Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2021-0201

Vol. **31**, No. 4, pp. 1162-1179
April 2022

Deep Unfitted Nitsche Method for Elliptic Interface Problems

Hailong Guo[1,*] and Xu Yang[2]

[1] *School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC 3010, Australia.*
[2] *Department of Mathematics, University of California, Santa Barbara, CA, 93106, USA.*

Received 15 October 2021; Accepted (in revised version) 9 January 2022

Abstract. This paper proposes a deep unfitted Nitsche method for solving elliptic interface problems with high contrasts in high dimensions. To capture discontinuities of the solution caused by interfaces, we reformulate the problem as an energy minimization problem involving two weakly coupled components. This enables us to train two deep neural networks to represent two components of the solution in high-dimensional space. The curse of dimensionality is alleviated by using the Monte-Carlo method to discretize the unfitted Nitsche energy functional. We present several numerical examples to show the performance of the proposed method.

AMS subject classifications: 78M10, 78A48, 47A70, 35P99
Key words: Deep learning, unfitted Nitsche method, interface problem, deep neural network.

1 Introduction

In this paper, we continue our previous studies on elliptic interface problems [11, 14, 15], arising in many applications such as fluid dynamics and materials science, where the background consists of rather different materials on the subdomains separated by smooth curves (or surfaces) called interfaces. We aim to address the high-dimensional challenge, which is well known as the curse of dimensionality leading to unaffordable computational time in traditional numerical methods (*e.g.*, finite difference and finite element methods).

Deep neural networks have been shown as a powerful tool to overcome the curse of dimensionality [4, 6, 9, 37], and have been applied to solve partial differential equations (PDEs), *e.g.*, the deep BSDE method [8, 16], the deep Galerkin method (DGM) [33],

*Corresponding author. *Email addresses:* `hailong.guo@unimelb.edu.au` (H. Guo), `xuyang@math.ucsb.edu` (X. Yang) the physics-informed neural networks (PINNs) [31], the deep Ritz method (DRM) [7], and the weak adversarial networks (WAN) [38]. The deep BSDE reformulates the time-dependent equations into stochastic optimization problems. DGM and PINNs train neural networks by minimizing the mean squared error loss of the equation residual, while DRM trains networks by minimizing the energy functional of the variational problem equivalent to the PDEs. WAN uses the weak formulation and trains the primary and adversarial network alternatively using the min-max weak formulation. Moreover, the convergence of DRM was studied in [5, 27], and the deep Nitsche method was proposed in [26], which enhanced the deep Ritz method with natural treatment of essential boundary conditions. In a recent work [32], Sheng and Yang trained an additional neural network to impose the Dirichlet boundary conditions. However, in general, these neural network-based methods require the smoothness of the solutions to the PDEs. They thus can not be directly used to solve the elliptic interface problems, where the solutions are only piecewise smooth.

In literature, there are some recent works of solving elliptic interface problems using neural networks. For example, [36] proposed a network architecture similar to the deep Ritz method [7], and solved the equivalent variational problem with the boundary conditions approximated by a shallow neural network. [19] used different neural networks to approximate the solutions in disjoint subdomains. They reformulated the interface problem as a least-squares problem and solved it by stochastic gradient descent. [23] proposed the discontinuity capturing shallow neural network (DCSNN) to approximate piecewise continuous functions and solved elliptic interface problems by minimizing the mean squared error loss consisting of the residual of the equation, boundary and interface jump conditions.

In this paper, we propose a deep learning method for interface problems based on the minimization of the unfitted Nitsche energy functional, inspired by our previous studies [11–13] on the unfitted Nitsche method. One of the most significant differences between the unfitted Nitsche method [2, 12, 13, 17, 21] and other numerical methods for interface problems (*e.g.*, the immersed type numerical methods [25, 34, 35, 39]) is that the unfitted Nitsche finite element functions can be discontinuous inside elements. This is possible by adopting two different sets of basis functions on the interface elements (*i.e.*, the elements cut by the interface) which are weakly coupled together using Nitsche methods. Based on the unfitted Nitsche formulation, we can define the so-call unfitted Nitsche energy functional (see equation (2.11)). It turns out that the weak formulation of unfitted Nitsche method is just the Euler-Lagrange equation of unfitted Nitsche energy functional. To address the challenges of the curse of dimensionality, we naturally use deep neural networks to represent functions in high dimensions. Following the idea of classical unfitted Nitsche method [2, 12, 13, 17], we use two deep neural networks: one for the part inside the interface and the other one for the region outside the interface. These two parts are weakly connected using Nitsche method. The deep unfitted Nitsche method trains the two neural network functions independently using the same unfitted Nitsche energy functional.

The rest of the paper is organized as follows. In Section 2, we introduce the model setup of the elliptic interface problems and its unfitted Nitsche weak form. In Section 3, we describe the formulation of deep unfitted Nitsche method with details. We present numerical examples in Section 4, and make conclusive remarks in Section 5.

2 Model equation

Let Ω be a bounded Lipschitz domain in $\mathbb{R}^d$. Furthermore, we assume there is a smooth closed curve (or surface) Γ separating Ω into two parts: Ω_1 and Ω_2. In general, Γ can be described as a zero level set of some level set function ϕ. Then we have $\Omega_1=\{x\in\Omega|\phi(x)<0\}$ and $\Omega_2=\{x\in\Omega|\phi(x)>0\}$.

In this paper, we shall consider the following elliptic interface problem

$$-\nabla\cdot(\beta(x)\nabla u(x))=f(x), \quad \text{in } \Omega_1\cup\Omega_2, \tag{2.1a}$$
$$u=g, \quad \text{on } \partial\Omega, \tag{2.1b}$$
$$[\![u]\!]=p, \quad \text{on } \Gamma, \tag{2.1c}$$
$$[\![\beta\partial_n u]\!]=q, \quad \text{on } \Gamma, \tag{2.1d}$$

where $\partial_n u=(\nabla u)\cdot n$ with n being the unit outward normal vector of Γ and the jump $[\![w]\!]$ on Γ is defined as

$$[\![w]\!]=w_2|_\Gamma-w_1|_\Gamma, \tag{2.2}$$

with $w_i=w|_{\Omega_i}$ being the restriction of w on Ω_i. The diffusion coefficient $\beta(x)$ is a piecewise constant, *i.e.*,

$$\beta(x)=\begin{cases}\beta_1 & \text{if } x\in\Omega_1,\\ \beta_2 & \text{if } x\in\Omega_2,\end{cases} \tag{2.3}$$

which has a finite jump of function value at the interface Γ. Without loss of generality, we assume $\beta_0=\min(\beta_1,\beta_2)>0$. An illustration of the domain Ω with the interface Γ is given in Fig. 1.

To prepare the presentation of Nitsche weak formulation, we introduce two weights

$$\kappa_1=\frac{\beta_2}{\beta_1+\beta_2} \quad \text{and} \quad \kappa_2=\frac{\beta_1}{\beta_1+\beta_2}, \tag{2.4}$$

which satisfy that $\kappa_1+\kappa_2=1$. Then, we define the weighted averaging of a function w on the interface Γ as

$$\{\!\{w\}\!\}=\kappa_1 w_1|_\Gamma+\kappa_2 w_2|_\Gamma, \tag{2.5}$$

and also its dual weighted averaging as

$$\{\!\{w\}\!\}^*=\kappa_2 w_1|_\Gamma+\kappa_1 w_2|_\Gamma. \tag{2.6}$$

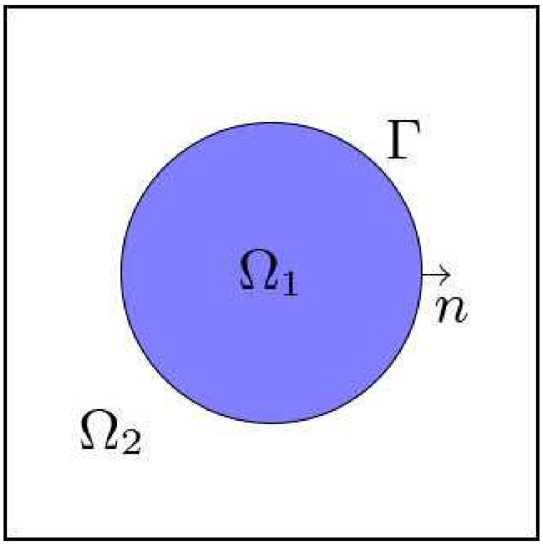


Figure 1: An illustrative example of a domain Ω with a circular interface Γ in two dimension. Here n stands for the outer normal direction of the inner domain Ω_1. Ω_2 is the outer domain.

Let $H^1(\Omega_1\cup\Omega_2)$ be the function space consisting of piecewise Sobolev functions w such that $w|_{\Omega_1}\in H^1(\Omega_1)$ and $w|_{\Omega_2}\in H^1(\Omega_2)$, whose norm is defined as

$$\|w\|_{1,\Omega_1\cup\Omega_2}=\left(\|w\|_{1,\Omega_1}^2+\|w\|_{1,\Omega_2}^2\right)^{1/2}, \tag{2.7}$$

where $\|\cdot\|_{1,\Omega_i}$ is the H^1-norm of a function in $H^1(\Omega_i)$. Similar notations are used for piecewise L^2 space and its corresponding norm. In addition, let $H_g^1(\Omega_1\cup\Omega_2)$ be the subset of $H^1(\Omega_1\cup\Omega_2)$ such that $u|_{\partial\Omega_2}=g$. In particular, $H_0^1(\Omega_1\cup\Omega_2)$ is the subspace of $H^1(\Omega_1\cup\Omega_2)$ with homogeneous Dirichlet boundary conditions.

The unfitted Nitsche weak formulation [2,11,17] of the interface problem (2.1a)-(2.1d) is to find $u\in H_g^1(\Omega_1\cup\Omega_2)$ such that

$$a(u,v)=\ell(v),\quad \forall v\in H_0^1(\Omega_1\cup\Omega_2), \tag{2.8}$$

where the bilinear form $a(\cdot,\cdot)$ is defined as

$$\begin{aligned} a(u,v)=&\sum_{i=1}^{2}(\beta\nabla u_i,\nabla v_i)_{\Omega_i}-\langle [\![u]\!],\{\{\beta\partial_n v\}\}\rangle_\Gamma \\ &-\langle [\![v]\!],\{\{\beta\partial_n u\}\}\rangle_\Gamma+\gamma_f\langle [\![u]\!],[\![v]\!]\rangle_\Gamma, \end{aligned} \tag{2.9}$$

and the linear functional $\ell(\cdot)$ is defined as

$$\ell(v)=\sum_{i=1}^{2}(f,v_i)_{\Omega_i}-\langle p,\{\{\beta\partial_n v\}\}\rangle_\Gamma+\gamma_f\langle p,[\![v]\!]\rangle_\Gamma+\langle q,\{\{v\}\}^*\rangle_\Gamma, \tag{2.10}$$

with $\gamma_f > 0$ being the stability parameter and $\langle\cdot,\cdot\rangle_\Gamma$ being the L^2-inner product on Γ.

To adopt the deep neural network, we reformulate the variational problem (2.8) as an energy minimization problem. To this end, we define the unfitted Nitsche energy functional L_n as

$$\begin{aligned} L_n(v) :=& \frac{1}{2}a(v,v) - \ell(v) \\ =& \frac{1}{2}\sum_{i=1}^{2}\int_{\Omega_i}\beta_i \nabla v_i \cdot \nabla v_i dx + \frac{\gamma_f}{2}\int_\Gamma (\llbracket v\rrbracket - p)^2 ds \\ &+ \int_\Gamma (p - \llbracket v\rrbracket)\{\{\beta\partial_n v\}\} ds - \int_\Omega f v dx \\ &- \int_\Gamma q\{\{v\}\}^* ds - \frac{\gamma_f}{2}\int_\Gamma p^2 ds. \end{aligned} \tag{2.11}$$

Then, we can show the variational problem is equivalent to the following energy minimization problem:

$$u = \arg\min_{v\in H^1_g(\Omega)} L_n(v). \tag{2.12}$$

In other words, Eqs. (2.1a)-(2.1d) is the Euler-Lagrangian equation of (2.12).

Remark 2.1. To simplify the presentation of this paper, we only consider inhomogeneous Dirichlet boundary conditions. For Neumann and Robin boundary conditions, they can be absorbed into the variational formulation. Other types of boundary conditions such as mixed Dirichlet and Neumann boundary conditions can be handled similarly.

Remark 2.2. We choose to impose the Dirichlet boundary condition strongly by building it into the solution space. Alternatively, we can impose the Dirichlet boundary condition weakly as in [26].

3 Deep unfitted Nitsche method

In this section, we shall take advantage of the universal approximation property of deep neural network and flexibility of the unfitted Nitsche weak form to develop the deep unfitted Nitsche method for solving elliptic interface problem (2.1a)-(2.1d). In the first subsection, we briefly introduce the deep neural networks used in this paper. In the second subsection, we describe the details of the deep unfitted Nitsche formulation.

3.1 Deep neural network

One of the critical ingredients for using deep learning to solve partial differential equations is to select a deep neural network as the ansatz function for trial functions. The commonly used deep neural networks to approximate the solutions to PDEs include feedforward neural network [10,28], and residual neural network (ResNet) [7,20]. Similar to the deep Ritz method in [7], we choose the ansatz function to be the ResNet.

For each integer i and some positive integer m, let $W^{[i_j]} \in \mathbb{R}^{m\times m}$ be the matrix of weights and $b^{[i_j]} \in \mathbb{R}^m$ be vector of biases for $j=1,2$. Then, the ith block of ResNet with m neurons can be written as

$$f_i(s) = \sigma(W^{[i_2]}\sigma(W^{[i_1]}s + b^{[i_1]}) + b^{[i_2]}) + s, \tag{3.1}$$

where σ is the activation function [10, 22]. To avoid the problem of vanishing gradient, smooth functions like sigmoid and hyperbolic tangent can be adopted.

Similarly, let $W^{[0]} \in \mathbb{R}^{m,d}$ (or $W^{[n+1]} \in \mathbb{R}^{1,m}$) be the matrix of weights in the first (or last) layer and $b^{[0]} \in \mathbb{R}^m$ (or $b^{[n+1]} \in \mathbb{R}^1$) be the vector of biases in the first (or last) layer. The ResNet with n blocks can be viewed as the composition of f_i's

$$u_\theta(x) = W^{[n+1]}\left(f_n \circ f_{n-1} \circ \cdots \circ f_1(W^{[0]}x + b^{[0]})\right) + b^{[n+1]}, \tag{3.2}$$

where θ denotes the set of parameters, *i.e.*,

$$\theta = \left\{W^{[0]}, b^{[0]}, W^{[1_1]}, b^{[1_1]}, W^{[1_2]}, b^{[1_2]}, \cdots, W^{[n_1]}, b^{[n_1]}, W^{[n_2]}, b^{[n_2]}, W^{[n+1]}, b^{[n+1]}\right\}.$$

In Fig. 2, we plot the architecture of the ResNet with four blocks.

3.2 Deep unfitted Nitsche formulation

The main advantage of the unfitted Nitsche method lies in the ability to use meshes independent of the location of the interface. This is possible by employing two different ansatz functions on the interface elements (elements cut through by the interface): one is for the interior domain, and the other one is for the exterior domain. Those two different ansatz functions are discontinuous across the interface Γ and patched together by Nitsche method. In the traditional unfitted Nitsche methods, the ansatz functions are piecewise polynomials. Following this line, we adopt two different deep neural network functions as the two ansatz functions to minimize the unfitted Nitsche energy functional (2.11).

Let u_{θ_i} be the ansatz function in Ω_i ($i=1,2$) and denote $u_\Theta(x) = (u_{\theta_1}(x), u_{\theta_2}(x))$. where $\Theta = (\theta_1, \theta_2)$. Before we proceed, we should make sure u_Θ satisfies the Dirichlet boundary condition (2.1b). For such purpose, we introduce an additional boundary penalty term as

$$L_b(u_\Theta) = \int_{\partial\Omega} |u_\Theta - g|^2 ds. \tag{3.3}$$

Ideally, we expect L_b close to zero. We define the extended unfitted Nitsche functional L as

$$L(u_\Theta) = L_n(u_\Theta) + \gamma_b L_b(u_\Theta), \tag{3.4}$$

where $\gamma_b > 0$ is the boundary penalty parameter.

Then, our deep unfitted Nitsche method is equivalent to the following optimization problem

$$\min_\Theta L(u_\Theta). \tag{3.5}$$

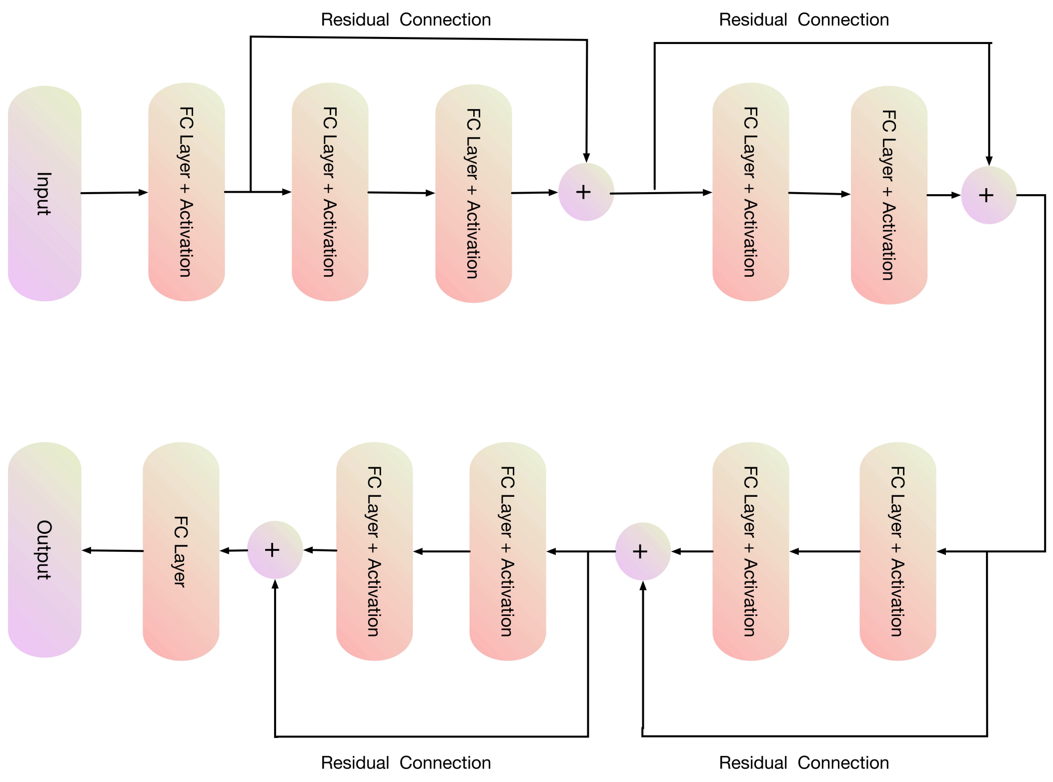


Figure 2: We present a diagram of ResNet with four blocks where FC layer denotes fully connected layer. Similar to the deep Ritz method in [7], we choose the ansatz function to be the ResNet.

Remark 3.1. In the literature, there are several alternative methods to impose the Dirichlet boundary conditions by building the Dirichlet boundary condition into the loss function [26] and training another deep neural network [32]. For the sake of simplicity, we choose to impose the Dirichlet boundary condition for deep neural network functions using the penalty method as in [7]. Interested readers are referred to [3] for a comparison study of different methods handling boundary conditions.

To solve the optimization problem using stochastic gradient descent type algorithms (*e.g.*, SGD [10] or ADAM [24]), we approximate the integrals by using the Monte-Carlo method, where the number of integral points is independent of the dimension of the underlying domain. Suppose $\{x_k^i\}_{k=1}^{N_i}$ are the uniformly sampled points in the domain Ω_i for $i=1,2$. Similarly, let $\{x_k^f\}_{k=1}^{N_f}$ and $\{x_k^b\}_{k=1}^{N_b}$ be the randomly sampled points on the interface Γ and the domain boundary $\partial\Omega$, respectively. The loss function $\hat{L}$ is defined as

$$\hat{L}(u_\Theta)=\frac{|\Omega_1|}{N_1}\sum_{k=1}^{N_1}\left(\frac{\beta_1}{2}\nabla u_{\theta_1}(x_k^1)\cdot\nabla u_{\theta_1}(x_k^1)-f(x_k^1)u_{\theta_1}(x_k^1)\right)$$

$$
\begin{aligned}
&+\frac{|\Omega_2|}{N_2}\sum_{k=1}^{N_2}\left(\frac{\beta_2}{2}\nabla u_{\theta_2}(x_k^2)\cdot\nabla u_{\theta_2}(x_k^2)-f(x_k^2)u_{\theta_2}(x_k^2)\right)\\
&+\frac{|\Gamma|}{N_f}\sum_{k=1}^{N_f}\left(\frac{\gamma_f}{2}([\![u_\Theta(x_k^f)]\!]-p(x_k^f))^2-\frac{\gamma_f}{2}p(x_k^f)^2\right)\\
&+\frac{|\Gamma|}{N_f}\sum_{k=1}^{N_f}\Big((p(x_k^f)-[\![u_\Theta(x_k^f)]\!])\{\!\{\beta\partial_n u_\Theta(x_k^f)\}\!\}\Big)\\
&-\frac{|\Gamma|}{N_f}\sum_{k=1}^{N_f}\Big(q(x_k^f)\{\!\{\beta\partial_n u_\Theta(x_k^f)\}\!\}^*\Big)\\
&+\frac{|\partial\Omega|}{N_b}\sum_{k=1}^{N_b}\gamma_b\Big(u_{\theta_2}(x_k^b)-g(x_k^b)\Big)^2,
\end{aligned}
\tag{3.6}
$$

where $|\Omega_i|$ is the measure of Ω_i in $\mathbb{R}^d$ $(i=1,2)$ and $|\Gamma|$ (or $|\partial\Omega|$) is the measure of Γ (or $\partial\Omega$) in $\mathbb{R}^{d-1}$. Then, the discrete counterpart of the optimization problem (3.5) reads as

$$\min_{\Theta}\hat{L}(u_\Theta). \tag{3.7}$$

The discrete optimization problem (3.7) actually involves the training of two deep neural network functions u_{θ_1} and u_{θ_2}. Those two neural networks can be trained independently using the same loss function $\hat{L}$. The gradient of deep neural network function can be efficiently calculated using automatic differentiation functionality [29] in the modern machine learning platform.

In the loss function(3.6), we need to compute the measure of each Ω_i. For problems with simple geometric shapes, we can compute them analytically. In general, we can use Monte-Carlo simulation like the hit-or-miss method to estimate the measure. Similarly, we can estimate the measure of Γ and $\partial\Omega$.

4 Numerical experiments

In this section, we present several numerical examples to illustrate the performance of the proposed deep unfitted Nitsche method. The proposed algorithm is implemented using the machine learning platform Pytorch [30]. In the following numerical experiments, we choose u_{θ_1} and u_{θ_2} have the same neural network architectures and select the activation function $\sigma=\tanh$. Both deep neural networks are initialized by Xavier initialization to prevent from exploding or vanishing and are trained independently using ADAM [24]. In all the following numerical experiments, we choose the learning rate $lr=0.001$ and generate new mini-batches every 10 epochs. To provide a qualitative description of training results, we use the following relative L^2-error

$$\text{Error}=\frac{\|u-u_\Theta\|_{0,\Omega_1\cup\Omega_2}}{\|u\|_{0,\Omega_1\cup\Omega_2}}. \tag{4.1}$$

The relative L^2-error is computed using Monte-Carlo methods with 10,000 uniformly sampled points. The relative L^2 errors are computed and recorded every 100 epochs. Similarly, we record the loss every 100 epochs.

4.1 Flower shape interface problem in 2D

In our first example, we consider the flower shape interface problem in the domain $\Omega = (-1,1)\times(-1,1)$ as in [11,14]. The flower shape interface Γ is given by the following polar coordinate

$$r = \frac{1}{2} + \frac{\sin(5\theta)}{7}. \tag{4.2}$$

The piecewise diffusion coefficient are $\beta_1 = 1$ and $\beta_2 = 10$. We choose the right hand side function to fit the exact solution

$$u(x) = \begin{cases} e^{(x_1^2+x_2^2)}, & \text{if } x = (x_1,x_2) \in \Omega_1, \\ 0.1(x_1^2+x_2^2)^2 - 0.01\ln(2\sqrt{x_1^2+x_2^2}), & \text{if } x = (x_1,x_2) \in \Omega_2. \end{cases}$$

The nonhomogeneous jump conditions (2.1d) and (2.1c) can be calculated from the above exact solution.

In this case, one can compute $|\Omega_1| = \frac{51}{192}\pi$ and $|\Omega_2| = 4 - \frac{51}{192}\pi$. To generate uniformly distributed random points in Ω_1 and Ω_2, we firstly generate uniformly distributed random points in the whole domain Ω. Then, we count the random points inside the interface Γ as the randomly sampled points in Ω_1 and the rest random points are in Ω_2. The randomly sampled points on Γ is produced by generating the uniformly sampled points on the interval $(0,2\pi)$ and then mapping them onto the interface Γ using (4.2).

In this numerical test, we choose ResNet with 3 blocks and $m = 10$ for each ansatz function, as illustrated in Fig. 2. Each ansatz function has 701 parameters. We uniformly sample 1024 points in the domain Ω. It turns out that there are 219 points inside the interface Γ. In other words, $N_1 = 219$ and $N_2 = 805$. In addition, we just choose $N_f = 256$ and $N_b = 128$. We choose $\gamma_b = 5000$ and $\gamma_f = 1000$. The corresponding decay of errors and loss functions during the training process are plotted in Fig. 4. We can see the error decays rapidly at the initial several thousand epochs and then fluctuate. After 50000 epochs, the relative L^2-error is reduced to 4.6%. In Fig. 3, we present a comparison between the deep unfitted Nitsche method (DUNM) solution and the exact solution. We can see the DUNM solution match well with the exact solution even though there is an inhomogeneous jump in the function values.

4.2 High-contrast interface problem in 2D

In this example, we consider the high contrast interface problem with homogeneous jump conditions in the domain $\Omega = (-1,1)\times(-1,1)$ as in [11, 14, 15]. The interface Γ

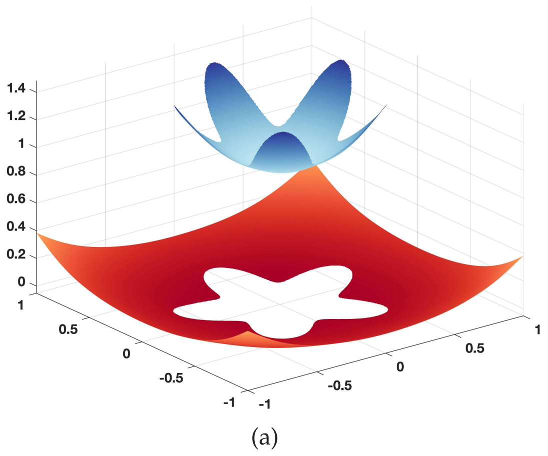


Figure 3: Comparison of solutions for flower shape interface problem: (a) Deep unfitted Nitsche solution; (b) Exact solution. Although there is an inhomogeneous jump of function values at the interface, both solutions match well.

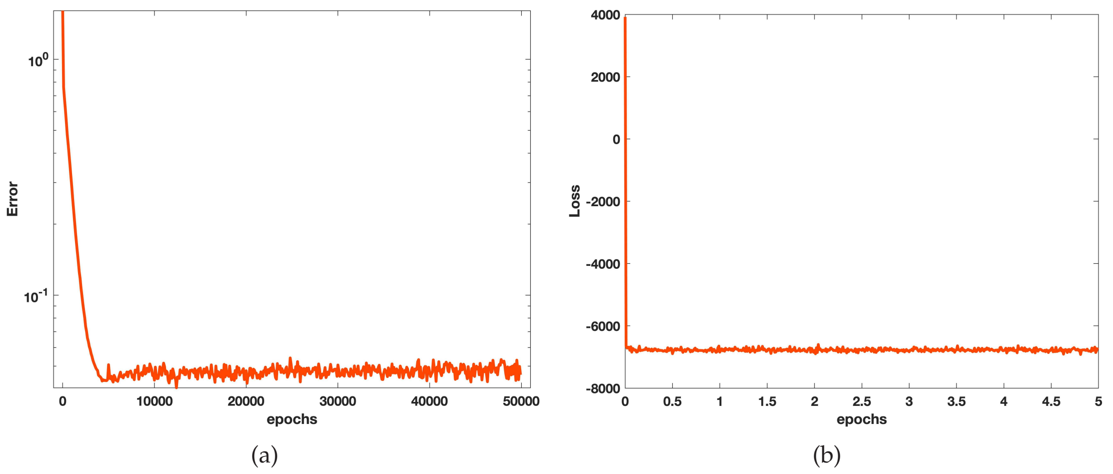


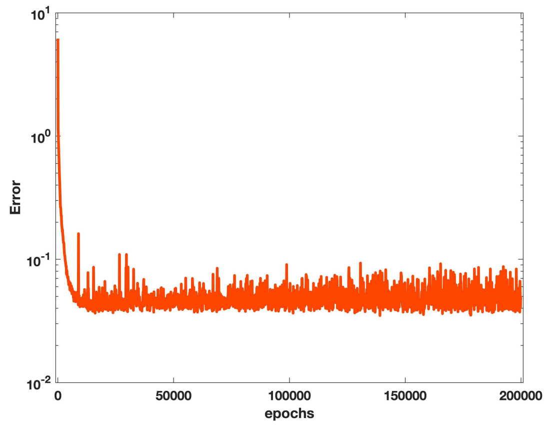
Figure 4: Training process of flower shape interface problem: (a) Decay of the relative L^2-error; (b) Decay of the loss.

is a circle with radius r_0 centered at the original. The exact solution is

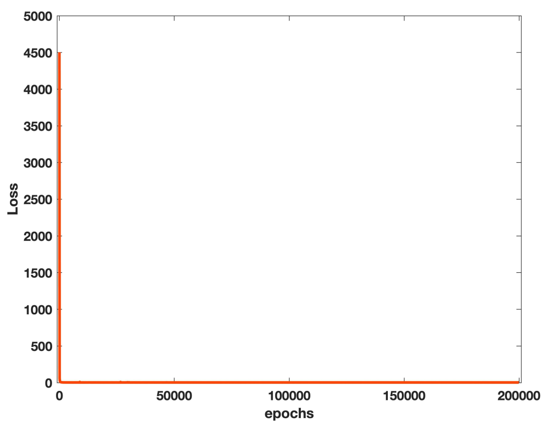
$$u(x)=\begin{cases} \frac{r^3}{\beta_1} & \text{if } x\in\Omega_1, \\ \frac{r^3}{\beta_2}+\left(\frac{1}{\beta_1}-\frac{1}{\beta_2}\right)r_0^3 & \text{if } x\in\Omega_2, \end{cases}$$

where $r=|x|$ is the Euclidean norm of x.

In this example, we consider the circle with radius $r_0=0.5$. We also use the ResNet with 3 blocks and 10 outputs in each block to represent each function. The uniformly

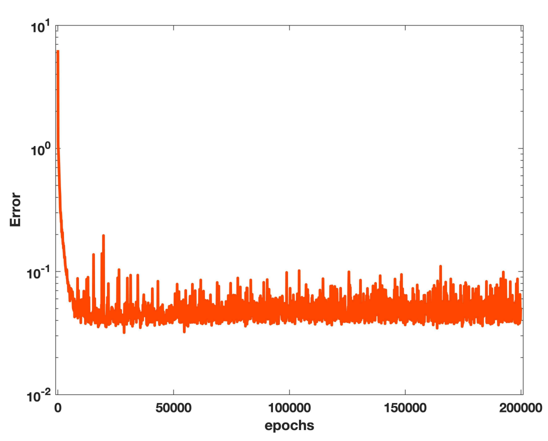


(a)

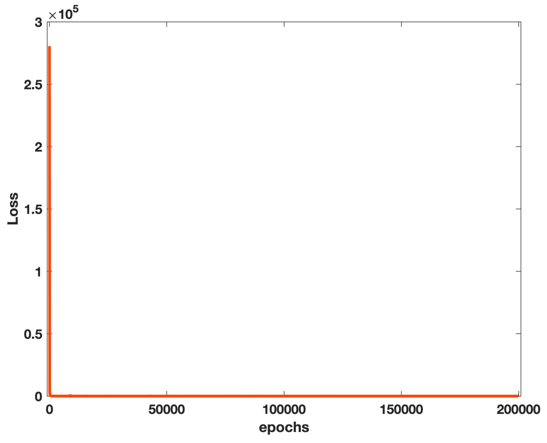


(b)

Figure 5: Training process of circular interface problem with $\beta_1 = 1$ and $\beta_2 = 1000$: (a) Decay of the relative L^2-error; (b) Decay of the loss.



(a)

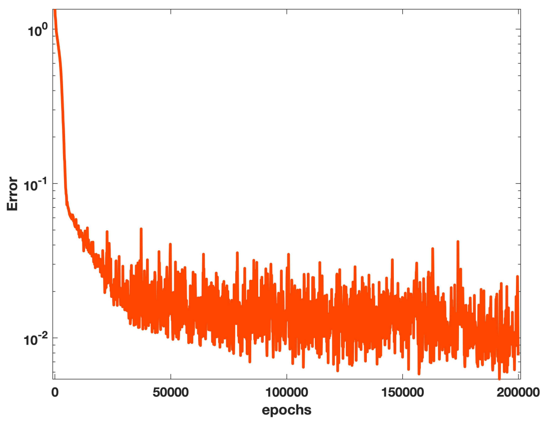


(b)

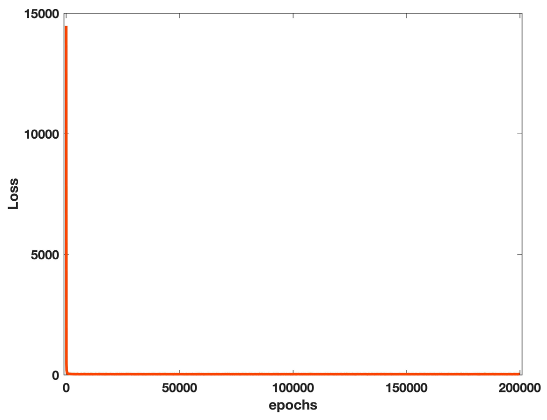
Figure 6: Training process of Training process of circular interface problem with $\beta_1 = 1$ and $\beta_2 = 100000$: (a) Decay of the relative L^2-error; (b) Decay of the loss.

randomly sampled points are generated by the same method as in previous example. Again, we generate 1024 randomly sampled points in Ω. In that case, $N_1 = 212$ points are inside Γ and $N_2 = 812$ points are outside Γ. Similarly, $N_f = 256$ and $N_b = 128$. The penalty parameters $\gamma_b = 5000$ and $\gamma_f = 1000$ are the exactly the same as previous case.

In the following numerical tests, we focus on the training of high contrast interface problem by considering the following four typical different jump ratios: $\beta_1/\beta_2 = 1000/1$ (large jump), $\beta_1/\beta_2 = 1/1000$ (large jump), $\beta_1/\beta_2 = 100000/1$ (huge jump), and $\beta_1/\beta_2 = 1/100000$ (huge jump). The training processes are described in Figs. 5-7 for each cases,



(a)



(b)

Figure 7: Training process of circular interface problem with $\beta_1=1000$ and $\beta_2=1$: (a) Decay of the relative L^2-error; (b) Decay of the loss.

Table 1: Relative L^2 errors for high contrast interface problems.

β_1/β_2	1000/1	1/1000	100000/1	1/100000
Error(%)	2.29	0.62	2.36	0.63

respectively. In these figures, we notice that the errors quickly decay to some specific errors and then oscillate around them. The relative L^2 errors after 200000 epochs are summarized in Table 1, where one can observe that the errors are not influenced by the jump ratios.

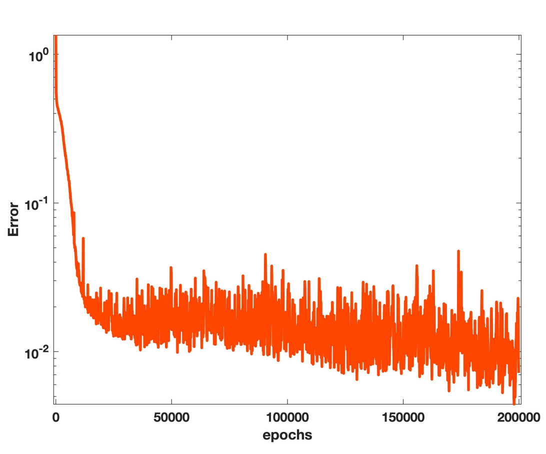
In Figs. 9 and 10, we plot the DUNM solution and the exact solution for the case $\beta_1/\beta_2=1000/1$ and $\beta_1/\beta_2=1/1000$, respectively. It is clear to see that the DUNM solutions match well with the corresponding exact solutions. We have also compared two other cases and observed the same phenomena.

4.3 High-dimensional interface problems

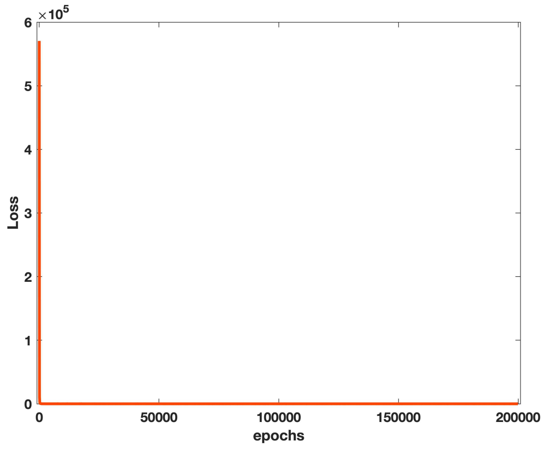
In this example, we consider the d-dimensional interface problem in the unit cube $\Omega=[-0.5,0.5]^d$. The unit cube is divided into two parts Ω_1 and Ω_2 by a d-dimensional sphere with radius $r_0=0.4$ centered at the origin. The diffusion coefficients are $\beta_1=1$ and $\beta_2=10$. The exact solution is

$$u(x)=\begin{cases} r^3 & \text{if } x\in\Omega_1, \\ \frac{r^3}{10}+0.0576 & \text{if } x\in\Omega_2, \end{cases}$$

where $r=|x|=\sqrt{\sum_{j=1}^d x_j^2}$. Therefore, we have homogeneous jump conditions. The right hand side function and boundary condition can be obtained from u.

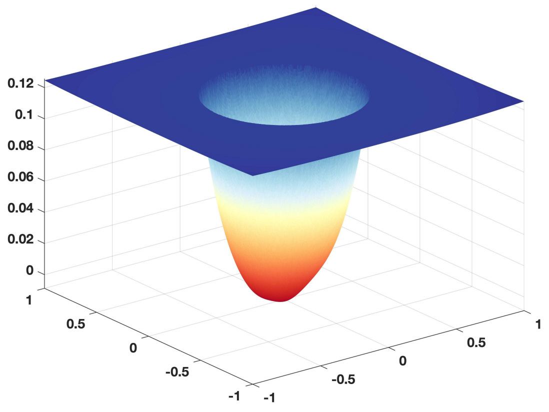


(a)



(b)

Figure 8: Training process of circle interface problem with $\beta_1 = 100000$ and $\beta_2 = 1$: (a) Decay of the relative L^2-error; (b) Decay of the loss.


(a)

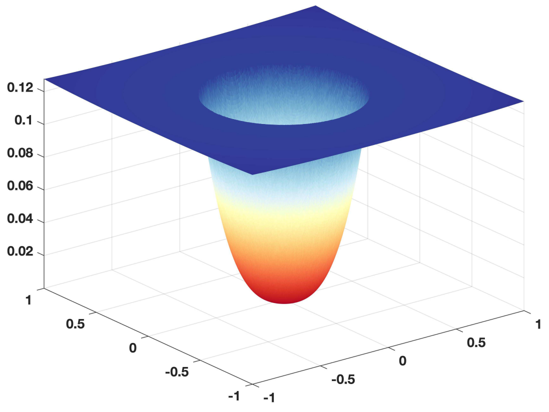

(b)

Figure 9: Comparison of solutions for circular interface problem with $\beta_1 = 1$ and $\beta_2 = 1000$: (a) Deep unfitted Nitsche solution; (b) Exact solution. The solutions match well for this high-contrast interface example.

In this case, the measure of $\Omega_1 = \frac{\pi^{d/2} r_0^d}{\Gamma(d/2+1)}$ and the measure of $\Omega_2 = 1 - \frac{\pi^{d/2} r_0^d}{\Gamma(d/2+1)}$. The measure of the interface is $\Gamma = \frac{2\pi^{d/2} r_0^{d-1}}{\Gamma(d/2)}$. We approximate each component of the solution by ResNet with 3 blocks and 20 neurons for each fully connected layer. In total, there are 2621 parameters for each deep neural network. To generate uniformly random points on d dimensional spheres, we use the `sample_hypersphere` function in BoTorch [1]. To guarantee there are enough randomly sampled points in Ω_1. We first generate 102 (about $1/10$ $N = N_1 + N_2$) uniformly random points inside the d-dimensional ball with r_0 using

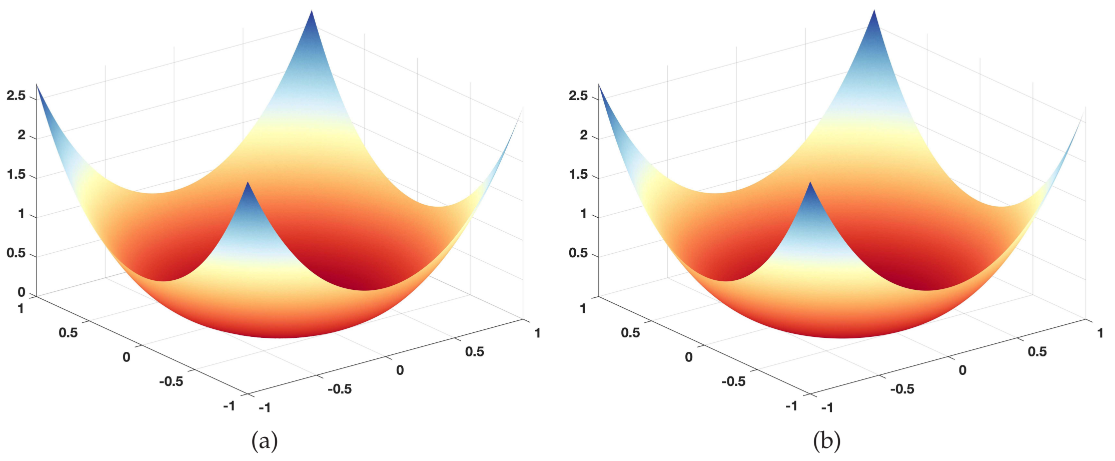


Figure 10: Comparison of solutions for circular interface problem with $\beta_1 = 10000$ and $\beta_2 = 1$: (a) Deep unfitted Nitsche solution; (b) Exact solution.

Table 2: Relative L^2 errors for high dimensional interface problems.

d	3	5	10	20
Error(%)	3.90	2.14	4.43	7.04

the dropped coordinates method [18]. In specific, we first generate 102 points on $d+2$ sphere with radius r_0 using the `sample_hypersphere` function in BoTorch [1] and drop the last two coordinates to get the uniform randomly sampled points. Then, we generate 912 random points in Ω which may also contain points in Ω_1. So we have $N_1 \geq 102$ and $N_2 = 1024 - N_1$. We take $N_f = 256$ and $N_b = 256d$. In the following test, we take $\gamma_b = 3000$ and $\gamma_f = 500$.

We consider four different high dimensional cases: $d = 3,5,10,20$. Fig. 11 displays the decay of the relative L^2 errors during the training process. Similar to the previous two examples, we can see that the errors decay quickly at the first few epochs and then fluctuate around some specific levels. In Table 2, we report the relative L^2 errors after 50000 epochs. We can see that we get almost the same level of relative L^2 errors independent of the dimensionality of the space even though we use the same number of points.

5 Conclusion

As a continued study of our previous works on elliptic interface problems [11,14,15], we propose a deep unfitted Nitsche method to address the high-dimensional challenge with high-contrasts. The challenge is also known as the curse of dimensionality. The classical numerical methods such as finite difference and finite element methods require unaffordable computational time. The proposed method deploys the deep neural network to

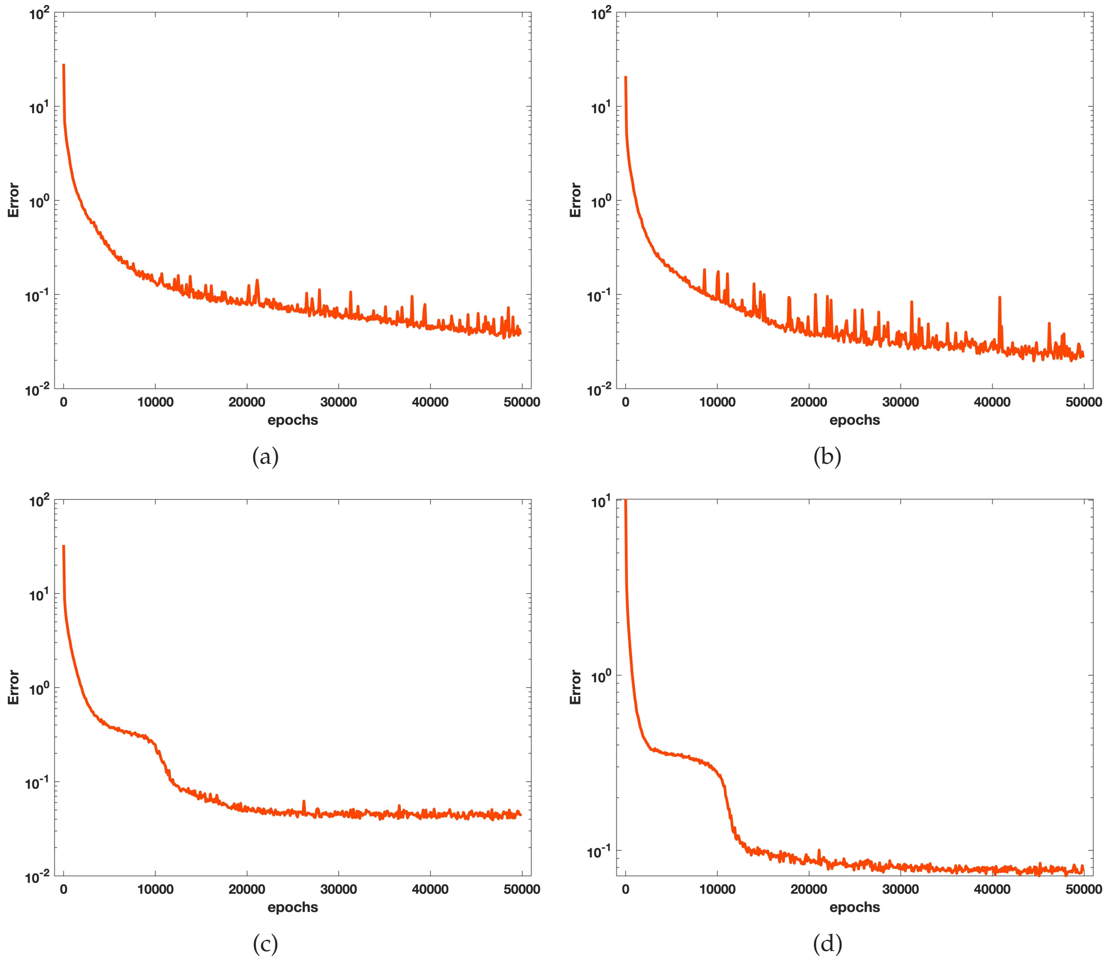


Figure 11: Decay of loss functions in high-dimensional interface problems: (a) $d=3$; (b) $d=5$; (c) $d=10$; (d) $d=20$.

solve the equivalent high-dimensional optimization problem. To address the high contrasts of the solution, we introduced a so-called unfitted Nitsche energy functional, which utilizes different deep neural networks to represent different components of the solution in the high dimensional case. Different deep networks are patched together weakly by Nitsche method and can be trained independently using the unfitted Nitsche functional. The unfitted Nitsche energy functional is approximated by the Monte-Carlo methods. An additional penalty term is added to the discrete energy functional to handle the Dirichlet boundary conditions. The proposed method is easy to be implemented and mesh-free, which is illustrated by several numerical examples including high contrasts and high dimensional cases.

Acknowledgments

H.G. is grateful to Prof. Zuoqiang Shi from Tsinghua University and Dr. Guozhi Dong from the Humboldt University of Berlin for stimulating and helpful discussions. H.G. was partially supported by Andrew Sisson Fund of the University of Melbourne. X.Y. was partially supported by the NSF grants DMS-1818592 and DMS-2109116.

References

[1] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, *BoTorch: A framework for efficient Monte-Carlo Bayesian optimization*, Advances in Neural Information Processing Systems 33, 2020.

[2] E. Burman, S. Claus, P. Hansbo, M. G. Larson, and A. Massing, *CutFEM: discretizing geometry and partial differential equations*, Internat. J. Numer. Methods Engrg. **104** (2015), no. 7, 472–501. MR 3416285

[3] J. Chen, R. Du, and 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), no. 3, 354–376. MR 4199960

[4] Gonalo dos Reis, Stefan Engelhardt, and Greig Smith, *Simulation of McKean-Vlasov SDEs with super linear growth*, IMA Journal of Numerical Analysis (2021).

[5] C. Duan, Y. Jiao, Y. Lai, X. Lu, and Z. Yang, *Convergence rate analysis for deep Ritz method*, arXiv preprint arXiv:2103.13330 (2021).

[6] R. M. Dudley, *The speed of mean Glivenko-Cantelli convergence*, The Annals of Mathematical Statistics **40** (1969), no. 1, 40–50.

[7] W. E and B. Yu, *The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, Commun. Math. Stat. **6** (2018), no. 1, 1–12. MR 3767958

[8] W. E, J. 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), no. 4, 349–380. MR 3736669

[9] N. Fournier and A. Guillin, *On the rate of convergence in Wasserstein distance of the empirical measure*, Probability Theory and Related Fields **162** (2015), no. 3, 707–738.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[11] H. Guo and X. Yang, *Gradient recovery for elliptic interface problem: III. Nitsche's method*, J. Comput. Phys. **356** (2018), 46–63. MR 3743629

[12] H. Guo, X. Yang, and Y. Zhu, *Unfitted Nitsche's method for computing band structures of phononic crystals with periodic inclusions*, Comput. Methods Appl. Mech. Engrg. **380** (2021), 113743, 17. MR 4236361

[13] ———, *Unfitted Nitsche's method for computing wave modes in topological materials*, J. Sci. Comput. **88** (2021), no. 1, 24, 1. MR 4270808

[14] H. Guo and X. Yang, *Gradient recovery for elliptic interface problem: II. Immersed finite element methods*, Journal of Computational Physics **338** (2017), 606–619.

[15] ———, *Gradient recovery for elliptic interface problem: I. Body-fitted mesh*, Commun. Comput. Phys. **23** (2018), no. 5, 1488–1511.

[16] J. Han, A. Jentzen, and W. E, *Solving high-dimensional partial differential equations using deep learning*, Proc. Natl. Acad. Sci. USA **115** (2018), no. 34, 8505–8510. MR 3847747

[17] A. Hansbo and P. Hansbo, *An unfitted finite element method, based on Nitsche's method, for elliptic interface problems*, Comput. Methods Appl. Mech. Engrg. **191** (2002), no. 47-48, 5537–5552. MR 1941489

[18] R. Harman and V. Lacko, *On decompositional algorithms for uniform sampling from n-spheres and n-balls*, J. Multivariate Anal. **101** (2010), no. 10, 2297–2304. MR 2719863

[19] C. He, X. Hu, and L. Mu, *A mesh-free method using piecewise deep neural network for elliptic interface problems*, arXiv preprint arXiv:2005.04847 (2020).

[20] K He, S. Zhang, X.and Ren, and J. Sun, *Deep residual learning for image recognition (2015)*, arXiv preprint arXiv:1512.03385 (2016).

[21] X. He, Song F., and W. Deng, *A well-conditioned, nonconforming Nitsche's extended finite element method for elliptic interface problems*, Numer. Math. Theory Methods Appl. **13** (2020), no. 1, 99–130. MR 4044417

[22] C. F. Higham and D. J. Higham, *Deep learning: An introduction for applied mathematicians*, SIAM Rev. **61** (2019), no. 4, 860–891. MR 4027841

[23] W.-F. Hu, T.-S. Lin, and M.-C. Lai, *A discontinuity capturing shallow neural network for elliptic interface problems*, arXiv preprint arXiv:2106.05587 (2021).

[24] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, Proc. of the 3rd International Conference for Learning Representations (ICLR), 2015.

[25] Z. Li and K. Ito, *The immersed interface method*, Frontiers in Applied Mathematics, vol. 33, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006, Numerical Solutions of PDEs Involving Interfaces and Irregular Domains. MR 2242805

[26] Y. Liao and P. Ming, *Deep Nitsche method: Deep Ritz method with essential boundary conditions*, Commun. Comput. Phys. **29** (2021), no. 5, 1365–1384. MR 4230976

[27] J. Lu, Y., and M. Wang, *A priori generalization analysis of the deep Ritz method for solving high dimensional elliptic equations*, arXiv preprint arXiv:2101.01708 (2021).

[28] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, *DeepXDE: A deep learning library for solving differential equations*, SIAM Rev. **63** (2021), no. 1, 208–228. MR 4209661

[29] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in PyTorch*, NIPS 2017 Workshop on Autodiff, 2017.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *PyTorch: An imperative style, high-performance deep learning library*, Advances in Neural Information Processing Systems 32 (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), Curran Associates, Inc., 2019, pp. 8024–8035.

[31] 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** (2019), 686–707.

[32] H. Sheng and C. Yang, *PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries*, J. Comput. Phys. **428** (2021), 110085, 13. MR 4196555

[33] J. Sirignano and K. Spiliopoulos, *DGM: a deep learning algorithm for solving partial differential equations*, J. Comput. Phys. **375** (2018), 1339–1364. MR 3874585

[34] C. Wang and P. Sun, *An augmented Lagrangian Uzawa iterative method for solving double saddle-point systems with semidefinite* (2,2) *block and its application to DLM/FD method for elliptic interface problems*, Commun. Comput. Phys. **30** (2021), no. 1, 124–143. MR 4258343

[35] H. Wang, J. Chen, P. Sun, and R. Lan, *An interface-unfitted conforming enriched finite element method for Stokes-elliptic interface problems with jump coefficients*, Commun. Comput. Phys. **27** (2020), no. 4, 1174–1200. MR 4161045

[36] Z. Wang and Z. Zhang, *A mesh-free method for interface problems using the deep learning approach*, J. Comput. Phys. **400** (2020), 108963, 16. MR 4019794

[37] Jonathan Weed and Francis Bach, *Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance*, Bernoulli **25** (2019), no. 4A, 2620–2648.

[38] Y. Zang, G. Bao, X. Ye, and H. Zhou, *Weak adversarial networks for high-dimensional partial differential equations*, J. Comput. Phys. **411** (2020), 109409, 14. MR 4079373

[39] C. Zhang, Z. Li, and X. Yue, *An acceleration technique for the augmented IIM for 3D elliptic interface problems*, Numer. Math. Theory Methods Appl. **14** (2021), no. 3, 773–796. MR 4275238