta/\sqrt{\beta}+\sqrt{\beta}\delta_1\right),$$

and

$$\sqrt{\beta}\|u-u_n\|_{L^2(\Gamma_D)}\leq 2\max(\sqrt{\Lambda},\sqrt{3}\Lambda,\sqrt{2})\left(\delta/\gamma+\delta/\sqrt{\beta}+\sqrt{\beta}\delta_1\right).$$

Combining the above two inequalities, we obtain (2.9). □

In what follows, we consider the numerical integration of Nitsche's formulation.

$$I^*[u_n^*]=\min_{v\in\mathcal{H}_n}I^*[v],\tag{2.11}$$

where $I^*[v]$ is the Monte Carlo approximation or Quasi-Monte Carlo approximation [14] of $I[v]$ or other numerical integration schemes acting on $I[v]$.

**Theorem 2.2.** *If the inverse trace inequality* (2.7) *is true, then there exists $C$ that depends only on $\Lambda$ and $\lambda$ such that the minimizer $u_n^*\in\mathcal{H}_n$ satisfies*

$$\begin{aligned}&\|\nabla(u-u_n^*)\|_{L^2(\Omega)}+\sqrt{\beta}\|u-u_n^*\|_{L^2(\Gamma_D)}\\ &\leq C\inf_{v\in\mathcal{H}_n}\Big((1/\gamma+1/\sqrt{\beta})\left(\|\nabla(u-v)\|_{L^2(\Gamma_D)}+\gamma\|\nabla(u-v)\|_{L^2(\Omega)}\right)+\sqrt{\beta}\|u-v\|_{L^2(\Gamma_D)}\\ &\qquad+2(1/\sqrt{\lambda}+\sqrt{2})\,|I^*[v]-I[v]|^{1/2}\Big)\\ &+2(1/\sqrt{\lambda}+\sqrt{2})\,|I^*[u_n^*]-I[u_n^*]|^{1/2}.\end{aligned}\tag{2.12}$$

The above inequalities (2.10) and (2.12) give the error estimate for the Deep Nitsche method without taking into account the iteration error in SGD. The first term in the right-hand side of both inequalities is the approximation error, which may be bounded once $\mathcal{H}_n$ is specified, and we shall discuss this in the next part. The second term and the third term are the estimation error, i.e., the consistency error in the sense of numerical analysis. The estimate for these terms are standard, and we refer to [14] for a review. The difference between the above estimate and the first Strang lemma [7] in finite element is the last term in the right-hand side of (2.12), which depends on the approximating solution $u_n^*$, which usual appears in the error estimate of the nonlinear problems, while there is no such term in the first lemma of Strang.

*Proof.* We start with the identity (2.5) and note that

$$\begin{aligned}\widetilde{I}[u-u_n^*] &= \widetilde{I}[u]+I[u_n^*]\\ &= \widetilde{I}[u]+I^*[u_n^*]+I[u_n^*]-I^*[u_n^*]\\ &\le \widetilde{I}[u]+I^*[v]+I[u_n^*]-I^*[u_n^*]\\ &= \widetilde{I}[u]+I[v]+I^*[v]-I[v]+I[u_n^*]-I^*[u_n^*]\\ &= \widetilde{I}[u-v]+(I^*[v]-I[v])+(I[u_n^*]-I^*[u_n^*]),\end{aligned}$$

where we have used the minimization problem (2.2) in the last step.

Proceeding along the same line that leads to (2.9), we obtain (2.12). □

## 3 Deep Nitsche Method

We minimize $I[v]$ over certain trial set $\mathcal{H}_n$ that will be specified below. We shall omit the subscript $n$ in what follows to avoid the cluttering of the notations. The resulting optimization problem is solved by the standard Stochastic Gradient Descent (SGD) method [22, §8].

$$\hat{u} = \operatorname{argmin}_{v\in\mathcal{H}} I[v]. \tag{3.1}$$

The trial function set $\mathcal{H}$ is modeled by ResNet [26]. The component of ResNet is shown in Fig. 1. The input layer is a fully connected layer with $m$ hidden nodes, which maps $x$ from $\mathbb{R}^d$ to $\mathbb{R}^m$. Assume that $\sigma$ is a scalar activation function and let $\phi$ be the tensor product of $\sigma$ as $\phi(x)=(\sigma(x_1),\cdots,\sigma(x_m))\in\mathbb{R}^m$, then

$$s_1 = \phi(W_1x+b_1),$$

where $W_1\in\mathbb{R}^{m\times d}$ and $b_1\in\mathbb{R}^m$. The hidden layers is constructed by $l$ residual blocks. Each block contains two fully connected layers and one residual connection layer. The $i-$th block takes the form

$$s_{i+1} = \phi(W_{2,i}\phi(W_{1,i}s_i+b_{1,i})+b_{2,i})+s_i,$$

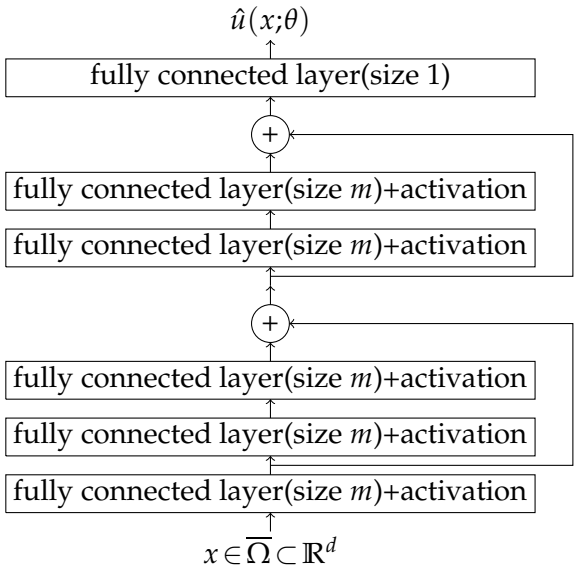


Figure 1: The component of ResNet.

where $W_{1,i},W_{2,i}\in\mathbb{R}^{m\times m}$ and $b_{1,i},b_{2,i}\in\mathbb{R}^m$.

The output layer is a fully connected layer with one hidden node. The approximation solution may be represented as

$$\hat{u}(x;\theta)=W_2 s_{l+1}+b_2,$$

where $W_2\in\mathbb{R}^{1\times m}$ and $b_2\in\mathbb{R}$, the parameter set $\theta$ is defined as

$$\theta:=\{W_1,W_2,b_1,b_2,W_{1,i},W_{2,i},b_{1,i},b_{2,i}\,|\,i=1,\cdots,l\}.$$

In each step of the SGD iteration, we randomly sample $N_i$ points $\{x_k^{(i)}\}_{k=1}^{N_i}\subset\Omega$, $N_d$ points $\{x_k^{(d)}\}_{k=1}^{N_d}\subset\Gamma_D$ and $N_n$ points $\{x_k^{(n)}\}_{k=1}^{N_n}\subset\Gamma_N$. The loss function is defined as

$$\begin{aligned}I^*[v]:=&\frac{|\Omega|}{N_i}\sum_{k=1}^{N_i}\left(\frac{1}{2}A(x_k^{(i)})\nabla_x\hat{u}(x_k^{(i)};\theta)\cdot\nabla_x\hat{u}(x_k^{(i)};\theta)-f(x_k^{(i)})\hat{u}(x_k^{(i)};\theta)\right)\\&+\frac{|\Gamma_D|}{N_d}\sum_{k=1}^{N_d}\left(\frac{\beta}{2}\hat{u}^2(x_k^{(d)};\theta)-\hat{u}(x_k^{(d)};\theta)A(x_k^{(d)})\nabla_x\hat{u}(x_k^{(d)};\theta)\cdot n\right)\\&-\frac{|\Gamma_D|}{N_d}\sum_{k=1}^{N_d}g_D\left(\beta\hat{u}(x_k^{(d)};\theta)-A(x_k^{(d)})\nabla_x\hat{u}(x_k^{(d)};\theta)\cdot n\right)\\&-\frac{|\Gamma_N|}{N_n}\sum_{k=1}^{N_n}g_N\hat{u}(x_k^{(n)};\theta).\end{aligned}$$

Now the minimization problem reads as

$$\min_{v\in\mathcal{H}} I^*[v].$$

In what follows, we discuss the theoretical assumptions on Theorem 2.2. The inverse trace inequality (2.7) seems to has been absent for neural network functions, which is even missing for the meshless methods; c.f., [35, §7]. For a function $v$ representing by a two-layer Gaussian network, MHASKAR [36, 37] and ERDÉLYI [18] proved the following inverse inequality:

$$\|\nabla^2 v\|_{L^2(\mathbb{R})} \le C_{\text{inv}}\sqrt{n}\|\nabla v\|_{L^2(\mathbb{R})},$$

where $C_{\text{inv}}$ is an absolute constant, and $n$ is the number of the neurons. However, it does not seem easy to extend the proof of the above inequality in [18, Theorem 2.2] to a finite interval. Once the above type inequality is valid for a bounded domain with $\sqrt{n}$ replaced by $n^\alpha$ with $\alpha>1/2$, then we may use the trace inequality to prove (2.7) with $\gamma=C_{\text{trace}}(1+C_{\text{inv}})n^\alpha$, where $C_{\text{trace}}$ is a constant appears in the classical trace inequality [1], which may depend on $\Omega$ but independent of $v$ and $n$.

The approximation estimates (2.8) for two-layer neural networks is well-established, and we refer to [39] for a review. While such results for multi-layer neural networks are not so complete, we refer to [9, 21] for the progress in this direction. Most estimates in this vein except [34] is asymptotical in the sense that the approximation bound is valid for specified width and depth of a multi-layer neural network. But what we need is a sharp approximation bound for a deep neural network with arbitrary width and depth in the energy norm. The way for bounding the estimation error is quite standard [14] provided that we assume certain smoothness on the functions in $\mathcal{H}_n$. Therefore, we may obtain the rate of convergence by combining these two type errors, and we shall leave it for further study.

## 4 Numerical experiments

We apply the Deep Nitsche Method to solve the mixed boundary value problem (2.1). In all the examples, we use the activation function $\sigma=\tanh$, and let the domain $\Omega$ be a hypercube unless otherwise stated, i.e., $\Omega=(0,1)^d$. We report the relative errors

$$e_{L^2}:=\frac{\|u-\hat{u}\|_{L^2}}{\|u\|_{L^2}} \quad \text{and} \quad e_{H^1}:=\frac{\|u-\hat{u}\|_{H^1}}{\|u\|_{H^1}}$$

for all examples.

There are lots of choices for generating sampling points to approximate the energy functional $I[v]$ during the training process, which is a crucial part for the efficiency of the method. The number of the uniform sampling points grows exponentially with the dimension, hence it quickly becomes unpractical. We use Quasi-Monte Carlo method [14] to approximate $I$, and the sampling points are generated by the Halton sequence [24]. To

be more specific, we use the growing prime values starting from 2 as the prime bases in each dimension. For example, in three dimensional problem, the prime base is 2 for $x$-axis, 3 for $y$-axis, and 5 for $z$-axis, respectively. There are many other sets of the low discrepancy sequences beyond the Halton sequence. However, an evaluation of their efficiency seems beyond the scope of the present work. Quasi-Monte Carlo is also used to approximate the relative errors $e_{H^1}$ and $e_{L^2}$ during the test process. We generate $10^5$ sampling points in $\Omega$ by the Halton sequence, with the same prime bases as above.

### 4.1 Two-dimensional examples

The solution $u$ is approximated by a neural network with five residual blocks and 10 hidden nodes per fully connected layer. Noticing that one residual block contains 2 fully connected layers and 1 residual connection, the number of trainable parameters is 1141. An Adam optimizer is employed to train with the learning rate 0.001 [28]. We train the model for 50000 epochs. For simplicity, we take the same number of points in the domain and on each of boundaries. In each epoch, we generate 64 points inside the domain $\Omega$ and 64 points on each edge of $\partial\Omega$, by a Quasi-Monte Carlo method based on low-discrepancy Halton sequence as discussed above. In order to save memory and running time, and at the same time to achieve a basically equivalent convergence effect, we set hyper-parameters, such as the number of layers and neural nodes, and the size of training set very small, because we train a relatively small neural network in this part.

In the first example, we test a mixed boundary value problem with the coefficient matrix $A$ given by

$$A=\begin{pmatrix}(x+1)^2+y^2 & -xy\\ -xy & (x+1)^2\end{pmatrix}.$$

We let the solution be

$$u(x,y)=x^3y^2+x\sin(2\pi xy)\sin(2\pi y),$$

and we set $\Gamma_D=\{1\}\times(0,1)\cup(0,1)\times\{1\}$ and $\Gamma_N=\{0\}\times(0,1)\cup(0,1)\times\{0\}$. The source term $f$ and the boundary data $g_D$ and $g_N$ are computed by (2.1). The relative errors $e_{L^2}$ and $e_{H^1}$ decrease with the number of iterations, as shown in Fig. 2, and the final relative errors are reported in Table 1, with different penalized parameters $\beta$.

In view of Fig. 2 and Table 1, it seems the accuracy for methods with different parameters $\beta$ are comparable, and method with bigger $\beta$ yields slightly better results.

Table 1: A mixed boundary value problem in two dimension with different penalized parameter $\beta$.

| $\beta$ | $e_{L^2}$ | $e_{H^1}$ |
|---|---|---|
| 500 | 3.925e-02 | 6.960e-02 |
| 1000 | 3.633e-02 | 7.981e-02 |
| 2000 | 2.036e-02 | 5.124e-02 |

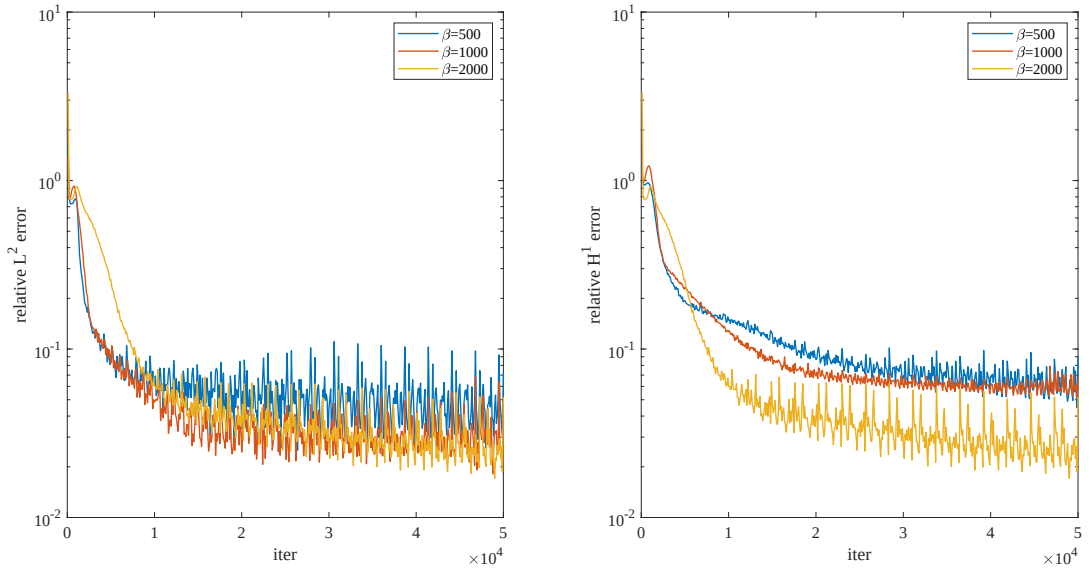


Figure 2: A mixed boundary problem in two dimension with different penalized parameter $\beta$.

In the second example, we consider

$$\begin{cases} -\triangle u(x)=0, & x\in\Omega, \\ u(x)=u(r,\theta)=r^{1/2}\sin(\theta/2), & x\in\partial\Omega, \end{cases}$$

where $\Omega=(-1,1)^2\backslash[0,1)\times\{0\}$ is a cracked domain. This problem has an analytical solution

$$u(x)=r^{1/2}\sin(\theta/2),$$

which belongs to $H^s(\Omega)$ with $s<3/2$. In fact, such solution usually stands for the singular part of the general solution [43]. We report the relative errors in Fig. 3 and Table 2 with different penalized parameters $\beta$.

In view of Fig. 3 and Table 2, the accuracy for the methods with three different parameters $\beta$ are also comparable. By contrast to the previous example, the method with smaller parameter gives better $L^2$ error.

Table 2: A singular solution in two dimension.

| $\beta$ | $e_{L^2}$ | $e_{H^1}$ |
|---|---|---|
| 500 | 5.298e-03 | 8.513e-02 |
| 1000 | 7.226e-03 | 9.148e-02 |
| 2000 | 1.019e-02 | 5.177e-02 |

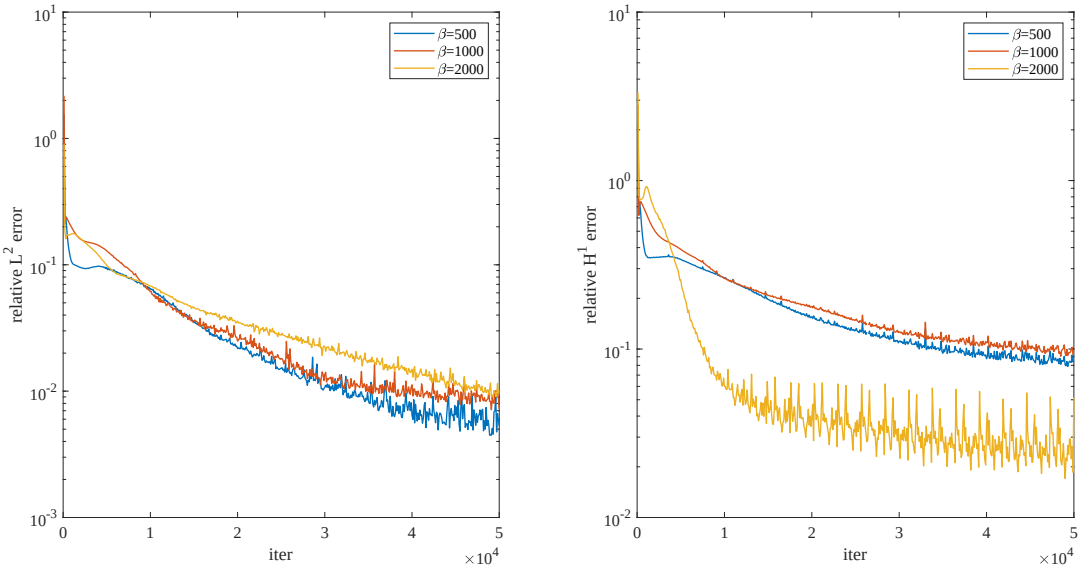


Figure 3: A singular solution in two dimension.

## 4.2 $p$-Laplace equation

In this part, we solve the $p$-Laplace equation posed on a unit square $\Omega=(0,1)^2$,

$$\begin{cases} -\nabla\cdot\left(|\nabla u|^{p-2}\nabla u\right)=f, & x\in\Omega, \\ u(x)=g_D, & x\in\partial\Omega. \end{cases}$$

The energy functional $I$ is written as

$$\begin{aligned} I[v]=&\frac{1}{p}\int_\Omega|\nabla v|^p\mathrm{d}x+\frac{\beta}{2}\int_{\partial\Omega}(g_D-v)^2\mathrm{d}\sigma(x)+\int_{\partial\Omega}(g_D-v)\partial_\nu v\mathrm{d}\sigma(x)\\ &-\left(\int_\Omega fv\mathrm{d}x+\frac{\beta}{2}\int_{\partial\Omega}g_D^2\mathrm{d}\sigma(x)\right), \end{aligned}$$

where $\partial_\nu v=|\nabla v|^{p-2}\partial_n v$.

In order to reduce unnecessary effects of hyper-parameters, we keep the configurations of the neural networks the same as that for the linear problem.

Firstly we test the classical example from [20]

$$u=2^{-1/(p-1)}(1-1/p)\left(1-r^{p/(p-1)}\right),\quad r=(x^2+y^2)^{1/2}.$$

A direct calculation gives that $f=1$. We note that $\nabla u(x)\to 0$ as $x\to 0$. Such singularity may cause difficulties in computation. We report the relative errors in Fig. 4 and Table 3 with different parameters $p$ and $\beta$.

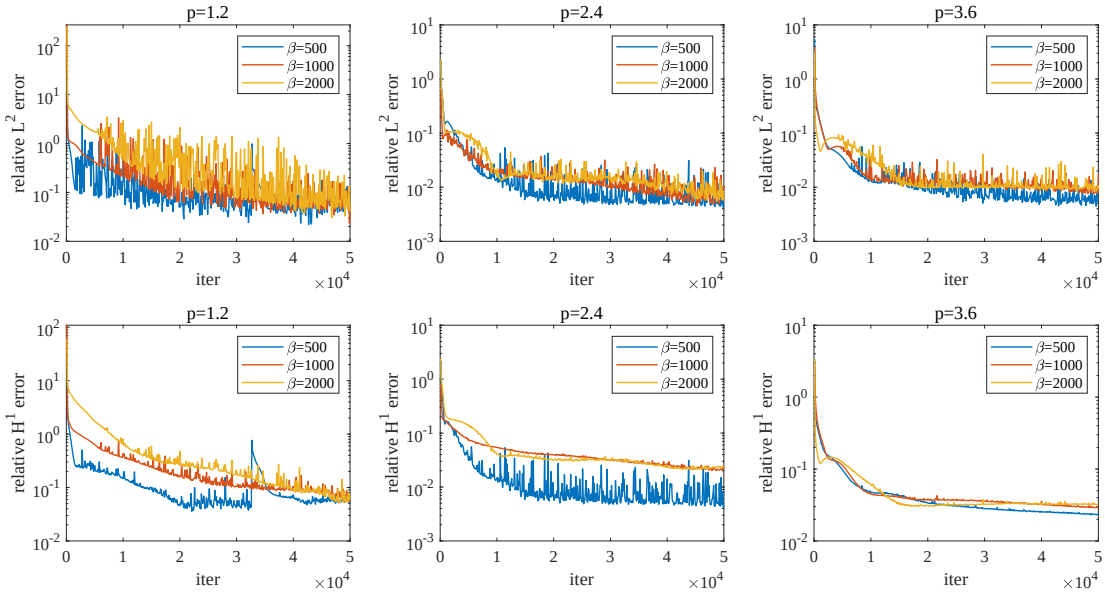


Figure 4: A smooth solution for $p$-Laplace equation: for $p=1.2$, 2.4 and 3.6, the upper figures show the relative $L^2$ error and the lower figures show the relative $H^1$ error.

Table 3: A smooth solution for $p$-Laplace equation.

| $p$ | $\beta$ | $e_{L^2}$ | $e_{H^1}$ |
|---|---|---|---|
| 1.2 | 500 | 6.968e-02 | 6.426e-02 |
| | 1000 | 4.993e-02 | 5.760e-02 |
| | 2000 | 4.057e-02 | 5.802e-02 |
| 2.4 | 500 | 4.646e-03 | 4.646e-03 |
| | 1000 | 1.092e-02 | 2.410e-02 |
| | 2000 | 5.835e-03 | 2.371e-02 |
| 3.6 | 500 | 6.310e-03 | 2.397e-02 |
| | 1000 | 1.747e-02 | 2.974e-02 |
| | 2000 | 9.412e-03 | 3.225e-02 |

In the second example, we test a less smooth solution:

$$u = r^{(p-2)/(p-1)}.$$

A direct calculation gives that $f=0$. The solution $u$ does not belong to $H^2(\Omega)$ when $p>2$, and $\nabla u \to \infty$ as $x \to 0$. We report the relative errors in Fig. 5 and Table 4.

In view of Figs. 4, 5, Tables 3, 4, Deep Nitsche Method equally works for the $p$-Laplace equation with small $p$ as well as large $p$. For the problem with less smooth solution, it seems wise to choose larger $\beta$ as $p$ grows.

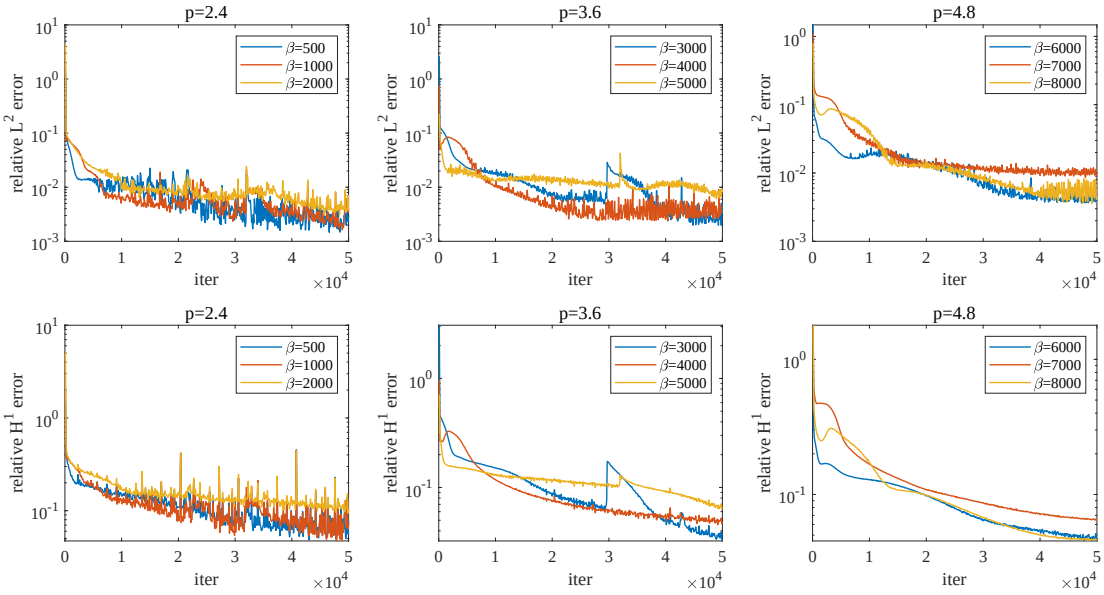


Figure 5: A singular solution for $p$-Laplace equation. The upper figures show the relative $L^2$ error and the lower figures show the relative $H^1$ error.

Table 4: A singular solution for $p$-Laplace equation: for $p=2.4$, 3.6 and 4.8

| $p$ | $\beta$ | $e_{L^2}$ | $e_{H^1}$ |
|---|---|---|---|
| 2.4 | 500 | 2.920e-03 | 7.758e-02 |
| | 1000 | 3.786e-03 | 7.312e-02 |
| | 2000 | 3.680e-03 | 1.035e-01 |
| 3.6 | 3000 | 3.294e-03 | 3.230e-02 |
| | 4000 | 3.332e-03 | 4.699e-02 |
| | 5000 | 8.085e-03 | 6.393e-02 |
| 4.8 | 6000 | 4.752e-03 | 4.605e-02 |
| | 7000 | 9.039e-03 | 6.439e-02 |
| | 8000 | 4.540e-03 | 4.577e-02 |

## 4.3 High-dimensional examples

We turn to high dimensional problems. We still employ an Adam optimizer with the learning rate 0.001 and train the model for 50000 epochs. In each epoch, we use a Quasi-Monte Carlo method based on a low-discrepancy Halton sequence to generate 512 points inside the domain $\Omega$ and 64 points on each face of $\partial\Omega$.

In the first example, we consider the problem (2.1) on a hypercube $\Omega=(0,1)^{20}$ with pure Dirichlet boundary condition, for which we choose $f$ and $g_D$ such that the solution

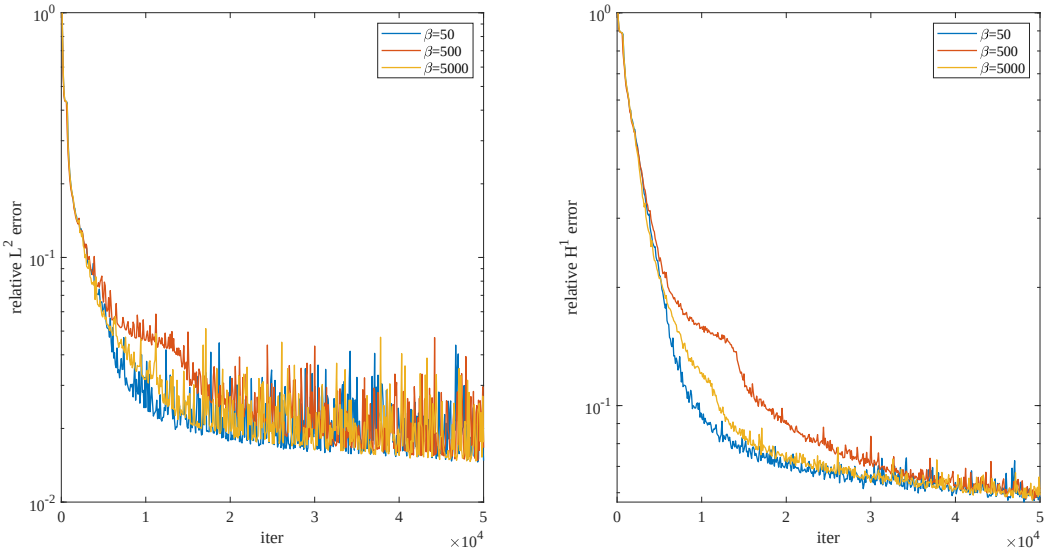


Figure 6: Less smooth solution in 20-dimension.

Table 5: Less smooth solution in 20 dimension.

| $\beta$ | $e_{L^2}$ | $e_{H^1}$ |
|---|---|---|
| 50 | 1.477e-02 | 5.807e-02 |
| 500 | 1.668e-02 | 6.137e-02 |
| 5000 | 1.756e-02 | 5.919e-02 |

to (2.1) is given by

$$u(x)=\left(\sum_{i=1}^{20}x_i^2\right)^{5/2}.$$

This example in three dimension has been test in [23] with a particle-partition of unit method. We approximate the solution by a neural network with five residual blocks and 50 hidden nodes per fully connected layer. Thus the number of trainable parameters is 26601. We report the relative errors in Fig. 6 and Table 5 with different penalized parameters $\beta$.

In the second example, we consider a smooth solution in 100 dimension,

$$u(x)=\exp\left(\frac{1}{100}\sum_{i=1}^{100}x_i\right),\quad x\in\Omega:=(0,1)^{100}$$

with a pure Dirichlet boundary condition. We compute $f$ and $g_D$ by (2.1). The exact solution $u$ is approximated by a neural network with five residual blocks and 100 hidden nodes per fully connected layer, and the number of trainable parameters is 111201. We report the relative errors in Fig. 7 and Table 6.

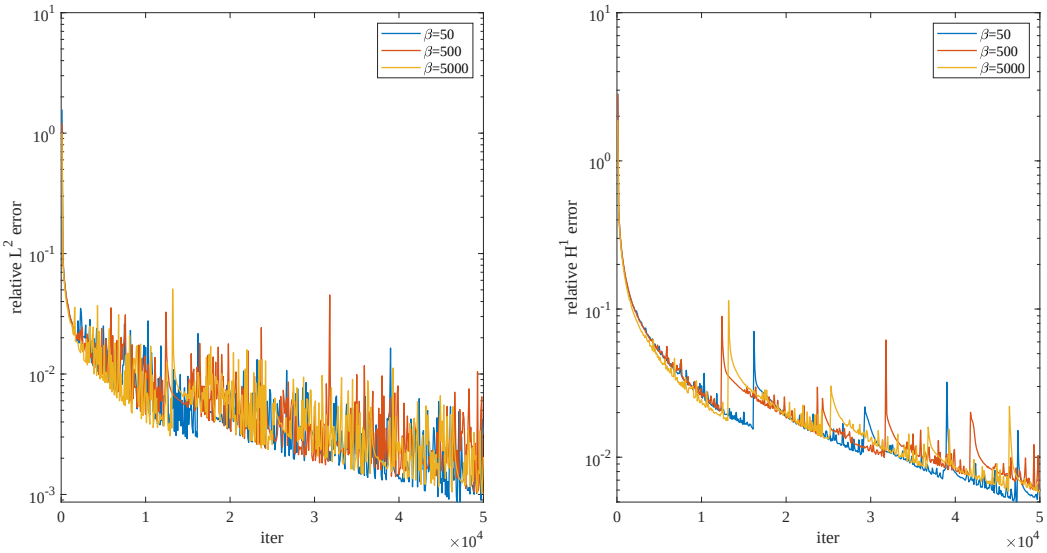


Figure 7: Smooth solution in 100-dimension.

Table 6: Smooth solution in 100 dimension.

| $\beta$ | $e_{L^2}$ | $e_{H^1}$ |
|---|---|---|
| 50 | 3.172e-03 | 5.924e-03 |
| 500 | 1.011e-03 | 5.850e-03 |
| 5000 | 2.049e-03 | 6.028e-03 |

Figs. 6, 7 and Tables 5, 6 show that our method has potential to work for boundary value problems in rather high dimension.

## 5 Conclusion

Based on Nitsche's idea and representing the trial functions by deep neural networks, we propose a new method to deal with the complicated boundary conditions for boundary value problems. The test examples and the error estimate show that the method has the following advantages:

1. It deals with the mixed boundary conditions in a unified variational way without significant extra costs, and it fits well with the stochastic gradient descent method. It lends itself to a rigorous error estimate.

2. It works for the problems in low dimension as well as high dimension. It also has potential to work for problems in rather high dimension. It equally works for nonlinear problems.

Besides the above remarks, it would be an interesting direction to extend the method to deal with the time-dependent problems, in particular for problems with mixed time varying boundary conditions. It is of great interest to prove the convergence rate of the method though we have derived an energy error bound. These will be left as future work.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China under the grant 11971467, and this work is also supported by Beijing Academy of Artificial Intelligence (BAAI). The computations were done on the high performance computers of the State Key Laboratory of Scientific and Engineering Computing (LSEC), Chinese Academy of Sciences.

### References

[1] R. Adams and J. J. R. Fourier, Sobolev Spaces, Academic Press, 2nd eds., 2003.
[2] A. K. Aziz, R. B. Kellogg and A. B. Stephens, Least squares methods for elliptic systems, Math. Comput., 44(1985), 53–70.
[3] I. Babuška, The finite element method with penalty, Math. Comp., 27(1973), 221–228.
[4] I. Babuška, The finite element method with Lagrangian multipliers, Numer. Math., 20(1973), 179-192.
[5] I. Babuška, U. Banerjee, and J. E. Osborn, Surveys of meshless and generalized finite element method: a unified approach, Acta Numer., 12 (2003), 1-125.
[6] J. Berg and K. Nystrom, A unified deep artificial neural network approach for partial differential equations in complex geometries, Neural Computing, 317 (2018), 28-41.
[7] A. Berger, R. Scott, and G. Strang, Approximate boundary conditions in the finite element method, Symposia Mathematica, X (1972), 295-313.
[8] P. B. Bochev and M. D. Gunzburg, Least-Squares Finite Element Methods, Springer Science+Business Media, LLC 2009.
[9] H. Bölcskei, P. Grohs, G. Kutyniok and P. Petersen, Optimal approximation with sparsely connected deep neural network, SIAM J. Math. Data Sci., 1(2019), 8-45.
[10] E. Burman and P. Zunino, Numerical approximation of large contrast problems with the unfitted Nitsche method, Frontiers in Numerical Analysis-Durham 2010, J. Blowey and M. Jensen Eds., Springer-Verlag Berlin Heidelberg, 2012, pp. 227-281.
[11] J. R. Chen, R. Du, P. C. Li and L. Y. Lyu, Quasi-Monte Carlo sampling for machine-learning partial differential equations, arXiv:1911.01612 (2019).
[12] J. R, Chen, R. Du and K. K. Wu, A comparison study of Deep Galerkin method and Deep Ritz method for elliptic problems with different boundary conditions, Commun. Math. Res., 36 (2020), 354-376.
[13] W. Dahmen and A. Kunoth, Appending boundary conditions by Lagrange multipliers: analysis of the LBB condition, Numer. Math., 88 (2001), 9-42.
[14] J. Dick, F. Y. Kuo and I. H. Sloan, High dimensional numerical integration–the Quasi-Monte Carlo way, Acta Numerica, 22(2013), 133-288.
[15] W. E, Machine Learning and computational mathematics, Commun. Comput. Phys., 28 (2020), 1639-1670.

[16] W. E, J. Q. Han, and A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat., 5 (2017), 349-380.
[17] W. E and B. Yu, The Deep Ritz Method: a deep-learning based numerical algorithm for solving variational problems, Commun. Math. Stat., 6 (2018), 1-12.
[18] T. Erdélyi, Bernstein-type inequalities for linear combinations of shifted Gaussians, Bull. London Math. Soc., 38(2006), 124-138.
[19] Y. W. Fan, C. O. Bohorquez and L. X. Ying, BCR-Net: a neural network based on the non-standard wavelet form, J. Comput. Phys., 384(2019), 1-15.
[20] R. Glowinski and A. Marrocco, Sur lapproximation, par éléments finis dordre un, et la résolution, par pénalisation-dualité, d'une classe de problèmes de Dirichlet nonlinéaires, Rev. Francaise Automat. Informat. Recherche Opérationnelle Sér. Rouge. Anal. Numér., 9 (1975), 41-76.
[21] I. Gühring, G. Kutyniok and P. Petersen, Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms, Anal. Appl. (Singap.) 18 (2020), 803-859.
[22] Y. Goodfellow, I. Bengio and A. Courville, Deep Learning, MIT Press, Cambridge, 2016.
[23] M. Griebel and M. A. Schweitzer, A particle-partition of unit method part V: boundary conditions, Geometric Analysis and Nonlinear Partial Differential Equations, S. Hildebrandt et al eds., Springer-Verlag Berlin Heidelberg, 2003, pp. 519-542.
[24] J. H. Halton, on the efficiency of certain quasi-random sequences of points in evaluating multidimensional integrals, Numer. Math., 2(1960), 84–90.
[25] J. Q. Han, A. Jentzen, and W. E, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci., 115 (2018), no. 34, 8505-8510.
[26] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, Deep residual learning for image reconganition, In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2016), 770-778.
[27] B. Houska and B. Chachuat, Global optimization in Hilbert space, Math. Program., Ser. A 173(2019), 221-249.
[28] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980; Published as a conference paper at ICLR 2015.
[29] J. Khoo, J. Lu, and L. X. Ying, Solving for high dimensional committor functions using artificial neural networks, Res. Math. Sci., 6(2019), 1.
[30] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and 981 partial differential equations, IEEE Transactions on Neural Networks, 9(1998), 987-1000.
[31] I. E. Lagaris, A. C. Likas, and G. D. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, IEEE Transactions on Neural Networks, 11(2000), 1041-1049.
[32] X.-A. Li, Z.-Q. J. Xu and L. Zhang, A multi-scale DNN algorithm for nonlinear elliptic equations with multiple scales, Commun. Comput. Phys., 28 (2020), 1886-1906.
[33] Z. Q. Liu, W. Cai and Z.-Q. J. Xu, Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains, Commun. Comput. Phys., 28 (2020), 1970-2001.
[34] J. Lu, Z. W. Shen, H. Z. Yang and S. J. Zhang, Deep network approximation for smooth functions, arXiv: 2001.03040.
[35] J. M. Melenk, On approximation in meshless method, Frontiers of Numerical Analysis II, J. F. Blowey, and A. W. Craig, eds., Cambridge University, Cambridge, 2005, pp. 65-141.
[36] H. N. Mhaskar, When is approximation by Gaussian networks necessarily a linear process? Neural Networks, 17(2004), 989–1001.

[37] H. N. Mhaskar, A Markov-Bernstein inequality for Gaussian networks, Trends and Applications in Constructive Approximations, M. G. de Bruin, D. H. Mache & J. Szabados (Eds.) International Series of Numerical Mathematics, Vol. 151, 165-180, 2005, Birkhäuser Verlag Basel/Switzerland.

[38] J. Nitsche, Über ein Variationsprinzip zur Lösung von Drichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind, Abh. Math. Sem. Univ. Hamburg, 36 (1971), 9-15.

[39] A. Pinkus, Approximation theory of the MLP model in neural network, Acta Numerica, 8(1999), 143-195.

[40] M. Rassi, 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, J. Comput. Phys., 378(2019), 686–707.

[41] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys., 375 (2018), 1339-1364.

[42] R. Stenberg, On some techniques for approximating boundary conditions in the finite element method, J. Comput. Appl. Math., 63(1995), 139-148.

[43] G. Strang and G. J. Fix, An Analysis of the Finite Element Method, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1973.

[44] B. Wang, W. Z. Zhang and W. Cai, Multi-scale deep neural network (MscaleDNN) methods for oscillatory Stokes flows in domplex domains, Commun. Comput. Phys., 28 (2020), 2139-2157.

[45] D. Zhang, L. Guo and G. E. Karniadakis, Learning in modal space: solving time-dependent stochastic pdes using physics-informed neural networks, SIAM J. Sci. Comput., 42(2020), A639–A665.

[46] Y. H. Zhang, G. Bao, X. J. Ye and H. M. Zhou, Weak adversarial networks for high-dimensional differential equations, J. Comput. Phys., 411(2020), 109409.