Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.OA-2020-0062

Published online 9 October 2020

Quasi-Monte Carlo Sampling for Solving Partial Differential Equations by Deep Neural Networks

Jingrun Chen[1], Rui Du[1,*], Panchi Li[2] and Liyao Lyu[3]

[1] *School of Mathematical Sciences and Mathematical Center for Interdisciplinary Research, Soochow University, Suzhou 215006, China*
[2] *School of Mathematical Sciences, Soochow University, Suzhou 215006, China*
[3] *School of Mathematical Sciences and CW Chu College, Soochow University, Suzhou 215006, China*

Received 14 April 2020; Accepted (in revised version) 7 September 2020

Abstract. Solving partial differential equations in high dimensions by deep neural networks has brought significant attentions in recent years. In many scenarios, the loss function is defined as an integral over a high-dimensional domain. Monte-Carlo method, together with a deep neural network, is used to overcome the curse of dimensionality, while classical methods fail. Often, a neural network outperforms classical numerical methods in terms of both accuracy and efficiency. In this paper, we propose to use quasi-Monte Carlo sampling, instead of Monte-Carlo method to approximate the loss function. To demonstrate the idea, we conduct numerical experiments in the framework of deep Ritz method. For the same accuracy requirement, it is observed that quasi-Monte Carlo sampling reduces the size of training data set by more than two orders of magnitude compared to that of Monte-Carlo method. Under some assumptions, we can prove that quasi-Monte Carlo sampling together with the deep neural network generates a convergent series with rate proportional to the approximation accuracy of quasi-Monte Carlo method for numerical integration. Numerically the fitted convergence rate is a bit smaller, but the proposed approach always outperforms Monte Carlo method.

AMS subject classifications: 11K50, 35J20, 65N99

Key words: Quasi-Monte Carlo sampling, deep Ritz method, loss function, convergence analysis.

1. Introduction

Deep neural networks (DNNs) have had great success in text classification, computer vision, natural language processing and other data-driven applications [13, 16,

*Corresponding author. *Email addresses:* jingrunchen@suda.edu.cn (J. Chen), durui@suda.edu.cn (R. Du), LiPanchi1994@163.com (P. Li), lylv@stu.suda.edu.cn (L. Lyu)

21, 30, 35]. Recently, DNNs have been applied to the field of numerical analysis and scientific computing, with the emphasis on solving high-dimensional partial differential equations (PDEs) [11,12,17,32], which are widely used in physics and finance. Notable examples include Schrödinger equation in the quantum many-body problem [15, 18], Hamilton-Jacobi-Bellman equation in stochastic optimal control [10,17], and nonlinear Black-Scholes equation for pricing financial derivatives [1, 7].

Classical numerical methods, such as finite difference method [23] and finite element method [3], share the similarity that the approximation stencil has compact support, resulting in the sparsity of stiffness matrix (or Hessian in the nonlinear case). Advantages of these methods are obvious for low dimensional PDEs (the dimension $K \leq 3$). However, the number of unknowns grows exponentially as K increases and classical methods run into the curse of dimensionality. In another line, spectral method [31] uses basis functions without compact support and thus sacrifices the sparsity, but often has the exponential accuracy. However, the number of modes used in the spectral method also grows exponentially as K increases. Sparse grid method [5, 8, 14] mitigates the aforementioned situation to some extent ($K \leq 9$ typically). Therefore, high-dimensional PDEs are far out of the capability of classical methods.

The popularity of DNNs in scientific computing results from its ability to approximate a high-dimensional function without the curse of dimensionality. To illustrate this, we focus on methods in which the loss function is defined as an integral over a bounded domain in high dimensions; see the deep Ritz method [11] and the deep Galerkin method [32] for examples. The success of DNNs relies on composition of functions without compact support and sampling strategy for approximating the high-dimensional integral. It is known that the choice of approximate functions in DNNs is of particular importance. For example, in the current work, the approximate function in one block of DNN consists of two linear transformations, two nonlinear activation functions, and one shortcut connection. Besides, since the network architecture is chosen a priori, the number of parameters can be independent of K or only grows linearly as K increases. On the other hand, only a fixed number of samples (or at most linear growth) is used to approximate the high-dimensional integral. Altogether, DNNs can overcome the curse of dimensionality when solving high-dimensional PDEs. In [2], the above step of numerical quadrature is viewed as approximating the expected risk by its empirical risk using Monte Carlo (MC) method. Consequently, the full gradient of the loss function is approximated by a finite number of samples and the stochastic gradient descent (SGD) method is used to find the optimal set of parameters in the network. It is shown that such a procedure converges under some assumptions.

From the perspective of numerical analysis, using N i.i.d. random points, MC method approximates an integral with $\mathcal{O}(N^{-\frac{1}{2}})$ error [24]. It is also known that using N carefully chosen (deterministic) points, quasi-Monte Carlo (QMC) method approximates an integral with $\mathcal{O}(\frac{(\log N)^K}{N^{-1}})$ error and the logarithmic factor can be removed under some assumptions [9, 27, 28, 34]. Therefore, it is natural to replace MC method by QMC method in the community of machine learning. One example is the usage of QMC method in variational inference and QMC method has been proved to perform better

than MC method [4]. Another example is the usage of QMC sampling in the stage of data generation for training DNNs; see an application in organic semiconductors [25]. In this work, we consider another application of QMC method, i.e., approximation of the high-dimensional integral when solving PDEs by DNNs.

In order to demonstrate the advantages of QMC method, we take deep Ritz method [11] as an example. Results obtained here shall be applicable to other methods, like deep Galerkin method [32], where a high-dimensional integral is defined as the loss function. Briefly speaking, deep Ritz method solves a variational problem for a high-dimensional PDE using a deep neural network with residual connection. Data are drawn randomly over the high-dimensional domain to train the parameters of the neural network. All numerical observations in the current work are based on deep Ritz method. In deep Galerkin method, the loss function contains not only the volume integral over the high-dimensional domain but also penalty terms for boundary conditions and initial conditions. We also demonstrate the advantage of QMC method in deep Ritz method when the penalty term is present. Theoretically, under certain assumptions, we prove a convergence result of the SGD method with respect to both the iteration number and the size of training data set.

The paper is organized as follows. We first introduce deep Ritz method for PDEs and QMC method in Section 2. Numerical results of QMC sampling and MC sampling are shown in Section 3 with convergence analysis given in Section 4. Conclusions are drawn in Section 5.

2. Quasi-Monte Carlo sampling for deep Ritz method

For completeness, we first introduce deep Ritz method. The basic idea is to solve a variational problem associated to a PDE using DNNs. The training data points are chosen randomly over the given domain using MC method. SGD method is then used to find an optimal solution. In the current work, QMC method is employed to replace MC method and the other components are remained almost the same. For consistency, we use superscripts for indices of sampling points and subscripts for coordinates of a vector throughout the paper.

2.1. Loss function

We take the variational problem associated to the Poisson equation [22] as an example

$$\min_{u\in H} I[u], \tag{2.1}$$

where the loss function (objective function) $I[u]$ reads as

$$I[u] = \int_\Omega \left(\frac{1}{2} |\nabla u(x)|^2 - f(x)u(x) \right) \mathrm{d}x, \tag{2.2}$$

and the set of trial functions H is of infinite dimension. Here f is a given function, representing the external force to the system and Ω is a bounded domain in $\mathbb{R}^K$.

When the solution of a PDE is approximated by a neural network $u(x) \approx \hat{u}_\theta(x)$, i.e., H is restricted to a highly nonlinear manifold, our goal is to find the optimal set of parameters in the neural network, denoted by θ, such that

$$I(\theta) = \int_\Omega \left(\frac{1}{2} \left| \nabla_x \hat{u}_\theta(x) \right|^2 - f(x)\hat{u}_\theta(x) \right) \mathrm{d}x \tag{2.3}$$

is minimized. Numerically, a quadrature scheme is needed and the above objective function is approximated by

$$I(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2} \left| \nabla_x \hat{u}_\theta(x^i) \right|^2 - f(x^i)\hat{u}_\theta(x^i) \right) \tag{2.4}$$

with N sampling points $\{x^i\}_{i=1}^N$ which will be specified in Section 2.3.

2.2. Trial function and network architecture

The neural network we use here is stacked by several blocks with each containing two linear transformations, two activation functions and one shortcut connection. The i-th block can be formulated as

$$t = f_i(s) = \sigma\big(\theta_{i,2} \cdot \sigma(\theta_{i,1} \cdot s + b_{i,1}) + b_{i,2}\big) + s. \tag{2.5}$$

Here $s \in R^m$ is the input with the dimension $m, t \in R^m$ is the output, weights $\theta_{i,1}, \theta_{i,2} \in R^{m\times m}, b_{i,1}, b_{i,2} \in R^m$ and σ is the activation function. Fig. 1 demonstrates one block of the network.

To balance simplicity and accuracy, we use the following swish function

$$\sigma(x) = \frac{x}{1 + \exp(-x)} \tag{2.6}$$

as the activation function [29], which is different from the one used in [11].

The last term on the right-hand side of (2.5) is called the shortcut connection or residual connection. Benefits of using it are [19]:

1) It can automatically solve the notorious problem of vanishing/exploding gradient.

2) Without adding any parameters or computational complexity, the shortcut connection performing as an identity mapping can resolve the degradation issue. That is, with the network depth increasing, accuracy gets saturated and then degrades rapidly.

With these components, the fully connected n-layer network can be expressed as

$$f_{\theta'}(x) = f_n \circ f_{n-1} \circ \cdots \circ f_1(x), \tag{2.7}$$

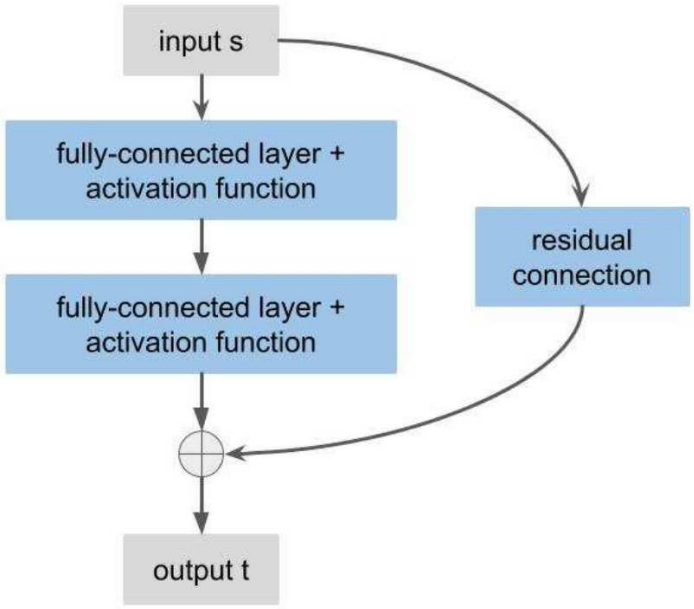


Figure 1: One block of the network. Typically a deep neural network contains a sequence of blocks, each of which consists of two fully-connected layers and one shortcut connection.

where θ' denotes the set of parameters in the network. Since the input x in the first block is in $\mathbb{R}^K$, not in $\mathbb{R}^m$, we need to apply a linear transformation on x before putting it into the network structure. Having $f_{\theta'}(x)$, we obtain $\hat{u}_\theta(x)$ by

$$\hat{u}_\theta(x) = a \cdot f_{\theta'}(x) + b, \tag{2.8}$$

where $\theta = \{\theta', a, b\}$. Note that the parameters a and b in (2.8) also need to be trained.

To make (2.1)-(2.2) have a unique solution, a boundary condition has to be imposed. Consider the inhomogeneous Dirichlet boundary condition for example

$$u(x) = g(x), \quad x \in \partial\Omega. \tag{2.9}$$

One way to implement it is to select trial functions that satisfy the boundary condition and thus have to be problem-dependent. To avoid this, we build trial functions of the form

$$\hat{u}_\theta(x) = A(x) \cdot \big(a \cdot f_{\theta'}(x) + b\big) + B(x), \quad x \in \Omega, \tag{2.10}$$

where by choice $A(x) = 0$ and $B(x) = g(x)$ when $x \in \partial\Omega$. Therefore, $\hat{u}_\theta(x)$ satisfies the boundary condition automatically. For Neumann boundary condition, however, we have to add a penalty term into the loss function; see (3.7) in Section 3.2 for example.

2.3. Sampling strategies

The loss function is defined over a high-dimensional domain, thus only a fixed size of points (mini-batch $\{x^i\}_{i=1}^N$) is allowed to approximate the integral. Due to the curse

of dimensionality, standard quadrature rules may run into the risk that the integrand is minimized on fixed points but the functional itself is far away from being minimized [11]. Therefore, points are chosen randomly and the approximation accuracy is of $\mathcal{O}(N^{-\frac{1}{2}})$ [24]. For stochastic problems, from the perspective of sampling strategies, it is well known that QMC method performs much better with the same size of sampling points [6, 9, 27]. We briefly review both methods here.

Consider $\Omega = [0,1]^K$ ($K \gg 1$) for convenience and let u be an integrable function in Ω

$$I(u) = \int_\Omega u(x)\mathrm{d}x < \infty, \tag{2.11}$$

which is approximated by N points of the form

$$Q_N(u) = \frac{1}{N}\sum_{i=1}^{N} u(x^i). \tag{2.12}$$

Let $P_N = \{x^i\}_{i=1}^N \subset \Omega$ be the prescribed sampling points. In MC method, these points are chosen randomly and independently from the uniform distribution in Ω. There exists a probabilistic error (root mean square error) estimate for MC method

$$\sqrt{\mathbb{E}\left[|I(u) - Q_N(u)|^2\right]} = \frac{\sigma(u)}{\sqrt{N}},$$

where $\sigma^2(u)$ is the variance of u of the form

$$\sigma^2(u) = I\big(u^2\big) - \big(I(u)\big)^2.$$

It is easy to check that MC method is unbiased, i.e., $\mathbb{E}[Q_N(u)] = I(u)$. The variance of MC method is

$$\begin{aligned}
\mathrm{Var}\big(Q_N(u)\big) &= \mathbb{E}\left[|I(u) - Q_N(u)|^2\right] \\
&= \frac{1}{N(N-1)}\sum_{i=1}^{N}\big(u(x^i) - Q_N(u)\big)^2 \\
&= \frac{1}{N(N-1)}\left(\sum_{i=1}^{N} u^2(x^i) - N[Q_N(u)]^2\right).
\end{aligned}$$

In QMC method, however, sampling points are chosen in a deterministic way to approximate the integral with the best approximation accuracy; see for example [9, 27, 28, 34]. The deterministic feature of QMC method leads to a guaranteed error bound and faster convergence rate for smooth integral functions. More explicitly, an upper bound of the deterministic error, known as Koksma-Hlawka error bound [27], is

$$|Q_N(u) - I(u)| \le D(P_N)V(u). \tag{2.13}$$

Here the variation $V(u)$ is defined as

$$V(u) = \sum_{k=1}^{K} \sum_{1 \le i_1 < \cdots < i_k \le K} V_{Vit}^{(k)}(u; i_1, \ldots, i_k),$$

where $V_{Vit}^{(k)}(u; i_1, \ldots, i_k)$ is the variation in the sense of Vitali applied to the restriction of u to the space of dimension $k\{(u_1, \ldots, u_K) \in \Omega : u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$. Precisely, let $\Delta(u, J)$ be the alternative sum of u values at the edges of sub-interval J when P_N is a partition of $\Omega = [0,1]^K$, we give the definition of the variation in the sense of Vitali as

$$V_{V_{it}}(u) = \sup_{P_N} \sum_{J \in P_N} |\Delta(u, J)| .$$

$D(P_N)$ is defined to measure the discrepancy of the set P_N as

$$D(P_N) = \sup_{x \in [0,1]^K} \left| \frac{1}{N} \sum_{n=1}^{N} 1_B(x^n) - \prod_{i=1}^{K} x_i \right|,$$

where $x = (x_1, \ldots, x_K)$. Note that the error bound in (2.13) is controlled by this discrepancy and $\lim_{N \to +\infty} D(P_N) = 0$ if and only if the sequence P_N is equi-distributed. A sequence P_N is said to be a low-discrepancy sequence if $D(P_N) = \mathcal{O}(\frac{(\ln N)^K}{N})$, which is the best-known result for infinite sequences. Therefore, QMC method converges much faster than MC method. Practically, the commonly used Sobol sequence is one of the low-discrepancy sequences [33, 34].

2.4. Stochastic gradient descent method

In deep Ritz method, we use the SGD method to find the optimal set of parameters. The SGD method finds the optimal solution in an iterative way with the i-th iteration of the form

$$\theta_{i+1} = \theta_i + \alpha_i g(\theta_i, \xi_i), \tag{2.14}$$

where α_i is the stepsize and $g(\theta_i, \xi_i)$ is a stochastic vector defined as

$$g(\theta_i, \xi_i) = \frac{1}{N} \sum_{k=1}^{N} \nabla I\big(\theta_i, \xi_i^k\big). \tag{2.15}$$

At the i-th iteration, ξ_i is a random variable when interpreting the spatial variable x in the loss function (2.2) or (2.3) as an expectation, ξ_i^k is the k-th realization of ξ_i generated by a sampling strategy such as QMC or MC method, $\nabla I(\theta_i, \xi_i^k)$ represents the gradient of the loss function evaluated at θ_i and ξ_i^k. Practically, we use ADAM [20] to accelerate the training process for both MC and QMC methods. It is worth to mention that without taking the derivative with respect to θ, (2.15) is equivalent to (2.4) by interpreting the high-dimensional integral (2.2) or (2.3) as an expectation of a random variable ξ, which is in an infinite-dimensional space and ξ_i^ks are the finite dimensional realizations.

3. Numerical results

Now we are ready to apply QMC sampling strategy to train the network structure of deep Ritz method in Section 2. For Dirichlet problems and some special Neumann problems, we do not need penalty terms on the boundary. However, for general Neumann problems, a penalty term must be added to the loss function and thus we have to approximate this term by sampling on the boundary. No matter in which case, numerically QMC method always performs better than MC method. We use the relative L_2 error for quantitative comparison in all examples

$$\text{error} = \left(\int_\Omega \left(\hat{u}_\theta(x) - u(x)\right)^2\right)^{\frac{1}{2}} \mathrm{d}x \left(\int_\Omega u(x)^2 \mathrm{d}x\right)^{-\frac{1}{2}}. \tag{3.1}$$

3.1. A linear Dirichlet boundary condition problem

Consider the Poisson equation

$$\begin{cases} -\triangle u = \pi^2 \sum_{k=1}^{K} \cos(\pi x_k), & x \in \Omega, \\ u(x) = \sum_{k=1}^{K} \cos(\pi x_k), & x \in \partial\Omega. \end{cases} \tag{3.2}$$

over $\Omega = [-1, 1]^K$.

The exact solution is $u(x) = \sum_{k=1}^{K} \cos(\pi x_k)$. We construct the network in the form of

$$\begin{cases} \hat{u}_\theta(x) = A(x) \cdot f_\theta(x) + B(x), & x \in \Omega, \\ A(x) = \exp\left(\prod_{k=1}^{K} \left(x_k^2 - 1\right)\right) - 1, & x \in \Omega, \\ B(x) = \exp\left(\prod_{k=1}^{K} \left(x_k^2 - 1\right)\right) \sum_{k=1}^{K} \cos(\pi x_k), & x \in \Omega. \end{cases} \tag{3.3}$$

It is easy to verify that $A(x) = 0, B(x) = \sum_{k=1}^{K} \cos(\pi x_k)$ on the boundary, satisfying the structure defined in (2.10). Fig. 2 plots exact and trained solutions to (3.2) in 2D and qualitative agreement is observed.

Detailed setup of the neural network used for Dirichlet problem in different dimensions is recorded in Table 1.

Relative L_2 errors in different dimensions and the corresponding convergence rates with respect to the mini-batch size are shown in Tables 2 and 3, respectively. Since there are some oscillations as the iteration increases, each point here represents the error averaged over 50 iterations. The total number of iterations is set to be 10000. It is reasonable to find that QMC method performs better than MC method as shown in

Table 1: Detailed setup of the neural network used for Dirichlet problem in different dimensions, where m denotes the number of nodes contained in each layer.

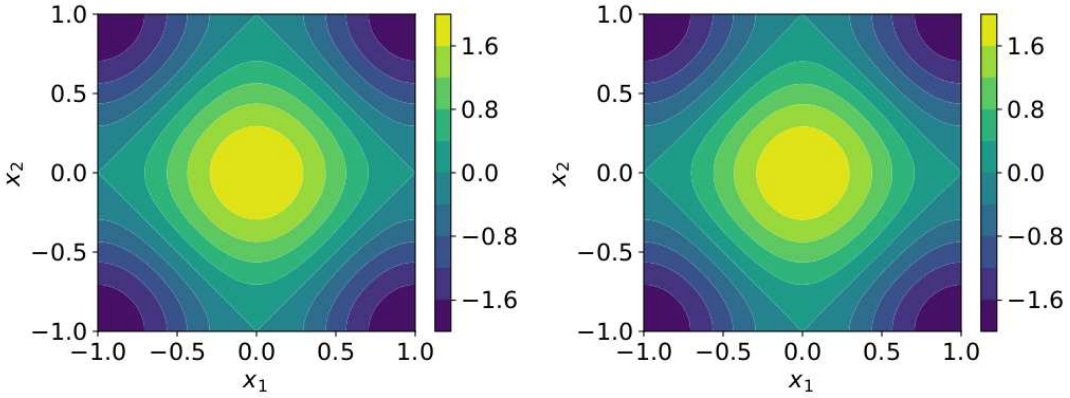
Dimension	Blocks Num	m	Parameters
2	3	8	465
4	4	16	2274
8	4	20	3561
16	4	48	19681

Table 2: Relative L^2 errors in different dimensions with different mini-batch sizes for Dirichlet problem and the convergence order is recorded with respect to $\frac{1}{N}$.

Dimension	Mini-batch size	QMC		MC	
		error($\times 10^{-2}$)	order	error($\times 10^{-2}$)	order
2D	500	1.7141		4.2706	
	1000	1.1420	0.59	3.4157	0.32
	2000	0.7702	0.59	2.6225	0.38
	4000	0.6401	0.27	2.2505	0.22
4D	500	1.8735		3.5183	
	1000	1.4468	0.37	3.0786	0.19
	2000	1.0557	0.45	2.6561	0.21
	4000	0.8076	0.39	2.0410	0.38
8D	500	2.0737		2.4514	
	1000	1.5714	0.40	2.2083	0.15
	2000	1.1607	0.43	2.0297	0.12
	10000	0.8139	0.22	1.1551	0.35
16D	2000	0.8613		2.0754	
	5000	0.6863	0.25	1.3506	0.47
	10000	0.5361	0.36	1.1383	0.25
	20000	0.4623	0.21	1.2298	0.11

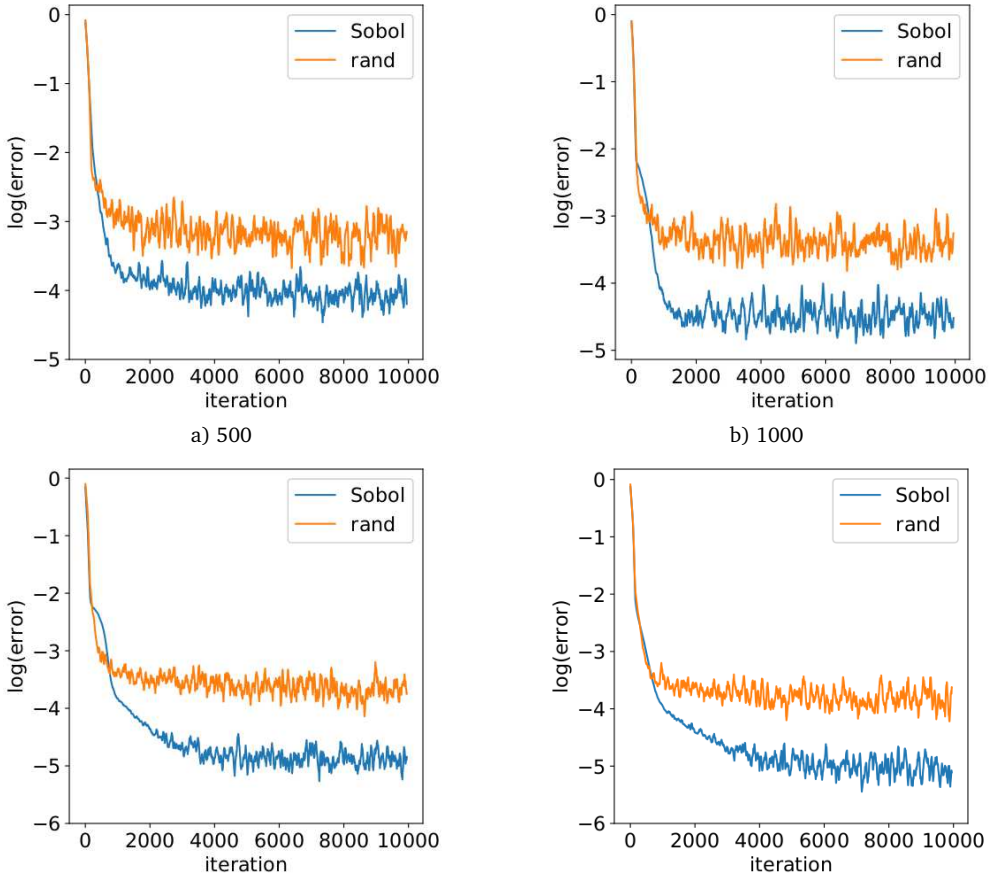
Table 3: Fitted convergence rates of the relative L^2 error with respect to the mini-batch size in different dimensions (recorded in terms of $\frac{1}{N}$) for Dirichlet problem.

Dimension	QMC	MC
2D	0.48	0.32
4D	0.41	0.26
8D	0.31	0.26
16D	0.28	0.24



a) Exact solution b) Trained solution

Figure 2: Exact and trained solutions to problem (3.2) in 2D. The exact solution $u(x) = \sum_{k=1}^{2} \cos(\pi x_k)$ and the approximate solution is trained with 5000 points (Sobol sequence) used at each iteration.



a) 500 b) 1000

c) 2000 d) 4000

Figure 3: Detained training processes of QMC and MC methods with different mini-batch sizes for Dirichlet problem in 2D.

Tables 2 and 3. When the size of mini-batches increases, the advantage of QMC method over MC method reduces. This is natural in the sense that both methods converge when the number of sampling points increases. We further plot detained training processes of QMC and MC methods in Fig. 3. Throughout the paper, the log function uses e as the base. Clearly, QMC sampling reduces the magnitude of error of MC method by about three times on average with the same mini-batch size for the 2D problem.

Fig. 4 plots relative L^2 error in terms of mini-batch size for QMC and MC methods for (3.2) from 2D to 16D. A clear evidence is that QMC method always outperforms MC method and the error reduction is significant.

From a different perspective, in order to achieve the same approximation accuracy, we may ask how much cheaper QMC method is compared to MC method. To do this, we fix the mini-batch size in MC method to be 10000 and 100000, and check the corresponding size in QMC method for the same accuracy requirement. Results are recorded in Table 4, and detailed training processes in 2D are plotted in Fig. 5. For the

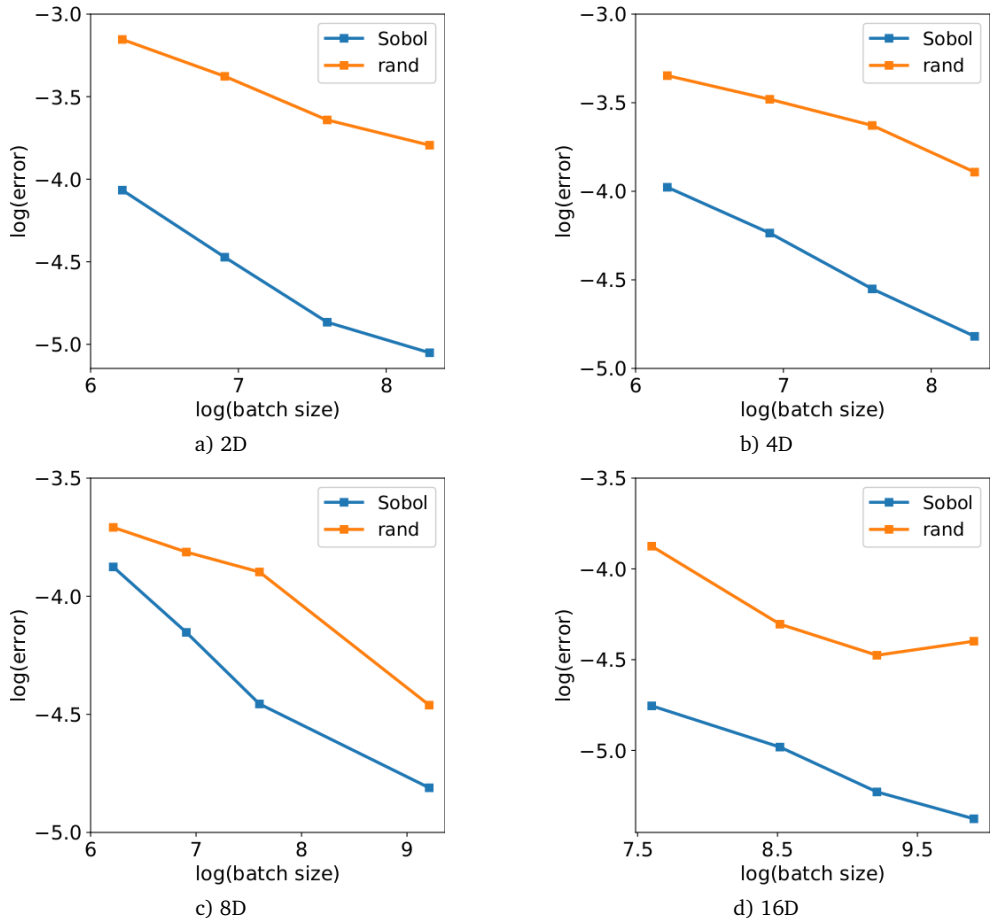
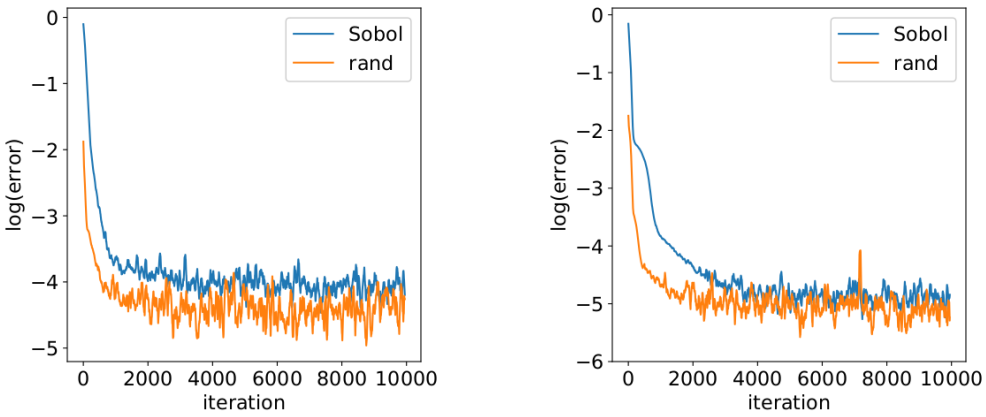


Figure 4: Relative L^2 error versus mini-batch size in QMC and MC methods for (3.2) from 2D to 16D.



a) 10000 (MC) vs 500 (QMC) b) 100000 (MC) vs 2000 (QMC)

Figure 5: Detailed training processes in QMC and MC methods for Dirichlet problem in 2D with different mini-batch sizes for the same accuracy requirement.

same accuracy requirement, compared to MC method, QMC method reduces the size of training data set by more than two orders of magnitude. Also in Table 4, we record CPU time which stands for the execution time of one iteration in the SGD method using QMC or MC method. Since runtime of the optimiser at each iteration is dominated by the approximation of the loss function where QMC or MC method is applied (determined by the number of samples) and the evaluation of the gradient with respect to parameters (determined by the number of parameters), we observe that CPU time with QMC method is about one fifth of that with MC method on average.

All above results show that QMC method always performs better than MC method, typically by several times in terms of accuracy when the same mini-batch size is enforced or by more than two orders of magnitude in terms of efficiency when the same

Table 4: Comparison of mini-batch sizes in QMC and MC methods for Dirichlet problem in different dimensions when the same approximation accuracy is required. CPU time stands for the execution time of one iteration in the SGD method using QMC or MC method.

Dimension	MC			QMC			$\frac{\text{QMC time}}{\text{MC time}}$
	size	error $\times 10^{-2}$	CPU time(s)	size	error $\times 10^{-2}$	CPU time(s)	
2D	10000	1.3566	0.0201	500	1.7141	0.0175	87%
	100000	0.7702	0.1927	2000	0.6149	0.0203	11%
4D	10000	0.1329	0.1122	500	1.8735	0.0167	15%
	100000	0.8986	0.3020	4000	0.8076	0.0221	7 %
8D	10000	1.1551	0.1399	2000	1.1607	0.0210	15%
	100000	0.8668	0.4012	10000	0.8139	0.1185	30%
16D	10000	1.1383	0.1718	1000	1.0472	0.0255	15%
	100000	0.7993	0.5064	5000	0.6863	0.1387	27%

accuracy is required. However, their convergence rates with respect to both the iteration number and the size of training data set seem to be the same, due to the ADAM optimizer we use and the nonconvex nature of loss function.

3.2. A linear Neumann boundary condition problem

Consider the Neumann problem

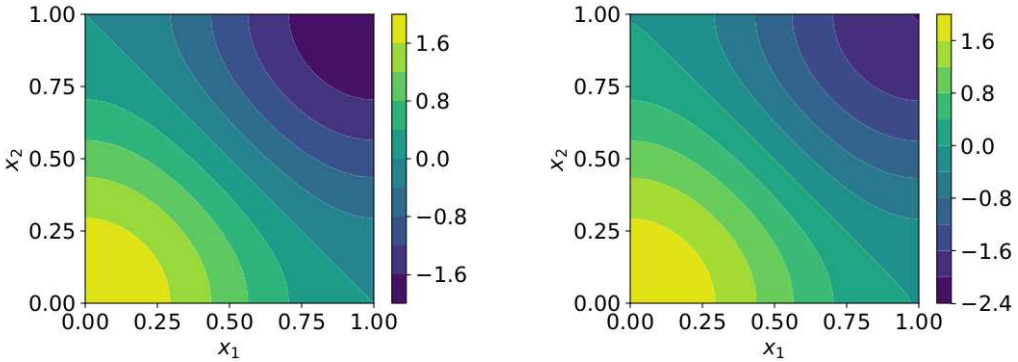
$$\begin{cases} -\Delta u + \pi^2 u = 2\pi^2 \sum_{k=1}^{K} \cos(\pi x_k), & x \in \Omega, \\ \dfrac{\partial u}{\partial n} = 0, & x \in \partial\Omega \end{cases} \tag{3.4}$$

over $\Omega = [0,1]^K$ with the exact solution $u(x) = \sum_k \cos(\pi x_k)$ and n being the unit outward normal vector. For this problem, we still do not need to add any penalty term for the boundary condition and the loss function is defined as

$$I(u) = \int_\Omega \left(\frac{1}{2} \Big(|\nabla u(x)|^2 + \pi^2 u(x)^2 \Big) - f(x)u(x) \right) \mathrm{d}x. \tag{3.5}$$

The network structure is the same as before; see (2.8). Detailed setup is listed in Table 5. Exact and trained solutions in 2D are visualized in Fig. 6. Relative L_2 errors in differential dimensions are recorded in Table 6 and the corresponding convergence rates are shown in Table 7.

Fig. 7 plots detailed training processes of QMC and MC methods with different mini-batch sizes in 2D. Similarly, QMC method outperforms MC method by several times in terms of accuracy for the same size of training data set. Besides, for the same accuracy requirement, QMC reduces the size of training data set by orders of magnitudes; see Table 8 as well as Fig. 8 for details. Again, we observe that CPU time with QMC method is about one fifth of that with MC method on average.



a) Exact solution b) Trained solution

Figure 6: Exact and trained solutions to problem (3.4) in 2D. The exact solution $u(x) = \sum_{k=1}^{2} \cos(\pi x_k)$ and the approximate solution is trained with 5000 points (Sobol sequence) used at each iteration.

Table 5: Detailed setup of the neural network used for Neumann problem in different dimensions, where m denotes the number of nodes contained in each layer.

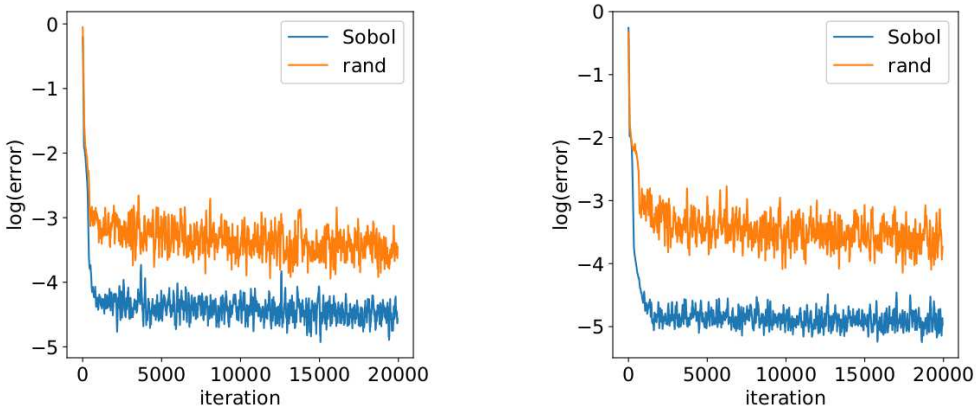
Dimension	Blocks	m	Parameters
2	4	10	921
4	4	15	2011
8	4	30	7741
16	5	48	24385

Table 6: Relative L^2 errors in different dimensions with different mini-batch sizes for Neumann problem and the convergence order is recorded with respect to $\frac{1}{N}$.

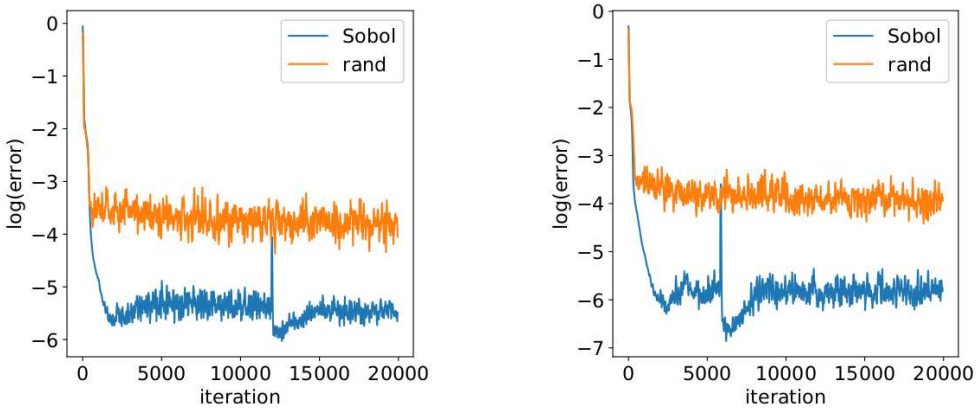
Dimension	Mini-batch size	QMC		MC	
		error($\times 10^{-2}$)	order	error($\times 10^{-2}$)	order
2D	250	1.1218		2.9571	
	500	0.7484	0.58	2.6754	0.14
	1000	0.4314	0.79	2.2572	0.25
	2000	0.3140	0.46	1.9914	0.18
4D	500	1.4219		3.9305	
	1000	0.9185	0.63	3.6990	0.09
	2000	0.3289	1.48	2.4779	0.58
	4000	0.2649	0.24	2.0152	0.23
8D	500	3.8542		7.7874	
	1000	2.7353	0.49	6.2379	0.32
	2000	2.4249	0.17	5.4659	0.19
	10000	1.8235	0.18	2.6860	0.44
16D	1000	3.2668		6.0573	
	2000	2.9308	0.16	5.8566	0.05
	5000	2.9205	0.01	4.8035	0.22
	10000	1.5636	0.90	3.8518	0.32

Table 7: Fitted convergence rates of the relative L^2 error with respect to the mini-batch size in different dimensions (recorded in terms of $\frac{1}{N}$) for Neumann problem.

Dimension	QMC	MC
2D	0.63	0.20
4D	0.78	0.32
8D	0.23	0.35
16D	0.28	0.20

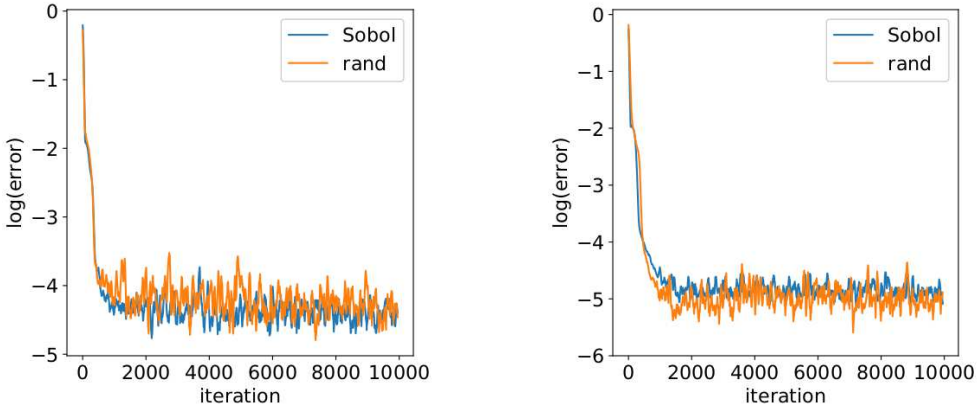


a) 250 b) 500



c) 1000 d) 2000

Figure 7: Detailed training processes of QMC and MC methods with different mini-batch sizes for Neumann problem in 2D.



a) 10000 (MC) vs 250 (QMC) b) 100000 (MC) vs 500 (QMC)

Figure 8: Detailed training processes in QMC and MC methods for Neumann problem in 2D with different mini-batch sizes for the same accuracy requirement.

Table 8: Comparison of mini-batch sizes in QMC and MC methods for Neumann problem in different dimensions when the same approximation accuracy is required. CPU time stands for the execution time of one iteration in the SGD method using QMC or MC method.

Dimension	MC			QMC			$\frac{\text{QMC time}}{\text{MC time}}$
	size	error $\times 10^{-2}$	CPU time(s)	size	error $\times 10^{-2}$	CPU time(s)	
2D	10000	1.3138	0.0264	250	1.1218	0.0208	79%
	100000	0.6536	0.2967	500	0.7484	0.0228	8%
4D	10000	2.0778	0.1372	500	1.4218	0.0234	17%
	100000	0.4622	0.3082	2000	0.3289	0.5116	61%
8D	10000	2.6860	0.1496	1000	2.7353	0.0264	17%
	100000	2.5154	0.4931	2000	2.4249	0.0256	5%
16D	10000	3.8518	0.1756	1000	3.2668	0.0261	14%
	100000	2.7805	0.5116	5000	2.9205	0.1314	26%

Generally speaking, not every Neumann problem can be modeled by a loss function without the penalty term on the boundary. Therefore, for general Neumann problem

$$\begin{cases} -\Delta u + \pi^2 u = f(x), & x \in \Omega, \\ \dfrac{\partial u}{\partial x} = g(x), & x \in \partial\Omega, \end{cases} \tag{3.6}$$

we add the penalty term into the loss function (3.5)

$$\text{loss} = I(u) + \beta \int_{\partial\Omega} \left(\frac{\partial u}{\partial x} - g(x) \right)^2 \mathrm{d}x. \tag{3.7}$$

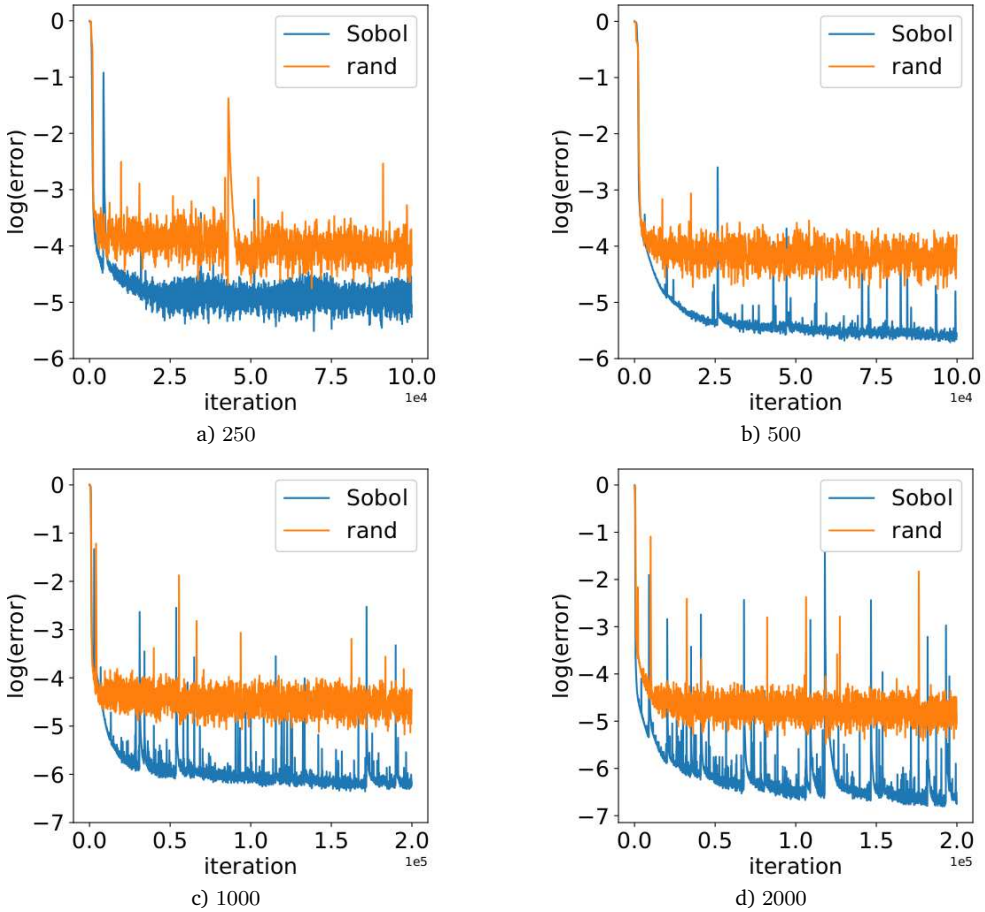
The first term is a volume integral while the second term is a boundary integral. Therefore, we have to sample these two terms separately: one for $I(u)$ in Ω and the other for the penalty term on the boundary. In principle, we need to optimize sizes of both training sets in order to minimize the approximation error. Practically, we find that QMC method always performs better than MC method. In Table 9, we show relative L^2 errors in different dimensions with different mini-batch sizes for Neumann problem with the penalty term on the boundary.

Fig. 9 plots detailed training processes of QMC and MC methods with different mini-batch sizes for Neumann problem with the penalty term on the boundary in 2D. Size of the training data set for the penalty term is fixed to be 100 in all cases.

3.3. Problem with less regular solution

Consider the problem over $\Omega = [0,1]^K$, defined as

$$\begin{cases} -\Delta u + \pi^2 u = f(x), & x \in \Omega, \\ \dfrac{\partial u}{\partial n} = 0, & x \in \partial\Omega \end{cases} \tag{3.8}$$



a) 250 b) 500

c) 1000 d) 2000

Figure 9: Detailed training processes of QMC and MC methods with different mini-batch sizes for Neumann problem with the penalty term over the boundary in 2D.

with the exact solution $u(x) = \sum_{k=1}^{K} \cos(\pi x_k^{2.5})$, whose third derivative is singular at the origin. The loss function is the same as (3.5). Detailed setup is listed in Table 10. Relative L_2 errors in different dimensions are recorded in Table 11. Detailed training processes of QMC and MC methods with different mini-batch sizes in 2D are shown in Fig. 10. Again, QMC method outperforms MC method in this case.

3.4. A nonlinear problem

Consider the nonlinear problem

$$\begin{cases} -\Delta u + u^3 = f(x,) & x \in \Omega, \\ u(x) = 0, & x \in \partial\Omega \end{cases} \tag{3.9}$$

over the unit sphere $\Omega = \{x : |x| < 1\}$ with the exact solution $u(x) = \sin(\frac{\pi}{2}(1 - |x|))$.

Table 9: Relative L^2 errors in different dimensions with different mini-batch sizes for Neumann problem with the penalty term on the boundary.

Dimension	Mini-batch size		Relative L^2 error	
	Volume	Boundary	QMC($\times 10^{-2}$)	MC($\times 10^{-2}$)
2D	250	100	0.8079	1.6151
	500	100	0.4176	1.5027
	1000	100	0.2086	1.0101
	2000	100	0.1470	0.8327
4D	500	100	0.4434	1.7622
	1000	100	0.3729	1.2550
	2000	100	0.3160	0.9799
	4000	100	0.2651	0.7530
8D	500	100	1.1770	3.8285
	1000	100	0.8989	3.0505
	2000	100	0.8780	2.9661
	10000	100	0.7643	1.6552
16D	1000	100	1.2181	3.1398
	2000	100	1.1250	2.2055
	5000	100	0.8998	1.9235
	10000	100	0.7698	1.5244

For this problem, the loss function is defined as

$$I(u) = \int_\Omega \left(\frac{1}{2} |\nabla u(x)|^2 + \frac{1}{4} u(x)^4 - f(x) u(x) \right) \mathrm{d}x. \tag{3.10}$$

Similar to (3.3), we construct the network in the form of

$$\begin{cases} \hat{u}_\theta(x) = A(x) \cdot f_\theta(x), & x \in \Omega, \\ A(x) = 1 - |x|, & x \in \Omega. \end{cases} \tag{3.11}$$

The detailed setup is listed in Table 12. Relative L_2 errors in different dimensions are recorded in Table 13. Detailed training processes of QMC and MC methods with different mini-batch sizes in 2D are shown in Fig. 11. Note that the exact solution here is only continuously differentiable and its mixed first derivative does not exist at the origin. It is known that QMC method works with provable convergence when the mixed first derivative of the integrand exists and is bounded [9]. Therefore, we should not expect that QMC method works better than MC method in this scenario. To our surprise, as demonstrated later, QMC method outperforms MC method consistently.

Table 10: Detailed setup of the neural network used for the linear problem with less regular solution in different dimensions, where m denotes the number of nodes contained in each layer.

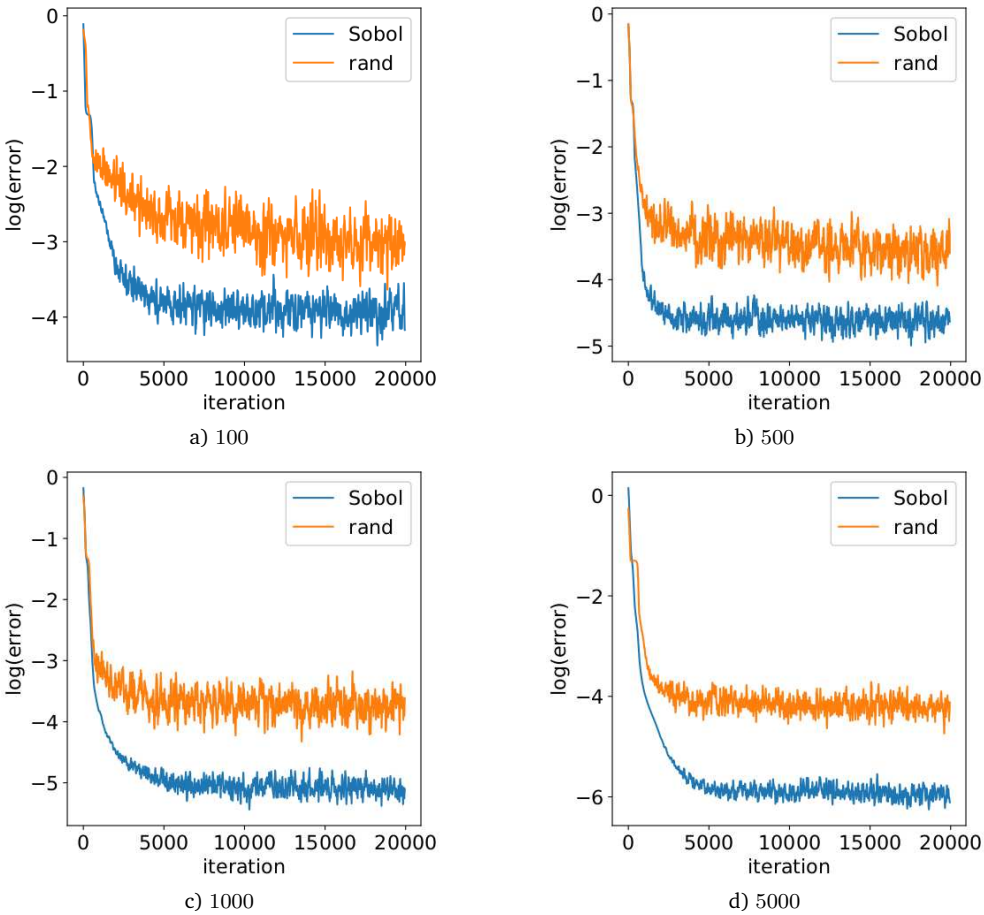
Dimension	Blocks	m	Parameters
2	3	8	465
4	3	16	1729
8	3	16	1793
16	3	20	1921

Table 11: Relative L^2 errors in different dimensions with different mini-batch sizes for the linear problem with less regular solution and the convergence order is recorded with respect to $\frac{1}{N}$.

Dimension	Mini-batch size	QMC		MC	
		error($\times 10^{-2}$)	order	error($\times 10^{-2}$)	order
2D	100	2.0921		4.9784	
	500	0.9959	0.4612	2.8479	0.3466
	1000	0.6002	0.7306	2.4802	0.2003
	5000	0.2653	0.5071	1.4717	0.3242
4D	500	1.1917		3.1804	
	1000	1.1742	0.0213	2.7650	0.2019
	5000	0.5670	0.4523	1.7535	0.2896
	10000	0.4903	0.2094	1.5728	0.15689
8D	500	1.9235		3.8439	
	1000	1.6064	0.2598	3.1926	0.2678
	5000	1.0859	0.2433	1.8432	0.3413
	10000	0.8102	0.4225	1.7183	0.1012
16D	5000	0.6315		1.9112	
	10000	0.6258	0.1298	1.4771	0.3718
	20000	0.4931	0.3439	1.3127	0.1702
	25000	0.3266	1.4523	1.2121	0.3574

Table 12: Detailed setup of the neural network used for the nonlinear problem in different dimensions, where m denotes the number of nodes contained in each layer.

Dimension	Blocks	m	Parameters
2	3	8	465
4	3	12	1009
6	3	14	1373
8	3	16	1793



a) 100 b) 500

c) 1000 d) 5000

Figure 10: Detailed training processes of QMC and MC methods with different mini-batch sizes for the linear problem with less regular solution in 2D.

4. Convergence analysis

To understand numerical results in Section 3 from a theoretical perspective, in this section, we shall analyze the convergence behavior of SGD method with QMC sampling and understand how the size of training data set affects the convergence for different sampling strategies. Our analysis follows [2], but differs in terms of assumptions. Under the boundedness assumption of parameter sequence, we prove the Lipschitz continuity of loss function for smooth activation functions, such as (2.6), instead of the direct assumption of Lipschitz continuity of loss function. From a practical perspective, our assumption can be easily verified during the iteration while the Lipschitz continuity is difficult to be verified. Moreover, we prove the convergence of function sequences generated by MC and QMC methods under a stronger assumption on the second moment of the stochastic vector used in the SGD method, which explicitly characterizes the dependence of convergence rate on both the size of training data set and the it-

Table 13: Relative L^2 errors in different dimensions with different mini-batch sizes for the nonlinear problem and the convergence order is recorded with respect to $\frac{1}{N}$.

Dimension	mini-batch size	QMC		MC	
		error($\times 10^{-2}$)	order	error($\times 10^{-2}$)	order
2D	500	0.4762		1.7222	
	1000	0.3249	0.5516	1.4203	0.2781
	5000	0.2897	0.0713	0.9195	0.2701
	10000	0.1893	0.6136	0.8042	0.1933
4D	1000	1.1532		1.9681	
	5000	0.8452	0.1931	1.4303	0.1967
	10000	0.7778	0.1199	1.1198	0.2583
	15000	0.6753	0.3484	1.0651	0.2919
6D	5000	0.9728		2.4343	
	10000	0.6227	0.6451	2.3791	0.3734
	15000	0.5340	0.3789	2.2035	0.3928
	20000	0.4272	0.7752	2.1200	0.1346
8D	5000	3.6587		5.6010	
	10000	1.6207	1.1747	4.1136	0.4452
	15000	1.3479	0.4546	3.2380	0.5903
	20000	1.7102	0.8277	2.9620	0.3096

eration number. We shall stress that the convexity assumption of loss functions does not hold practically. Even when the loss function for the original PDE is convex in the functional space, the problem loses its convexity when DNN approximations are used. Therefore, the analysis here shows how the convergence rates are in the ideal case. But in practice, numerical results of both MC and QMC do not agree well with theoretical results.

For convenience, derivatives are taken with respect to the parameter set if subscripts are not specified in what follows. We start with the following assumption.

Assumption 4.1 (Boundedness of the iteration sequence)**.** The iteration sequence $\{\theta_i\}_{i=0}^{\infty}$ is bounded, i.e., $\forall i \in \mathbb{N}^+$,

$$\|\theta_i\|_2 \leq C_\theta.$$

A direct consequence of Assumption 4.1 is the existence of convergent sub-sequences. It will be shown later that the whole sequence of function values generated by the SGD method with a sampling strategy converges if the loss function is strongly convex with respect to θ. In practice, this assumption can be verified easily in the iteration. Furthermore, we choose a C^∞ activation function $\sigma(x)$; see (2.6). Consequently, the residual network also belongs to C^∞ with respect to both θ and x. Therefore, the Lipschitz continuity of the approximate solution by the neural network is guaranteed in a bounded domain.

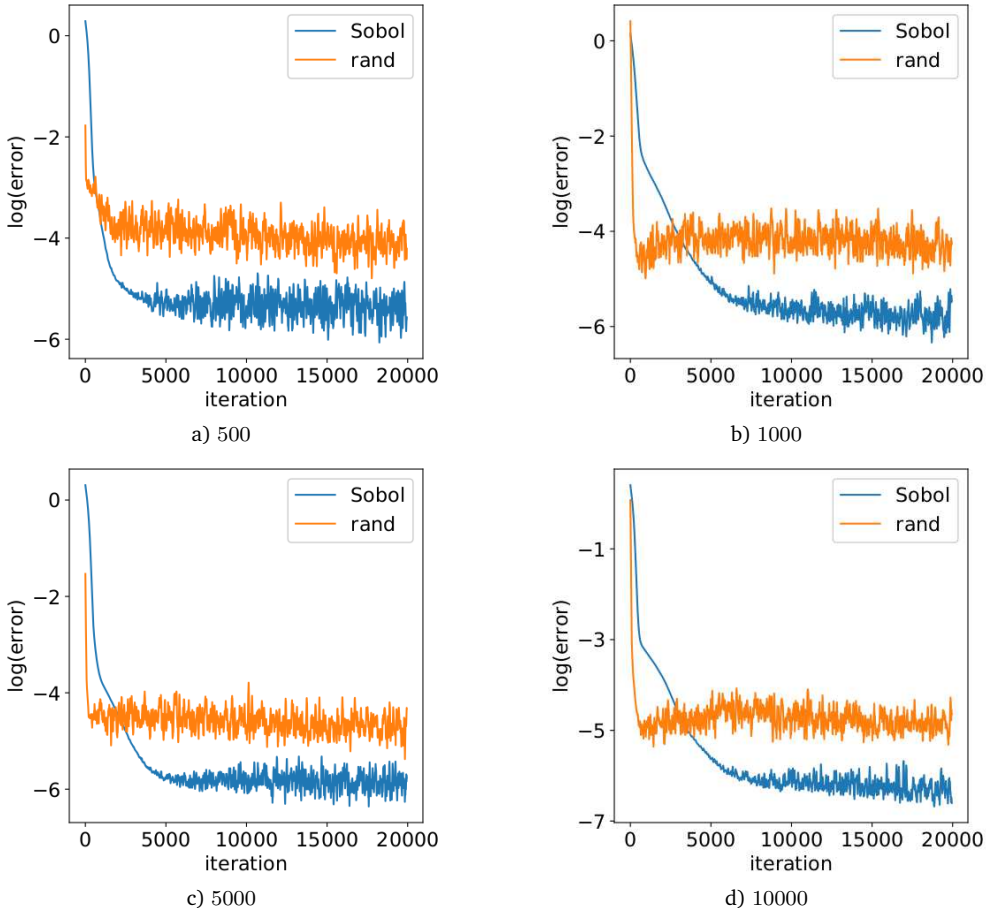


Figure 11: Detailed training processes of QMC and MC methods with different mini-batch sizes for the nonlinear problem in 2D.

Lemma 4.1. *If the activation function belongs to $C^\infty(\Omega)$ with the bounded domain Ω, then under Assumption 4.1 there exist constants L_0, L_1 and L_2 such that*

$$\begin{aligned}
&\|\nabla \hat{u}_\theta - \nabla \hat{u}_{\bar{\theta}}\|_2 \le L_0 \|\theta - \bar{\theta}\|_2, \\
&\|\nabla_x \hat{u}_\theta - \nabla_x \hat{u}_{\bar{\theta}}\|_2 \le L_1 \|\theta - \bar{\theta}\|_2, \\
&\|\nabla \nabla_x \hat{u}_\theta - \nabla \nabla_x \hat{u}_{\bar{\theta}}\|_2 \le L_2 \|\theta - \bar{\theta}\|_2
\end{aligned}$$

for any $\theta, \bar{\theta} \in \{\theta_i\}_{i=1}^\infty$ and $x \in \Omega$.

A natural corollary of Lemma 4.1 is that there exist constants M_1 and M_2 such that

$$\|\nabla_x \hat{u}_\theta(x)\|_2 \le M_1, \quad \|\nabla \nabla_x \hat{u}_\theta(x)\|_2 \le M_2 \tag{4.1}$$

for any $\theta \in \{\theta_i\}_{i=1}^\infty$ and $x \in \Omega$.

Remember that the above results hold for smooth activation functions, but do not hold for ReLU activation function. Then we can only expect the boundness of $\nabla_x \hat{u}_\theta(x)$

and $\nabla\nabla_x\hat{u}_\theta(x)$, instead of Lipschitz continuity. For example, consider the simplest network with two layers. For the ReLU activation function [26], there exists a constant M depending on Ω such that

$$\|\nabla\hat{u}_\theta - \nabla\hat{u}_{\bar{\theta}}\|_2 \leq L_0\|\theta - \bar{\theta}\|_2 + M. \tag{4.2}$$

Based on Lipschitz continuity of the neural network solution and boundedness of derivatives, Lipschitz continuity of the loss function can be proven.

Theorem 4.1 (Lipschitz continuity of the loss function)**.** *For any θ and $\bar{\theta}$, the loss function of the neural network satisfies*

$$\|\nabla I(\theta) - \nabla I(\bar{\theta})\|_2 \leq L\|\theta - \bar{\theta}\|_2. \tag{4.3}$$

Proof. From (2.3), we have

$$\nabla I(\theta) = \int_\Omega \nabla_x\hat{u}_\theta(x)^T \cdot \nabla\nabla_x\hat{u}_\theta(x)\mathrm{d}x - \int_\Omega f(x)\nabla\hat{u}_\theta(x)\mathrm{d}x.$$

By the triangle inequality, we have

$$\begin{aligned}
\|\nabla I(\theta) - \nabla I(\bar{\theta})\|_2 \leq & \left\| \int_\Omega \big(\nabla\nabla_x\hat{u}_\theta(x)\cdot\nabla_x\hat{u}_\theta(x) - \nabla\nabla_x\hat{u}_\theta(x)\cdot\nabla_x\hat{u}_{\bar{\theta}}(x)\big)\mathrm{d}x \right\|_2 \\
& + \left\| \int_\Omega \big(\nabla\nabla_x\hat{u}_\theta(x)\cdot\nabla_x\hat{u}_{\bar{\theta}}(x) - \nabla\nabla_x\hat{u}_{\bar{\theta}}(x)\cdot\nabla_x\hat{u}_{\bar{\theta}}(x)\big)\mathrm{d}x \right\|_2 \\
& + \left\| \int_\Omega f(x)\big(\nabla\hat{u}_\theta(x) - \nabla\hat{u}_{\bar{\theta}}(x)\big)\mathrm{d}x \right\|_2 \\
\leq & M_2L_1\|\theta - \bar{\theta}\|_2 + M_1L_2\|\theta - \bar{\theta}\|_2 + L_0\max_{x\in\Omega}|f(x)|\,\|\theta - \bar{\theta}\|_2 \\
\leq & L\|\theta - \bar{\theta}\|_2
\end{aligned}$$

with $L = M_2L_1 + M_1L_2 + L_0\max_{x\in\Omega}|f(x)|$. □

And an important consequence of the Theorem 4.1 is that for $\forall\theta, \bar{\theta} \in \{\theta_i\}_{i=1}^\infty$,

$$I(\theta) \leq I(\bar{\theta}) + \nabla I(\bar{\theta})^T(\theta - \bar{\theta}) + \frac{1}{2}L\|\theta - \bar{\theta}\|_2^2. \tag{4.4}$$

To proceed, we need the following assumptions.

Assumption 4.2 (First and second moment assumption)**.** For the stochastic vector, assume that the first and second moments satisfy

- There exists $0 < \mu \leq \mu_G$ such that, $\forall i \in \mathbb{N}^+$,

$$\nabla I(\theta_i)^T\mathbb{E}_{\xi_i}[g(\theta_i, \xi_i)] \geq \mu\big(1 + r(N)\big)\|\nabla I(\theta_i)\|_2^2, \tag{4.5}$$
$$\|\mathbb{E}_{\xi_i}[g(\theta_i, \xi_i)]\|_2 \leq \mu_G\big(1 + r(N)\big)\|\nabla I(\theta_i)\|_2. \tag{4.6}$$

- For the second moment, there exist $C_V \geq 0$ and $M_V \geq 0$ such that, $\forall i \in \mathbb{N}^+$,

$$\mathbb{V}_{\xi_i}[g(\theta_i, \xi_i)] \leq C_V r(N) + M_V \big(1 + r(N)\big) \|\nabla I(\theta_i)\|_2^2, \tag{4.7}$$

where

$$r(N) = \big(D(P_N)\big)^2, \quad r(N) = \mathcal{O}\left(\frac{1}{N}\right)$$

for MC sampling and

$$r(N) = \mathcal{O}\left(\frac{\big(\ln(N)\big)^{2K}}{N^2}\right)$$

for QMC sampling.

Note that $g(\theta_i, \xi_i)$ is an unbiased estimate of $\nabla I(\theta_i)$ as $N \to \infty$ when $\mu = \mu_G = 1$ in Assumption 4.2. Moreover, according to (4.7), we have

$$\begin{aligned} \mathbb{E}_{\xi_i}\left[\|g(\theta_i, \xi_i)\|_2^2\right] &= \mathbb{V}_{\xi_i}[g(\theta_i, \xi_i)] + \|\mathbb{E}_{\xi_i}[g(\theta_i, \xi_i)]\|_2^2 \\ &\leq C_V r(N) + M_G(1 + r(N))\|\nabla I(\theta_i)\|_2^2 \end{aligned}$$

with $M_G = M_V + \mu_G^2(1 + r(N))$. For a fixed stepsize $\alpha_i = \alpha$, according to (4.4), we have

$$\begin{aligned} &\mathbb{E}_{\xi_i}[I(\theta_{i+1})] - I(\theta_i) \\ &\leq \frac{L}{2}\alpha^2 C_V r(N) - \alpha\left(\mu - \frac{L}{2}\alpha M_G\right)\big(1 + r(N)\big)\|\nabla I(\theta_i)\|_2^2. \end{aligned} \tag{4.8}$$

Assumption 4.3 (Strong convexity)**.** The loss function $I(\theta)$ is strongly convex with respect to θ, i.e., there exists a constant c such that

$$I(\bar{\theta}) \geq I(\theta) + \nabla I(\theta)^T(\bar{\theta} - \theta) + \frac{c}{2}\|\bar{\theta} - \theta\|_2^2$$

for any $\theta, \bar{\theta} \in \{\theta_i\}_{i=1}^{\infty}$.

The strong convexity assumption is necessary for the convergence analysis of SGD method. However, due to function compositions and nonlinear activation functions, this assumption does not hold for DNNs. We will come back to this issue at the end of the section.

Following [2], if $I(\theta)$ is strongly convex and θ^* is the unique minimizer, then for any $\theta \in \{\theta_i\}_{i=1}^{\infty}$ we have

$$2c\big(I(\theta) - I(\theta^*)\big) \leq \|\nabla I(\theta)\|_2^2 \tag{4.9}$$

with $c \leq L$.

Theorem 4.2 (Strongly convex loss function, fixed stepsize, MC and QMC samplings)**.** *Let the data set $P_N = \{x^1, x^2, \ldots, x^N\} \subset \Omega$ be generated by MC or QMC method, the stepsize $\alpha_i = \alpha$ be a constant satisfying*

$$0 < \alpha \leq \frac{\mu}{LM_G}.$$

Under Assumptions 4.1-4.3*, the expected optimality gap satisfies*

$$\lim_{i\to\infty} \mathbb{E}[I(\theta_i) - I(\theta^*)] = \frac{\alpha L C_V}{2c\mu} r(N) \tag{4.10}$$

with θ^* being the unique minimizer.

Proof. Using (4.8) and (4.9), for any $i \in \mathbb{N}$, we have

$$\begin{aligned}
\mathbb{E}_{\xi_i}[I(\theta_{i+1})] - I(\theta_i) &\leq \frac{L}{2}\alpha^2 C_V r(N) - \alpha\Big(\mu - \frac{L}{2}\alpha M_G\Big)\big(1 + r(N)\big)\|\nabla I(\theta_i)\|_2^2 \\
&\leq \frac{L}{2}\alpha^2 C_V r(N) - \frac{\mu\alpha}{2}\big(1 + r(N)\big)\|\nabla I(\theta_i)\|_2^2 \\
&\leq \frac{L}{2}\alpha^2 C_V r(N) - c\mu\alpha\big(1 + r(N)\big)\big(I(\theta_i) - I(\theta^*)\big) \\
&\leq \frac{L}{2}\alpha^2 C_V r(N) - c\mu\alpha\big(I(\theta_i) - I(\theta^*)\big),
\end{aligned}$$

since $I(\theta_i) - I(\theta^*) \geq 0$ holds under the assumption of strongly convexity. Adding $I(\theta_i) - I(\theta^*)$ to both sides of the above inequality and taking total expectation yields

$$\mathbb{E}[I(\theta_{i+1}) - I(\theta^*)] \leq (1 - c\mu\alpha)\mathbb{E}[I(\theta_i) - I(\theta^*)] + \frac{L}{2}\alpha^2 C_V r(N).$$

Subtracting $\frac{\alpha L C_V}{2c\mu} r(N)$ from both sides produces

$$\mathbb{E}[I(\theta_{i+1}) - I(\theta^*)] - \frac{\alpha L C_V}{2c\mu} r(N) \leq (1 - c\mu\alpha)\Big(\mathbb{E}[I(\theta_i) - I(\theta^*)] - \frac{\alpha L C_V}{2c\mu} r(N)\Big).$$

Note that

$$0 < c\mu\alpha \leq \frac{c\mu^2}{LM_G} \leq \frac{c\mu^2}{L\mu^2} = \frac{c}{L} \leq 1.$$

A recursive argument yields

$$\mathbb{E}[I(\theta_{i+1}) - I(\theta^*)] - \frac{\alpha L C_V}{2c\mu} r(N) \leq (1 - c\mu\alpha)^i\Big(\mathbb{E}[I(\theta_1) - I(\theta^*)] - \frac{\alpha L C_V}{2c\mu} r(N)\Big),$$

and thus

$$\lim_{i\to\infty} \mathbb{E}[I(\theta_i) - I(\theta^*)] = \frac{\alpha L C_V}{2c\mu} r(N). \qquad \square$$

Theorem 4.2 provides a quantitative estimate for the accuracy of sampling strategies on the convergence rate of the iteration sequence. Regardless of the sampling strategy, the convergence rate is linear which is determined by the SGD method.

Remark 4.1. When $N \to \infty$, for both MC and QMC methods, we have $r(N) \to 0$. From Theorem 4.2, we conclude that

$$\lim_{N\to\infty} \lim_{i\to\infty} \mathbb{E}[I(\theta_i) - I(\theta^*)] = 0.$$

In practice, however, with the increasing size of training data set, the gap usually does not tend to 0 under the fixed stepsize condition. We attributes this to the irrationality of (4.7). Instead, (4.7) shall be relaxed to [2]

$$\mathbb{V}_{\xi_i}[g(\theta_i, \xi_i)] \leq C_V + M_V\big(1 + r(N)\big)\|\nabla I(\theta_i)\|_2^2.$$

Then we have

$$\lim_{N\to\infty} \lim_{i\to\infty} \mathbb{E}[I(\theta_i) - I(\theta^*)] = \frac{\alpha L C_V}{2c\mu},$$

which implies the convergence of the sequence of function values near the optimal value. Therefore, an optimizer with fixed stepsize is generally not the best choice [2]. Instead, the SGD method with diminishing stepsize is popular in real applications, such as ADAM method [20] used in the implementation.

Remark 4.2. From the convergence analysis (4.10), we expect that QMC method outperforms MC method in terms of convergence order. It is reasonable to find from Tables 3, 7, 11 and 13 that QMC method has better rates than MC method. However, numerical convergence rates are not as significant as theoretical convergence rates for both QMC and MC methods. We attribute this difference to the nonconvexity of the loss function and the above analysis relies crucially on the strong convexity assumption of the same function (Assumption 4.3). Whatever, for the same accuracy requirement in practice, QMC method outperforms MC method in terms of efficiency by orders of magnitude. Therefore, we recommend the usage of QMC sampling when solving PDEs by DNNs whenever a sampling strategy is needed.

5. Conclusion

In this paper, we have proposed to approximate the loss function using quasi-Monte Carlo method, instead of Monte Carlo method that is commonly used when solving PDEs by DNNs. Numerical results based on deep Ritz method have shown the significant advantage of quasi-Monte Carlo method in terms of accuracy or efficiency. All the codes that generate numerical results included in this work are available from `https://github.com/Lyupinpin/DeepRitzMethod`.

Theoretically, we have proved the convergence of neural network solver based on quasi-Monte Carlo sampling in terms of the sampling size and the iteration number. Although there are still some practical issues such as the nonconvexity of the loss function, our analysis does provide a comprehensive understanding of why quasi-Monte Carlo method always outperforms Monte-Carlo method and suggests the usage of the former whenever an approximation of high-dimensional integrals is needed.

Acknowledgements

This work is supported in part by the grants NSFC (No. 11971021) and by National Key R&D Program of China (No. 2018YF645B0204404) (J. Chen), NSFC (No. 11501399) (R. Du). P. Li is grateful to Ditian Zhang for very helpful discussions and acknowledges the financial support of the Postgraduate Research and Practice Innovation Program of Jiangsu Province (Project KYCX20_2711). L. Lyu acknowledges the financial support of Undergraduate Training Program for Innovation and Entrepreneurship, Soochow University (Project 201810285019Z). Part of the work was done when L. Lyu was doing a summer internship at Department of Mathematics, Hong Kong University of Science and Technology. L. Lyu would like to thank its hospitality.

References

[1] C. BECK, W. E, AND A. JENTZEN, *Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations*, J. Nonlinear Sci., 29 (2019), 1563–1619.

[2] L. BOTTOU, F. E. CURTIS, AND J. NOCEDAL, *Optimization for large-scale machine learning*, SIAM Review, 60 (2018), 223–311.

[3] S. C. BRENNER AND L. R. SCOOT, *The Mathematical Theory of Finite Element Method*, Springer, 2007.

[4] A. BUCHHOLZ, F. WENZEL, AND S. MANDT, *Quasi-Monte Carlo variational inference*, arXiv, 1807.01604 (2018).

[5] H.-J. BUNGARTZ AND M. GRIEBEL, *Sparse grids*, Acta Numer., 13 (2004), 147–269.

[6] R. CAFLISCH, *Monte Carlo and Quasi-Monte Carlo Methods*, Vol. 7, Cambridge University Press.

[7] J. A. G. CERVERA, *Solution of the Black-Scholes equation using artificial neural networks*, J. Phys. Conf. Ser., 1221 (2019), 012044.

[8] M. DAUGE AND R. STEVENSON, *Sparse tensor product wavelet approximation of singular functions*, SIAM J. Math. Anal., 42 (2010), 2203–2228.

[9] J. DICK, F. Y. KUO, AND I. H. SLOAN, *High-dimensional integration: The quasi-Monte Carlo way*, Acta Numer., 22 (2013), 133–288.

[10] 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), 349–380.

[11] 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.

[12] Y. FAN, F.-F. JORDI, L. LIN, L. YING, AND L. ZEPEDA-NÚÑEZ, *A multiscale neural network based on hierarchical nested bases*, Res. Math. Sci., 6 (2019), 21.

[13] H. GEOFFREY, D. LI, Y. DONG, D. GEORGE, M. ABDEL-RAHMAN, J. NAVDEEP, S. ANDREW, V. VINCENT, N. PATRICK, K. BRIAN, AND S. TARA, *Deep neural networks for acoustic modeling in speech recognition*, IEEE Signal Process. Mag., 29 (2012), 82–97.

[14] T. GERSTNER AND M. GRIEBEL, *Numerical integration using sparse grids*, Numer. Algorithms, 18 (1998), 209.

[15] C. GIUSEPPE AND T. MATTHIAS, *Solving the quantum many-body problem with artificial neural networks*, Science, 355 (2017), 602–606.

[16] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep Learning*, MIT Press, 2016.

[17] J. HAN, A. JENTZEN, AND W. E, *Solving high-dimensional partial differential equations using deep learning*, Proceedings of the National Academy of Sciences of the United States of America, 115 (2018), 8505–8510.
[18] J. HAN, L. ZHANG, AND W. E, *Solving many-electron Schrödinger equation using deep neural networks*, J. Comput. Phys., 399 (2019).
[19] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2 (2016), 770–778.
[20] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, CoRR, 1412.6980 (2014).
[21] A. KRIZHEVSKY, I. SUTSKEVER, AND G. E. HINTON, *Imagenet classification with deep convolutional neural networks*, Commun. ACM, 60 (2012), 84–90.
[22] E. C. LAWRENCE, *Partial Differential Equations (Second Edition)*, AMS, 2010.
[23] R. J. LEVEQUE, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, SIAM, 2007.
[24] J. S. LIU, *Monte Carlo strategies in scientific computing*, Springer Science & Business Media, 2008.
[25] L. LYU, Z. ZHANG, AND J. CHEN, *A QMC-deep learning method for diffusivity estimation in random domains*, Numer. Math. Theor. Meth. Appl., 13 (2020), 908–927.
[26] V. NAIR AND G. E. HINTON, *Rectified linear units improve restricted boltzmann machines*, In Proceedings of the 27th international conference on machine learning (ICML-10), 2010, 807–814.
[27] H. NIEDERREITER, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, 1992.
[28] Y. OGATA, *A Monte Carlo method for high dimensional integration*, Numer. Math. (Heidelb), 55 (1989), 137–157.
[29] P. RAMACHANDRAN, B. ZOPH, AND Q. V. LE, *Searching for activation functions*, arXiv, 1710.05941 (2017).
[30] R. SARIKAYA, G. E. HINTON, AND A. DEORAS, *Application of deep belief networks for natural language understanding*, IEEE Trans. Audio Speech Lang. Process., 22 (2014), 778–784.
[31] J. SHEN, T. TANG, AND L.-L. WANG, *Spectral Methods: Algorithm, Analysis and Application*, Springer, 2011.
[32] J. SIRIGNANO AND K. SPILIOPOULOS, *DGM: A deep learning algorithm for solving partial differential equations*, J. Comput. Phys., 375 (2018), 1339–1364.
[33] I. M. SOBOL, *Uniformly distributed sequences with an additional uniform property*, USSR Comput. Maths. Math. Phys., 16 (1976), 236–242.
[34] B. TUFFIN, *Randomization of Quasi-Monte Carlo methods for error estimation: survey and normal approximation*, Monte Carlo Methods Appl., 10 (2004), 617–628.
[35] T. WANG, D. J. WU, A. COATES, AND A. Y. NG, *End-to-end text recognition with convolutional neural networks*, Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), (2012), 3304–3308.