CSIAM Trans. Appl. Math.
doi: 10.4208/csiam-am.SO-2020-0006

Vol. **3**, No. 2, pp. 299-317
June 2022

Implicit Bias in Understanding Deep Learning for Solving PDEs Beyond Ritz-Galerkin Method

Jihong Wang[1], Zhi-Qin John Xu[2,*], Jiwei Zhang[3,*] and Yaoyu Zhang[2]

[1] *Beijing Computational Science Research Center, Beijing 100193, P.R. China.*
[2] *School of Mathematical Sciences, Institute of Natural Sciences, MOE-LSC and Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai 200240, P.R. China.*
[3] *School of Mathematics and Statistics, and Hubei Key Laboratory of Computational Science, Wuhan University, Wuhan 430072, P.R. China.*

Received 18 November 2020; Accepted 28 September 2021

Abstract. This paper aims at studying the difference between Ritz-Galerkin (R-G) method and deep neural network (DNN) method in solving partial differential equations (PDEs) to better understand deep learning. To this end, we consider solving a particular Poisson problem, where the information of the right-hand side of the equation f is only available at n sample points, that is, f is known at finite sample points. Through both theoretical and numerical studies, we show that solution of the R-G method converges to a piecewise linear function for the one dimensional problem or functions of lower regularity for high dimensional problems. With the same setting, DNNs however learn a relative smooth solution regardless of the dimension, this is, DNNs implicitly bias towards functions with more low-frequency components among all functions that can fit the equation at available data points. This bias is explained by the recent study of frequency principle. In addition to the similarity between the traditional numerical methods and DNNs in the approximation perspective, our work shows that the implicit bias in the learning process, which is different from traditional numerical methods, could help better understand the characteristics of DNNs.

AMS subject classifications: 35Q68, 65N30, 65N35

Key words: Deep learning, Ritz-Galerkin method, partial differential equations, F-Principle.

1 Introduction

Deep neural networks (DNNs) become increasingly important in scientific computing fields [5–7,10–13,16,17,22,26,31]. A major potential advantage over traditional numerical

*Corresponding author. *Email addresses:* jhwang@csrc.ac.cn (J. Wang), xuzhiqin@sjtu.edu.cn (Z. Xu), jiweizhang@whu.edu.cn (J. Zhang), zhyy.sjtu@sjtu.edu.cn (Y. Zhang)

methods is that DNNs could overcome the curse of dimensionality in high-dimensional problems. With traditional numerical methods, several studies have made progress on the understanding of the algorithm characteristics of DNNs. For example, by exploring ReLU DNN representation of continuous piecewise linear function in FEM, the work [13] theoretically establishes that a ReLU DNN can accurately represent any linear finite element functions. In the aspect of the convergence behavior, the works [32, 33] show a Frequency Principle (F-Principle) that DNNs often learn low-frequency components first while most of the conventional methods (e.g., Jacobi method) exhibit the opposite convergence behavior—higher-frequency components are learned faster. These understandings could lead to a better use of DNNs in practice, such as DNN-based algorithms are proposed based on the F-Principle to fast eliminate high-frequency error [3, 17].

As the DNN-based algorithms are increasingly important in solving PDEs, it is important to study the property of the DNN solution. The aim of this paper is to investigate the different behaviors between DNNs and Ritz-Galerkin (R-G) method (as a traditional numerical method). To this end, we utilize an example to show their stark difference, that is, solving PDEs only with a few given sample points. We denote n by the sample number and m by the basis number in the Ritz-Galerkin method or the neuron number in DNNs. In traditional PDE models, we consider the situation where the source functions in the equation are completely known, i.e. the sample number n can go to infinity. But in practical applications, such as signal processing, statistical mechanics, chemical and biophysical dynamic systems, we often encounter the problems that only a few sample values can be obtained. It is interesting to ask what effect R-G methods would have on solving this particular problem, and what the solution would be obtained by the DNN method. On the other hand, DNN is well-known often over-parameterized in real applications. For a fair comparison, the R-G method is also set as over-parameterized when the number of basis functions goes to infinity.

In this paper, we show that R-G method considers the discrete sampling points as linear combinations of Dirac delta functions, while DNN method always uses a relatively smooth function to interpolate the discrete sampling points. And we incorporate the F-Principle to show how DNN method is different from the R-G method, that is, for all functions that can fit the training data, DNNs implicitly bias towards functions with more low-frequency components. In addition to the similarity between the traditional numerical methods and DNNs in the approximation perspective [13], our work shows that the implicit bias in the learning process, which is different from traditional numerical methods, could help better understand the characteristics of DNNs.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the R-G method and the DNN method. In Sections 3 and 4, we present the difference between the two methods in solving PDEs numerically, and provide some theoretical analysis. We end the paper with the conclusion in Section 5.

2 Preliminary

In this section we take the toy model of Poisson's equation as an example to investigate the difference of solution behaviors between R-G method and DNN method.

2.1 Poisson problem

We consider the d-dimensional Poisson problem posed on the bounded domain $\Omega \subset \mathbb{R}^d$ with Dirichlet boundary condition as

$$\begin{cases} -\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega, \end{cases} \tag{2.1}$$

where Δ represents the Laplace operator, $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$ is a d-dimensional vector. It is known that the problem (2.1) admits a unique solution for $f \in L^2(\Omega)$, and its regularity can be raised to $C_b^{s+2}(\Omega)$ if $f \in C_b^s(\Omega)$ for some $s \geq 0$. In the literature, there are a number of effective numerical methods to solve problem (2.1) in general case. Here we consider a special situation: we only have the information of $f(\boldsymbol{x})$ at the n sample points $\boldsymbol{x}_i$ $(i = 1, \cdots, n)$. In practical applications, we may imagine that we only have finite experimental data, i.e., the value of $f(\boldsymbol{x}_i)$ $(i = 1, \cdots, n)$, and have no more information of $f(\boldsymbol{x})$ at other points. Through solving such a particular Poisson problem (2.1) with R-G method and deep learning method, we aim to find the bias of these two methods in solving PDEs.

2.2 R-G method

In this subsection, we briefly introduce the R-G method [2]. For problem (2.1), we construct a functional

$$J(u) = \frac{1}{2} a(u,u) - (f,u), \tag{2.2}$$

where

$$a(u,v) = \int_\Omega \nabla u(\boldsymbol{x}) \nabla v(\boldsymbol{x}) d\boldsymbol{x}, \quad (f,v) = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) d\boldsymbol{x}.$$

The variational form of problem (2.1) is the following:

$$\text{Find } u \in H_0^1(\Omega), \text{ s.t. } J(u) = \min_{v \in H_0^1(\Omega)} J(v). \tag{2.3}$$

The weak form of (2.3) is to find $u \in H_0^1(\Omega)$ such that

$$a(u,v) = (f,v), \quad \forall v \in H_0^1(\Omega). \tag{2.4}$$

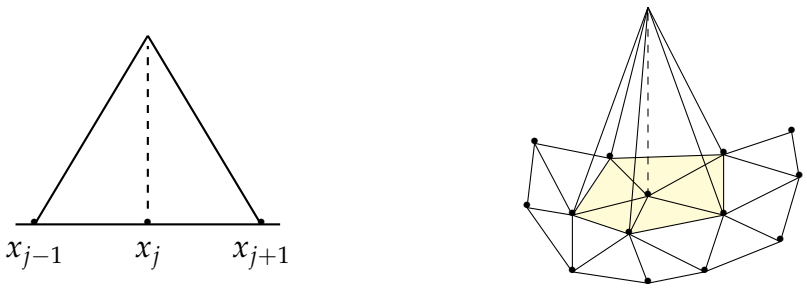

Figure 1: The finite element basis function in 1d and 2d.

The problem (2.1) is the strong form if the solution $u \in H_0^2(\Omega)$. To numerically solve (2.4), we now introduce the finite dimensional space U_h to approximate the infinite dimensional space $H_0^1(\Omega)$. Let $U_h \subset H_0^1(\Omega)$ be a subspace with a sequence of basis functions $\{\phi_1, \phi_2, \cdots, \phi_m\}$. The numerical solution $u_h \in U_h$ that we will find can be represented as

$$u_h = \sum_{k=1}^{m} c_k \phi_k, \tag{2.5}$$

where the coefficients $\{c_k\}$ are the unknown values that we need to solve. Replacing $H_0^1(\Omega)$ by U_h, both problems (2.3) and (2.4) can be transformed to solve the following system:

$$\sum_{k=1}^{m} c_k a(\phi_k, \phi_j) = (f, \phi_j), \quad j = 1, 2, \cdots, m. \tag{2.6}$$

From (2.6), we can calculate c_k, and then obtain the numerical solution u_h. We usually call (2.6) R-G equation.

For different types of basis functions, the R-G method can be divided into finite element method (FEM) and spectral method (SM) and so on. If the basis functions $\{\phi_k(\boldsymbol{x})\}$ are local, i.e., they are compactly supported, this method is usually taken as the FEM. Assume that Ω is a polygon, and we divide it into finite element grid $\mathcal{T}_h$ by simplex, $h = \max_{\tau \in \mathcal{T}_h} \mathrm{diam}(\tau)$. A typical finite element basis is the linear hat basis function, satisfying

$$\phi_k(\boldsymbol{x}_j) = \delta_{kj}, \quad \boldsymbol{x}_j \in \mathcal{N}_h, \tag{2.7}$$

where $\mathcal{N}_h$ stands for the set of the nodes of grid $\mathcal{T}_h$. The schematic diagram of the basis functions in 1-D and 2-D is shown in Fig. 1. On the other hand, if we choose the global basis function such as Fourier basis or Legendre basis [25], we call R-G method spectral method.

The error estimate theory of R-G method has been well established. Under suitable assumption on the regularity of solution, the linear finite element solution u_h has the

following error estimate

$$\|u-u_h\|_1 \leq C_1 h|u|_2,$$

where the constant C_1 is independent of grid size h and $|\cdot|_2$ is the H^2-seminorm [2, 28]. The spectral method has the following error estimate

$$\|u-u_h\| \leq \frac{C_2}{m^s},$$

where C_2 is a constant and the exponent s depends only on the regularity (smoothness) of the solution u. If u is smooth enough and satisfies certain boundary conditions, the spectral method has the spectral accuracy [25].

In this paper, we use the R-G method to solve the Poisson problem (2.1) in a special setting, i.e. we only have the information of $f(\boldsymbol{x})$ at the n sample points $\boldsymbol{x}_i$ $(i=1,\cdots,n)$. In this situation, the integral on the right hand side (r.h.s.) of R-G equation (2.6) is hard to be computed exactly, so we need to compute it with the proper numerical method. For higher dimensional case, it is known the Monte Carlo (MC) integration [23] may be the only viable approach. Then replacing the integral on the r.h.s. of (2.6) with the form of MC integral formula, we obtain

$$\sum_{k=1}^{m} c_k a(\phi_k,\phi_j) = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i)\phi_j(\boldsymbol{x}_i), \quad j=1,2,\cdots,m. \tag{2.8}$$

In fact, if we use the Gaussian quadrature rule to compute the integral on the r.h.s. of Eq. (2.6), we still have the similar form as (2.8), except that $1/n$ is replaced by the corresponding Gaussian quadrature weights w_i. Because of the inaccuracy of the right-hand side integral, Eq. (2.8) is actually different from the traditional R-G method. However, we can see clearly the bias of the R-G method under this special setting in the later numerical experiments.

2.3 DNN method

We now introduce the DNN method. The L-layer neural network is denoted by

$$u_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}^{[L-1]}\sigma\circ(\boldsymbol{W}^{[L-2]}\sigma\circ(\cdots(\boldsymbol{W}^{[0]}\boldsymbol{x}+\boldsymbol{b}^{[0]})\cdots)+\boldsymbol{b}^{[L-2]})+\boldsymbol{b}^{[L-1]}, \tag{2.9}$$

where $\boldsymbol{W}^{[l]} \in \mathbb{R}^{m_{l+1}\times m_l}$, $\boldsymbol{b}^{[l]} = \mathbb{R}^{m_{l+1}}$, $m_0 = d$, $m_L = 1$, σ is a scalar function and "$\circ$" means entry-wise operation. We denote the set of parameters by

$$\boldsymbol{\theta} = (\boldsymbol{W}^{[0]},\boldsymbol{W}^{[1]},\cdots,\boldsymbol{W}^{[L-1]},\boldsymbol{b}^{[0]},\boldsymbol{b}^{[1]},\cdots,\boldsymbol{b}^{[L-1]}),$$

and an entry of $\boldsymbol{W}^{[l]}$ by $\boldsymbol{W}^{[l]}_{ij}$.

Particularly, the one-hidden layer DNN with activation function σ is given as

$$u_\theta(\boldsymbol{x}) = \sum_{k=1}^{m} c_k\sigma(w_k\cdot\boldsymbol{x}+b_k)+\alpha, \tag{2.10}$$

where $w_k \in \mathbb{R}^d, c_k, b_k, \alpha \in \mathbb{R}$ are parameters. If we denote $\sigma(w_k \cdot \boldsymbol{x} + b_k) = \phi_k(\boldsymbol{x})$ and set $\alpha = 0$, then we obtain the similar form to R-G solution (2.5), i.e.,

$$u_\theta(\boldsymbol{x}) = \sum_{k=1}^{m} c_k \phi_k(\boldsymbol{x}). \tag{2.11}$$

The difference between the expressions of the solutions of these two methods is that the basis functions of the R-G solution are known, while the bases of the DNN solution are unknown, and need to be obtained together with the coefficients through the gradient descent algorithm with a loss function.

The loss function corresponding to problem (2.1) is given by

$$L_0(u_\theta, f) = \frac{1}{n} \sum_{i=1}^{n} (\Delta u_\theta(\boldsymbol{x}_i) + f(\boldsymbol{x}_i))^2 + \beta \int_{\partial\Omega} u_\theta(\boldsymbol{x})^2 ds, \tag{2.12}$$

or a variation form [10]

$$L_1(u_\theta, f) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{2} |\nabla_x u_\theta(\boldsymbol{x}_i)|^2 - f(\boldsymbol{x}_i) u_\theta(\boldsymbol{x}_i) \right) + \beta \int_{\partial\Omega} u_\theta(\boldsymbol{x})^2 ds, \tag{2.13}$$

where the last term is for the boundary condition and β is a hyper-parameter.

2.4 The connection between R-G method and DNN method

Here we further discuss the connection between R-G method and DNN method. If the DNN solution space is $U_h = \text{span}\{\phi_1, \phi_2, \cdots, \phi_m\}$, i.e., (2.11) holds on, then the loss function (2.13) can be written as

$$L_1(\boldsymbol{c}, \beta) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{2} |\sum_{k=1}^{m} c_k \nabla_x \phi_k(\boldsymbol{x}_i)|^2 - f(\boldsymbol{x}_i) \sum_{k=1}^{m} c_k \phi_k(\boldsymbol{x}_i) \right) + \beta \int_{\partial\Omega} \left(\sum_{k=1}^{m} c_k \phi_k(\boldsymbol{x}) \right)^2 ds. \tag{2.14}$$

The minimum point of $L_1(\boldsymbol{c}, \beta)$ satisfies the first-order necessary condition

$$\begin{cases} \dfrac{\partial L}{\partial c_j}(\boldsymbol{c}, \beta) = 0, \qquad j = 1, \cdots, m, \\ \dfrac{\partial L}{\partial \beta}(\boldsymbol{c}, \beta) = 0. \end{cases} \tag{2.15}$$

After simple calculations, one has

$$\begin{cases} \dfrac{1}{n} \displaystyle\sum_{i=1}^{n} \left(\sum_{k=1}^{m} c_k \nabla_x \phi_k(\boldsymbol{x}_i) \cdot \nabla_x \phi_j(\boldsymbol{x}_i) - f(\boldsymbol{x}_i) \phi_j(\boldsymbol{x}_i) \right) \\ \qquad + \beta \int_{\partial\Omega} 2 \left(\sum_{k=1}^{m} c_k \phi_k(\boldsymbol{x}) \right)^2 \phi_j(x) ds = 0, \qquad j = 1, \cdots, m, \\ \displaystyle\int_{\partial\Omega} \left(\sum_{k=1}^{m} c_k \phi_k(\boldsymbol{x}) \right)^2 ds = 0. \end{cases}$$

Since the basis functions in R-G method belong to space $H_0^1(\Omega)$, i.e., $\phi_k(\boldsymbol{x})_{|\partial\Omega}=0$, one obtains following algebraic system

$$\sum_{k=1}^{m} c_k \left(\frac{1}{n} \sum_{i=1}^{n} \nabla_x \phi_k(\boldsymbol{x}_i) \cdot \nabla_x \phi_j(\boldsymbol{x}_i) \right) = \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}_i) \phi_j(\boldsymbol{x}_i), \quad j=1,\cdots,m.$$

It is obvious that the above system is the system (2.8) except that the integral in the left-hand side of Eq. (2.8) is approximated by MC integration using the given n sample points. Therefore, the minimizing of the loss function (2.13) is equivalent to solving the approximated (2.8) where the integral in left hand is replaced by MC integration.

2.5 Frequency Principle

In this section, we illustrate and introduce a rigorous definition of the F-Principle.

We begin with considering a two-layer neural network, following [19, 34],

$$u(\boldsymbol{x},\boldsymbol{\theta}) = \sum_{j=1}^{m} a_j \sigma\left(\boldsymbol{w}_j^{\intercal} \boldsymbol{x} - |\boldsymbol{w}_j| c_j \right), \tag{2.16}$$

where $\boldsymbol{w}_j, \boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{\theta} = (\boldsymbol{a}^{\intercal}, \boldsymbol{w}_1^{\intercal}, \cdots, \boldsymbol{w}_m^{\intercal}, \boldsymbol{c}^{\intercal})^{\intercal}$, $\boldsymbol{a}, \boldsymbol{c} \in \mathbb{R}^m$ and $\boldsymbol{W} = (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m)^{\intercal} \in \mathbb{R}^{m\times d}$, and $\sigma(z) = \max(z, 0)$ ($z \in \mathbb{R}$) is the activation function of ReLU. Note that this two-layer model is slightly different from the model in (2.11) for easy calculation in [19, 34]. The target function is denoted by $U(\boldsymbol{x})$. The network is trained by mean-squared error (MSE) loss function

$$L = \int_{\mathbb{R}^d} \frac{1}{2} |u(\boldsymbol{x},\boldsymbol{\theta}) - U(\boldsymbol{x})|^2 \rho(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \tag{2.17}$$

where $\rho(\boldsymbol{x})$ is a probability density. Considering finite samples, we have

$$\rho(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \delta(\boldsymbol{x} - \boldsymbol{x}_i). \tag{2.18}$$

For any function g defined on $\mathbb{R}^d$, we use the following convention of the Fourier transform and its inverse:

$$\mathcal{F}[g](\boldsymbol{\xi}) = \int_{\mathbb{R}^d} g(\boldsymbol{x}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\xi}^{\intercal} \boldsymbol{x}} \mathrm{d}\boldsymbol{x}, \quad g(\boldsymbol{x}) = \int_{\mathbb{R}^d} \mathcal{F}[g](\boldsymbol{\xi}) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{x}^{\intercal} \boldsymbol{\xi}} \mathrm{d}\boldsymbol{\xi},$$

where $\boldsymbol{\xi} \in \mathbb{R}^d$ denotes the frequency.

The study in [19,34] shows that when the neuron number m is sufficient large, training the network in (2.16) with gradient flow dynamics at training time t is described by the following differential equation

$$\partial_t \mathcal{F}[u](\boldsymbol{\xi}) = -\Gamma(\boldsymbol{\xi}) \mathcal{F}[(u-U)\rho](\boldsymbol{\xi}) \tag{2.19}$$

with initial $\mathcal{F}[u_{\text{ini}}](\boldsymbol{\xi})$. This dynamics characterizes the evolution of each frequency. The convergence rate w.r.t. frequency depends on $\Gamma(\boldsymbol{\xi})$ and also affected by other frequencies due to the discrete sample distribution ρ. Consist with the supervised learning, the steady state of this dynamics is that DNN output equals to the target function at samples points.

The long time solution of (2.19) is equivalent to solve the following optimization problem

$$\min_{u-u_{\text{ini}}\in F_\gamma} \int_{\mathbb{R}^d} \Gamma^{-1}(\boldsymbol{\xi})|\mathcal{F}[u-u_{\text{ini}}](\boldsymbol{\xi})|^2 \mathrm{d}\boldsymbol{\xi}, \tag{2.20}$$

$$\text{s.t.} \quad u(\boldsymbol{x}_i)=U(\boldsymbol{x}_i) \quad \text{for} \quad i=1,\cdots,n, \tag{2.21}$$

where

$$\Gamma(\boldsymbol{\xi})=\frac{\frac{1}{m}\sum_{j=1}^m\left(\|\boldsymbol{w}_j(0)\|^2+a_j(0)^2\right)}{\|\boldsymbol{\xi}\|^{d+3}}+\frac{4\pi^2\frac{1}{m}\sum_{j=1}^m\left(\|\boldsymbol{w}_j(0)\|^2 a_i(0)^2\right)}{\|\boldsymbol{\xi}\|^{d+1}}, \tag{2.22}$$

here $\|\cdot\|$ represents the L^2-norm, $w_j(0)$ and $a_j(0)$ represent initial parameters before training, and

$$F_\gamma=\left\{u\Big|\int_{\mathbb{R}^d}\Gamma^{-1}(\boldsymbol{\xi})|\mathcal{F}[u](\boldsymbol{\xi})|^2\mathrm{d}\boldsymbol{\xi}<\infty\right\}. \tag{2.23}$$

Since $\Gamma(\boldsymbol{\xi})$ monotonically decreases with $\boldsymbol{\xi}$, the gradient flow in (2.19) rigorously defines the F-Principle, i.e., low frequency converges faster. The minimization in (2.20) clearly shows that the DNN has an implicit bias in addition to the sample constraint in (2.21). As $(\Gamma(\boldsymbol{\xi}))^{-1}$ monotonically increases with $\boldsymbol{\xi}$, the optimization problem prefers to choose a function that has less high frequency components, which explicates the implicit bias of the F-Principle — DNN prefers low frequency [32, 33].

Therefore, general DNN-based algorithms often encounter a high-frequency curse of slowly learning high-frequency information, for example, in solving multi-scale PDEs. Then, a series of algorithms, inspired by F-Principle, are developed to overcome the high-frequency curse of general DNN framework [1, 3, 4, 14, 15, 17, 27, 29, 30, 32].

3 Main results

3.1 R-G method in solving PDE

In the classical case, $f(\boldsymbol{x})$ is a given function, so we can compute exactly the integral on the r.h.s. of R-G equation (2.6). As the number of basis functions m approaches infinity, the numerical solution obtained by R-G method (2.6) approximates the exact solution of problem (2.1). It is interesting to ask if we only have the information of f at the finite n points, what could happen to numerical solution obtained by (2.8) when $m\to\infty$?

Fixing the number of sample points n, we study the property of the solution of the numerical method (2.8). We have the following theorem.

Theorem 3.1. *When $m\to\infty$, the numerical method* (2.8) *is solving the problem*

$$\begin{cases} -\Delta u(\boldsymbol{x})=\dfrac{1}{n}\displaystyle\sum_{i=1}^{n}\delta(\boldsymbol{x}-\boldsymbol{x}_i)f(\boldsymbol{x}_i), & \boldsymbol{x}\in\Omega, \\ u(\boldsymbol{x})=0, & \boldsymbol{x}\in\partial\Omega, \end{cases} \tag{3.1}$$

in the sense of generalized function, where $\delta(x)$ represents the Dirac delta function.

Proof. According to the filtering property of delta function, the formula for the integral on the right hand side (r.h.s.) of Eq. (2.8) is the exact integral between the function of the r.h.s. of Eq. (3.1) and the basis function $\phi_j(x)$, i.e.,

$$\frac{1}{n}\sum_{i=1}^{n}f(\boldsymbol{x}_i)\phi_j(\boldsymbol{x}_i)=\int_\Omega\frac{1}{n}\sum_{i=1}^{n}f(\boldsymbol{x}_i)\delta(\boldsymbol{x}-\boldsymbol{x}_i)\phi_j(\boldsymbol{x})d\boldsymbol{x},\quad j=1,2,\cdots.$$

Note that $\delta(x):\mathbb{R}^d\to\mathbb{R}$ is a continuous linear functional. According to the error estimation theory [2, 25], the finite dimensional system (2.8) evidently approximates the problem (3.1) when $m\to\infty$. □

Remark 3.1. For the 1-D case, the analytic solution to problem (3.1) defined in $[a,b]$ can be given as a piecewise linear function, namely

$$u(x)=\frac{1}{n}\sum_{i=1}^{n}f(x_i)(b-x_i)\frac{x-a}{b-a}-\frac{1}{n}\sum_{i=1}^{n}f(x_i)(x-x_i)H(x-x_i), \tag{3.2}$$

where $H(x)$ is the Heaviside step function

$$H(x)=\begin{cases} 0, & x<0, \\ 1, & x\geq 0. \end{cases}$$

For the 2-D case, [21] gives the exact solution in $[0,a]\times[0,b]$ by Green's function

$$u(x,y)=\frac{4}{nab}\sum_{i=1}^{n}f_i\sum_{k=1}^{\infty}\sum_{l=1}^{\infty}\frac{\sin(p_kx)\sin(q_ly)\sin(p_kx_i)\sin(q_ly_i)}{p_k^2+q_l^2}. \tag{3.3}$$

where $f_i=f(x_i,y_i)$, $p_k=\pi k/a$, $q_l=\pi l/b$. We can prove that this series diverge at the sampling point (x_i,y_i) $(i=1,2,\cdots,n)$ and converge at other points. Therefore, the 2-D exact solution $u(x,y)$ is highly singular.

3.2 Numerical experiments

Although R-G method and DNN method can be equivalent to each other in the sense of approximation, in this section, we present three examples to investigate the difference

between the solution obtained by the R-G method and the one obtained by the DNN with gradient descent optimization. We first consider the following 1-D Poisson problem

$$\begin{cases} -u''(x)=f(x), & x\in(-1,1), \\ u(-1)=u(1)=0, \end{cases} \tag{3.4}$$

where we only know what the value of $f(x)$ is at n points, i.e. $f(x_i)$ $(i=1,2,\cdots,n)$. In experiments, $f(x_i)$ are sampled from the function

$$f(x)=-(4x^3-6x)\exp(-x^2). \tag{3.5}$$

Note that there are infinite possible functions that can have the same values at the selected n positions as $f(x)$, therefore, there is not an exact solution, thus, we do not plot exact solutions for comparison in the following.

Example 3.1. Fixing the number of sampling points $n=5$, we use R-G method and DNN method to solve the problem (3.4), respectively. The reason why we choose fewer sample points here is that in this situation we can investigate the property of the solution more clearly. The results are shown as follows.

R-G method. First, we use R-G method to solve the problem (3.4), specially the spectral method with the Fourier basis function given as

$$\phi_k(x)=\sin(k\pi x), \quad k=1,2,\cdots,m.$$

We set the number of basis functions m=5,10,50,500, respectively. Fig. 2 plots the numerical solutions obtained by R-G method. The solutions (3.2) of problem (3.1) with boundary conditions and $f(x)$ given in (3.4) and (3.5) are presented in Fig. 2. One can see that the R-G solution approximates the piecewise linear function (3.2) when $m\to\infty$. This result is consistent with the property of solution analyzed in Theorem 3.1.

DNN method. For a better comparison with R-G method, we choose the activation function by $\sin(x)$ in DNN with one hidden layer. And the number of neurons are taken as $m=5,10,50,500$. The loss function (2.12) is selected with the parameter $\beta=10$. We reduce the loss to an order of 1e-4, and take learning rate by 1e-4. And 1000 test points are used to plot the figures. The DNN solutions are shown in Fig. 3, in which we observe that the DNN solutions are always smooth even when m is very large.

Example 3.2. In this example, we use the ReLU function as the basis function in R-G method and the activation function in DNN method to repeat the experiments in Example 1. Here we randomly choose another 10 sampling points.

Since the linear finite element function $\phi_j(x)$ can be expressed by ReLU functions for one dimensional case, namely,

$$\phi_j(x)=\frac{1}{h_{j-1}}\mathrm{ReLU}(x-x_{j-1})-\Big(\frac{1}{h_{j-1}}+\frac{1}{h_j}\Big)\mathrm{ReLU}(x-x_j)+\frac{1}{h_j}\mathrm{ReLU}(x-x_{j+1}),$$

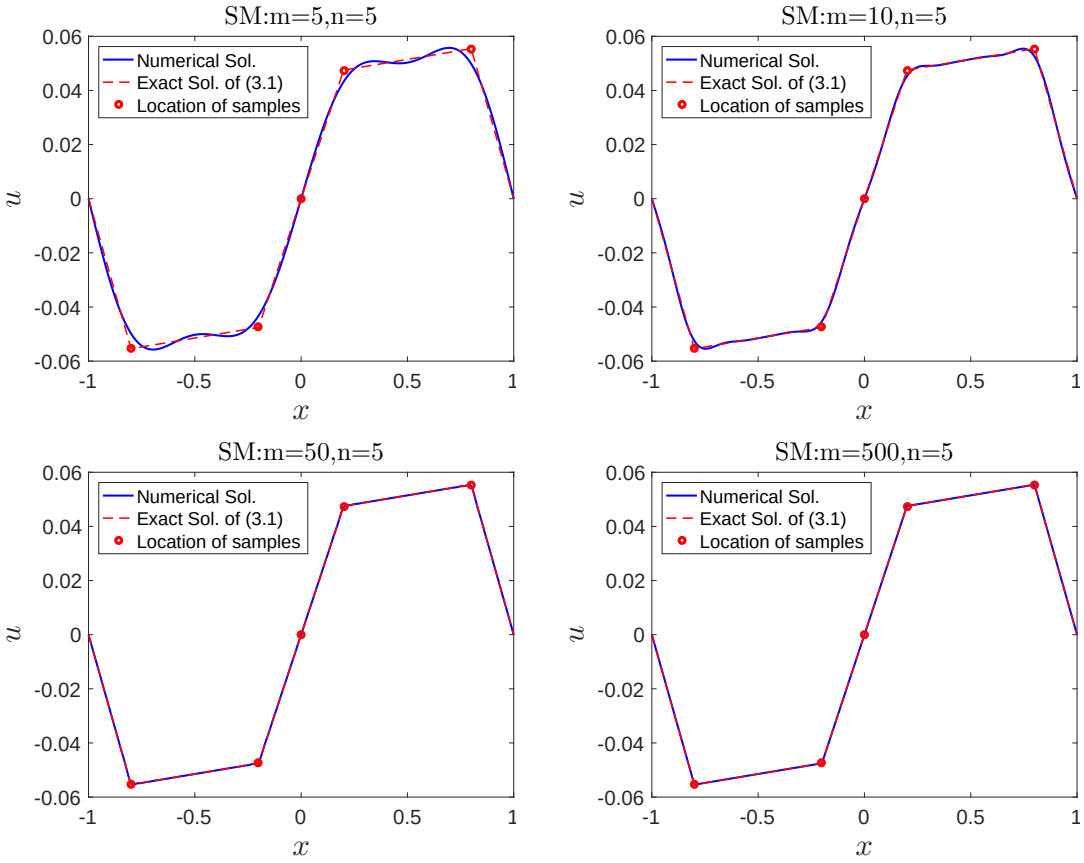

Figure 2: (Example 3.1): Numerical solutions in SM with 5 sampling points.

where $h_j = x_{j+1} - x_j$, we can use ReLU as the basis function for R-G method. For convenience, we just use the linear finite element function $\phi_j(x)$ instead of ReLU function as the basis function. Fig. 4 shows that the FEM solution undoubtedly approximates the piecewise linear solution (3.2).

In DNN method, we choose the number of neurons $m = 5,10,50,500$, respectively. And we use the variational form of the loss function (2.13) because of the second order derivative of ReLU function is always zero. Fig. 5 shows that the DNN learns the data as a relatively smoother function than the R-G method.

Example 3.3. We consider the 2-D case

$$\begin{cases} -\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in (0,1)^2, \\ u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial(0,1)^2, \end{cases}$$

where $\boldsymbol{x} = (x,y)$ and the values of f at n points sampled from the function $f(\boldsymbol{x}) := f(x,y) = 2\pi^2 \sin(\pi x)\sin(\pi y)$. We fix the number of sample points $n = 5^2$. The sampling points

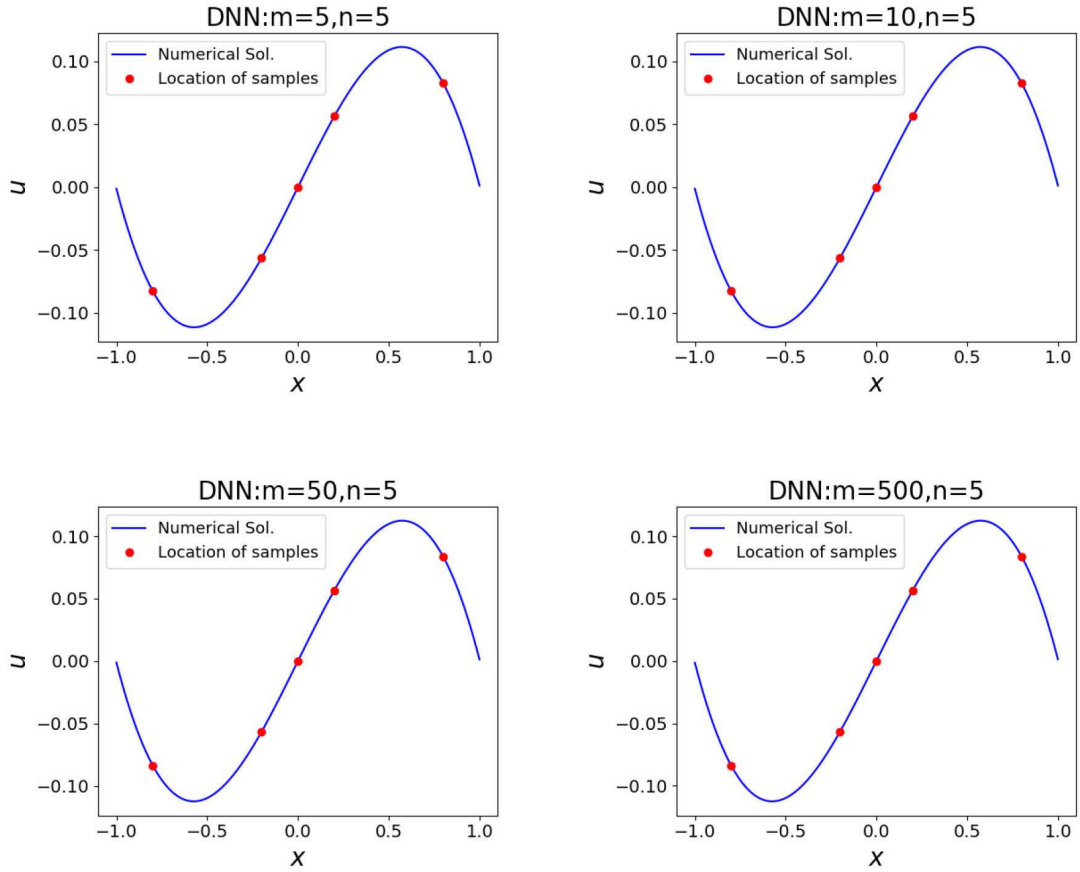

Figure 3: (Example 3.1): Numerical solutions in DNN method with 5 sampling points.

in the x direction and the y direction are both at $x_h = [0.1,0.25,0.5,0.8,0.9]$. We test the solution with the number of basis $m = 5,50,100,200$, respectively. Fig. 6 plots the R-G solutions with Legendre basis and piecewise linear basis function. It can be seen that the numerical solution is a function with strong singularity. Fig. 7 shows the profile of R-G solutions at $y = 0.5$ for various m, in which we can see that the values of numerical solutions at the sampling points get larger and larger with the increase of m. However, Fig. 8 shows the DNN solutions are stable without singularity for large m.

4 Discussion of DNN method in solving PDE

4.1 Frequency principle

DNNs are widely used in solving PDEs, especially for high-dimensional problems. The optimizing the loss functions in Eqs. (2.12) and (2.13) are equivalent to solving (2.6) ex-

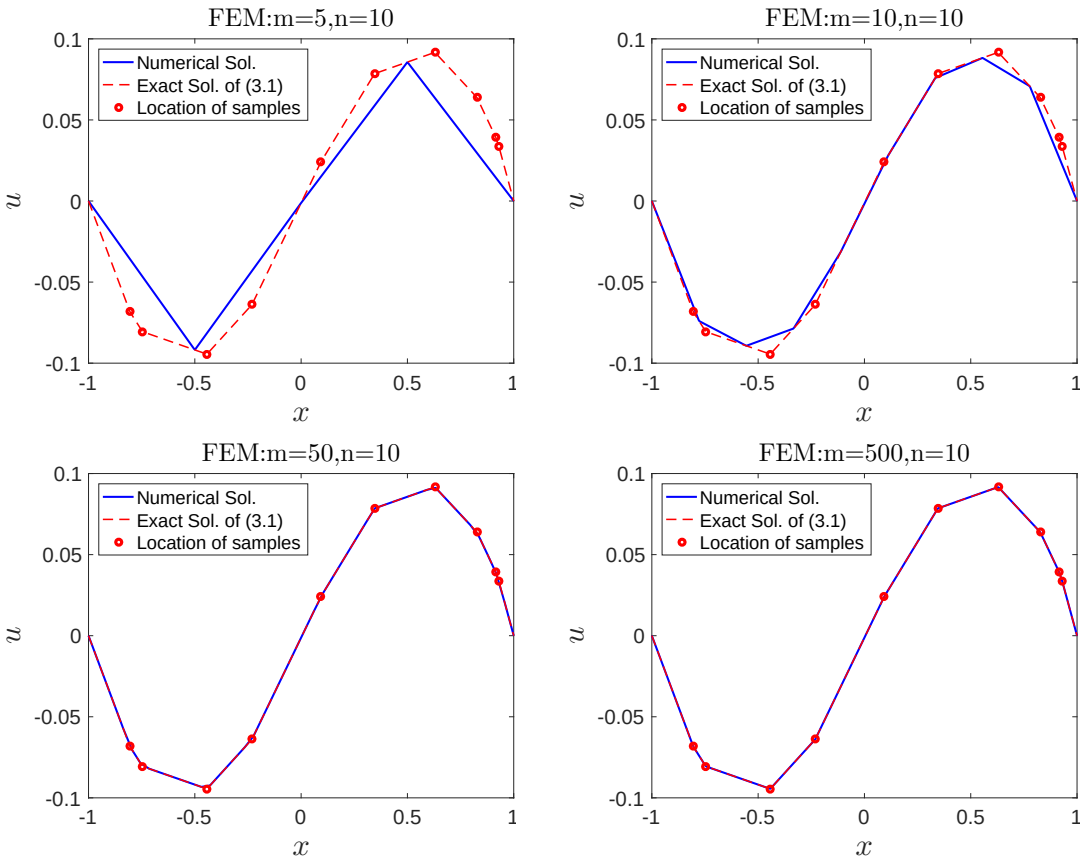

Figure 4: (Example 3.2): Numerical solutions in FEM with 10 sampling points.

cept that the bases in (2.12) and (2.13) are adaptive. In addition, the DNN problem is optimized by (stochastic) gradient descent. The experiments in the previous section have shown that when the number of bases goes to infinity, DNN methods solve (2.1) by a relatively smooth and stable function compared with the one obtained by Theorem 3.1. We now utilize the F-Principle to understand what leads to the smoothness.

For two-layer wide DNNs with $d=1$, the two terms of $\Gamma(\boldsymbol{\xi})$ in the minimization problem of (2.20) yield different fitting results. Note that $\min \int_R ||\boldsymbol{\xi}||^{-2}|\mathcal{F}[h](\boldsymbol{\xi})|^2 \mathrm{d}\boldsymbol{\xi}$ leads to a piecewise linear function, while $\min \int_R ||\boldsymbol{\xi}||^{-4}|\mathcal{F}[h](\boldsymbol{\xi})|^2 \mathrm{d}\boldsymbol{\xi}$ leads to a cubic spline. Since the DNN is a combination of both terms, therefore, the DNN would yield to a much smoother function than the piecewise linear function. For a general DNN, the coefficient $\Gamma(\boldsymbol{\xi})$ in (2.19) cannot be obtained exactly, however, the monotonically decreasing property of $\Gamma(\boldsymbol{\xi})$ with respect to $\boldsymbol{\xi}$ can be postulated based on the F-Principle. Theoretical works [19, 20, 24, 32, 34] have shown that the regularity of DNN converts into the decay rate of a loss function in the frequency domain. As the commonly used activation functions decay with a certain rate in the frequency domain due to their regularity, DNNs,

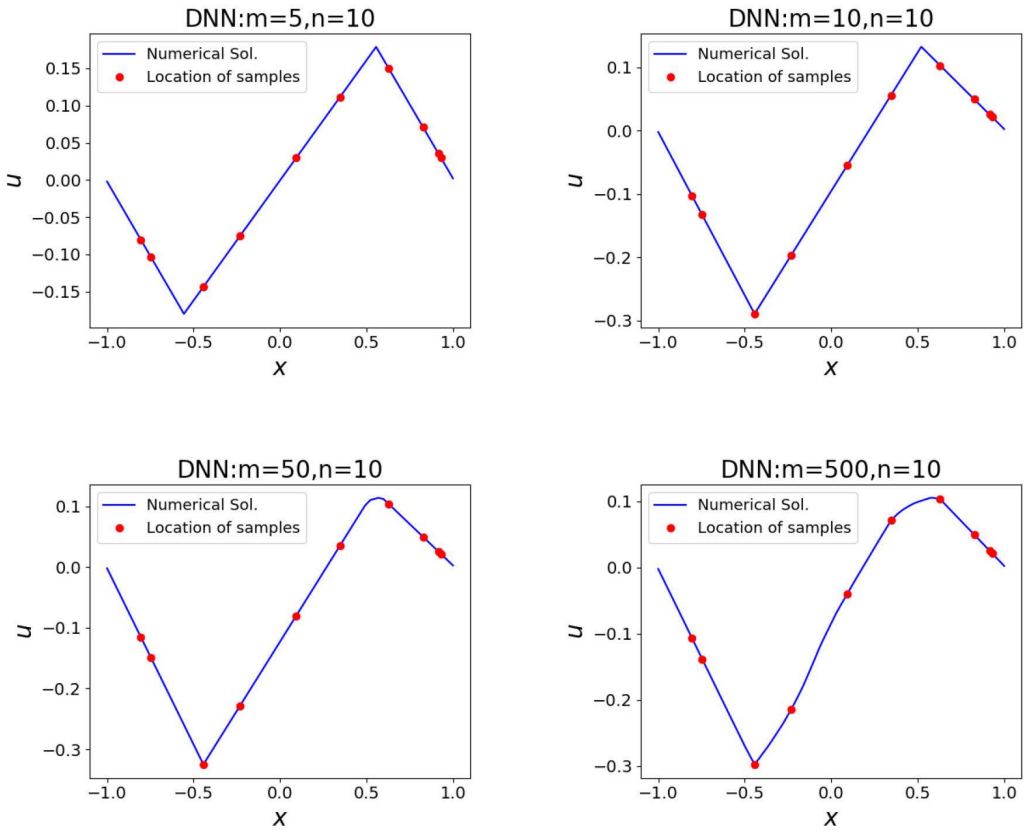

Figure 5: (Example 3.2): Numerical solutions in DNN method with ReLU activation function and 10 sampling points.

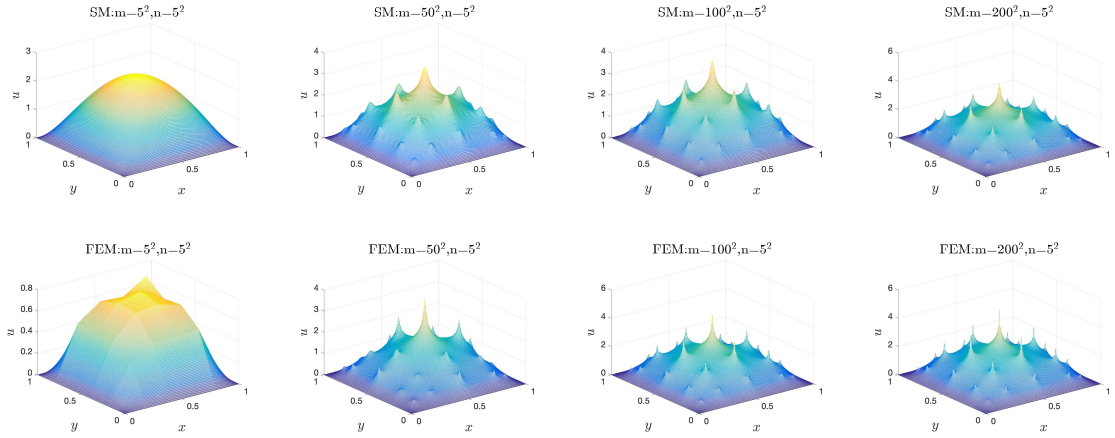

Figure 6: (Example 3.3): R-G solutions with different m. The basis functions for the first and the second row are Legendre basis function and piecewise linear basis function, respectively.

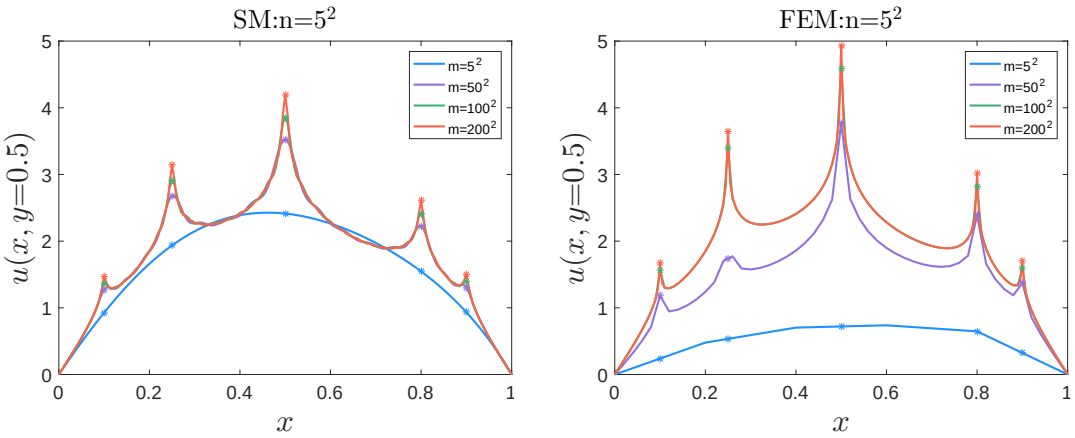

Figure 7: (Example 3.3): Profile of R-G solutions with different m.

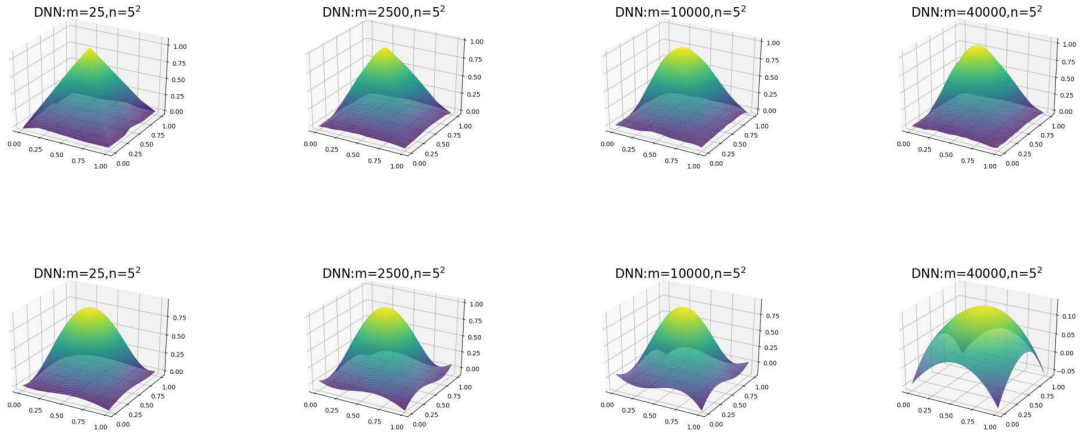

Figure 8: (Example 3.3): DNN solutions with different m. The activation functions for the first and the second row are $\mathrm{ReLU}(x)$ and $\sin(x)$, respectively.

with common loss functions, often show a low-frequency bias in the learning.

The key to F-Principle is the regularity of the activation function. Imposing high priority on high frequency can alleviate the effect of the F-Principle and sometimes an anti-F-Principle can be observed. For example, consider a loss function, containing the gradient of the DNN output w.r.t. the input

$$L_S(\theta) = \sum_{x \in S} (\nabla u(\theta, x) - \nabla U(x))^2. \tag{4.1}$$

The Fourier transform of $\nabla u(\theta, x)$ is $\xi \mathcal{F}[u(\theta, x)](\xi)$, that is, a higher frequency would have a higher weight in the loss function. Then, the convergence of different frequencies depends on the competition between the activation regularity and the loss function. If

the loss function endorses more priority for the high frequency to compensate the low-priority induced by the activation function, a high frequency may converges faster. Some analysis and numerical experiments can also be found in [9, 18].

Taken together, the above analysis qualitatively explains the DNN with sinusoidal activation function learns smoother function than that with ReLU activation function.

4.2 Spectral collocation method

Finally, we briefly compare the spectral collocation method and DNN method in solving PDEs. The spectral collocation method is similar to a random feature model but with orthogonal polynomials as basis functions. A DNN with infinite width and proper initial scaling is also similar to a random feature model [8] but with activation functions (e.g., ReLU or tanh) as basis functions.

We resolve the Example 3.1 in Section 3.2 using the spectral collocation method with the Legendre basis. Given any set of distinct collocation points $\{x_i\}_{i=1}^n$ on $[-1,1]$ in ascending order with $x_1=-1$ and $x_n=1$. Let $u_m=\sum_{j=1}^m w_j\phi_j(x)$. The basis function $\{\phi_j\}$ satisfies boundary conditions. The spectral collocation is essentially minimizing the loss function (2.12) with $\beta=0$, i.e.,

$$\min_{\{w_j\}_{j=1}^m} \frac{1}{n}\sum_{i=1}^n (\Delta u_m(\boldsymbol{x}_i)+f(\boldsymbol{x}_i))^2. \tag{4.2}$$

Fig. 9 shows the numerical solutions with different m. When $m=n$, the spectral collocation method performs well since the problem is well-posed. When $m>n$ (the over-parameterized case), the problem has infinite solutions. To make a fair comparison with the DNN method, the gradient descent method is used to solve problem (4.2). One can see that the collocation solution has the low regularity like the Ritz-Galerkin solution. The underlying mechanism of the spectral collocation method in the over-parameterized regime is yet to be clarified in the future work. These empirical studies indicate that the DNN method possesses special implicit bias towards low frequencies compared to the traditional numerical method.

5 Conclusion

This paper compares the different behaviors of Ritz-Galerkin method and DNN method through solving PDEs to better understand the working principle of DNNs. We consider a particular Poisson problem (2.1), where the r.h.s. term f is a discrete function. We analyze why the two numerical methods behave differently in theory. R-G method deals with the discrete f as the linear combination of Dirac delta functions, while DNN methods implicitly bias towards functions with more low-frequency components to interpolate the discrete sampling points due to the F-Principle. Furthermore, from the numerical experiments, as the number of bases increases, one can see that the solutions obtained

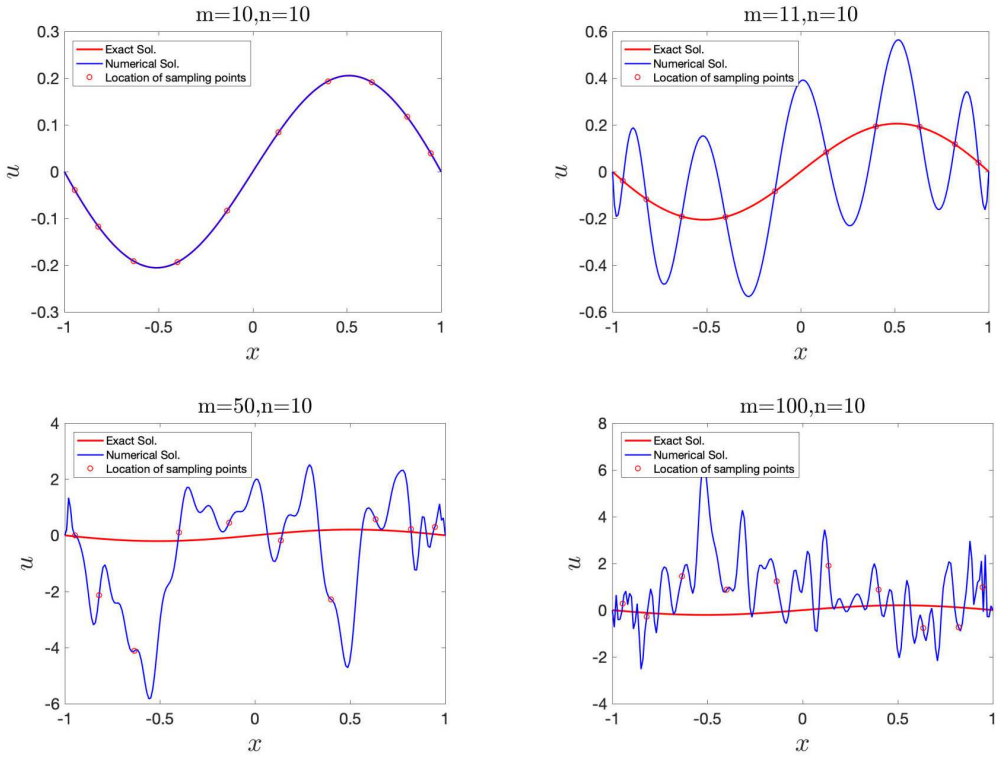

Figure 9: Numerical solutions obtained by spectral collocation method when $m \geq n$.

by R-G method approximate piecewise linear functions for 1d case and singular function for the 2d case, regardless of the basis function, but the solutions obtained by DNN method are smoother for 1d case and stable for the 2d case. In conclusion, based on the theoretical and numerical study in comparison to traditional methods in solving PDEs, DNN method possesses special implicit bias towards low frequencies, which leads to a well-behaved solution even in a heavily over-parameterized setting.

Acknowledgments

Zhiqin Xu is supported by National Key R&D Program of China (2019YFA0709503), Shanghai Sailing Program, Natural Science Foundation of Shanghai (20ZR1429000), NSFC 62002221. Jiwei Zhang is partially supported by NSFC under grant Nos. 11771035 and 12171376, 2020-JCJQ- ZD-029 and NSAF U1930402. Yaoyu Zhang is supported by Shanghai Municipal of Science and Technology Project Grant No. 20JC1419500. Zhiqin Xu and Yaoyu Zhang are also supported by Shanghai Municipal of Science and Technology Major Project NO. 2021SHZDZX0102, and HPC of School of Mathematical Sciences and Student Innovation Center at Shanghai Jiao Tong University.

References

[1] Simon Biland, Vinicius C. Azevedo, Byungsoo Kim, and Barbara Solenthaler. Frequency-aware reconstruction of fluid simulations with generative networks. In *Eurographics 2020 – Short Papers*. The Eurographics Association, 2020.

[2] Susanne Brenner and Ridgway Scott. *The Mathematical Theory of Finite Element Methods*, volume 15. Springer Science & Business Media, 2007.

[3] Wei Cai, Xiaoguang Li, and Lizuo Liu. A phase shift deep neural network for high frequency approximation and wave problems. *SIAM Journal on Scientific Computing*, 42(5):A3285–A3312, 2020.

[4] Wei Cai and Zhi-Qin John Xu. Multi-scale deep neural networks for solving high dimensional PDEs. *arXiv preprint arXiv:1910.11710*, 2019.

[5] Zhiqiang Cai, Jingshuang Chen, Min Liu, and Xinyu Liu. Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs. *Journal of Computational Physics*, 420:109707, 2020.

[6] Weinan E. Machine learning and computational mathematics. *Communications in Computational Physics*, 28(5):1639–1670, 2020.

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

[8] Weinan E, Chao Ma, and Lei Wu. A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics. *Science China Mathematics*, 63(7):1235–1258, 2020.

[9] Weinan E, Chao Ma, and Lei Wu. Machine learning from a continuous viewpoint. *Science China Mathematics*, 63(11):2233–2266, 2020.

[10] Weinan E and Bing Yu. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[11] Aidan Hamilton, Tuyen Tran, Maricela B. Mckay, Benjamin Quiring, and Panayot S. Vassilevski. DNN approximation of nonlinear finite element equations. Technical report, Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), 2019.

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

[13] Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. ReLU deep neural networks and linear finite elements. *Journal of Computational Mathematics*, 2019.

[14] Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.

[15] Xi-An Li, Zhi-Qin John Xu, and Lei Zhang. A multi-scale DNN algorithm for nonlinear elliptic equations with multiple scales. *Communications in Computational Physics*, 28(5):1886–1906, 2020.

[16] Yulei Liao and Pingbing Ming. Deep Nitsche method: Deep Ritz method with essential boundary conditions. *Communications in Computational Physics*, 29(5):1365–1384, 2021.

[17] Ziqi Liu, Wei Cai, and Zhi-Qin John Xu. Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains. *Communications in Computational Physics*, 28(5):1970–2001, 2020.

[18] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.
[19] Tao Luo, Zheng Ma, Zhi-Qin John Xu, and Yaoyu Zhang. On the exact computation of linear frequency principle dynamics and its generalization. *arXiv preprint arXiv:2010.08153*, 2020.
[20] Tao Luo, Zheng Ma, Zhi-Qin John Xu, and Yaoyu Zhang. Theory of the frequency principle for general deep neural networks. *CSIAM Transactions on Applied Mathematics*, 2(3):484–507, 2021.
[21] Andrei D. Polyanin and Vladimir E. Nazaikinskii. *Handbook of Linear Partial Differential Equations for Engineers and Scientists*. CRC Press, 2015.
[22] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
[23] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer-Verlag, New York, second edition, 2004.
[24] Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In *Advances in Neural Information Processing Systems*, pages 4763–4772, 2019.
[25] Jie Shen, Tao Tang, and Li-Lian Wang. *Spectral Methods: Algorithms, Analysis and Applications*, volume 41 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2011.
[26] Jonathan W. Siegel and Jinchao Xu. Approximation rates for neural networks with general activation functions. *Neural Networks*, 128:313–321, 2020.
[27] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020.
[28] Vidar Thomée. *Galerkin Finite Element Methods for Parabolic Problems*, volume 25. Springer Science & Business Media, 2007.
[29] Bo Wang. Multi-scale deep neural network (MscaleDNN) methods for oscillatory stokes flows in complex domains. *Communications in Computational Physics*, 28(5):2139–2157, 2020.
[30] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021.
[31] Zhongjian Wang and Zhiwen Zhang. A mesh-free method for interface problems using the deep learning approach. *Journal of Computational Physics*, 400:108963, 2020.
[32] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 28(5):1746–1767, 2020.
[33] Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. *International Conference on Neural Information Processing*, pages 264–274, 2019.
[34] Yaoyu Zhang, Zhi-Qin John Xu, Tao Luo, and Zheng Ma. Explicitizing an implicit bias of the frequency principle in two-layer neural networks. *arXiv preprint arXiv:1905.10264*, 2019.