Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2020-0179

Vol. **28**, No. 5, pp. 1970-2001
November 2020

Multi-Scale Deep Neural Network (MscaleDNN) for Solving Poisson-Boltzmann Equation in Complex Domains

Ziqi Liu[1], Wei Cai[2] and Zhi-Qin John Xu[3,*]

[1] *Beijing Computational Science Research Center, Beijing, 100193, P.R. China.*
[2] *Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA.*
[3] *Institute of Natural Sciences, MOE-LSC, School of Mathematical Sciences and Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China.*

Received 11 September 2020; Accepted (in revised version) 25 September 2020

Summary. In this paper, we propose multi-scale deep neural networks (MscaleDNNs) using the idea of radial scaling in frequency domain and activation functions with compact support. The radial scaling converts the problem of approximation of high frequency contents of PDEs' solutions to a problem of learning about lower frequency functions, and the compact support activation functions facilitate the separation of frequency contents of the target function to be approximated by corresponding DNNs. As a result, the MscaleDNNs achieve fast uniform convergence over multiple scales. The proposed MscaleDNNs are shown to be superior to traditional fully connected DNNs and be an effective mesh-less numerical method for Poisson-Boltzmann equations with ample frequency contents over complex and singular domains.

AMS subject classifications: 35Q68, 65N99, 68T07

Key words: Deep neural network, Poisson-Boltzmann equation, multi-scale, frequency principle.

1 Introduction

Deep neural network (DNN) has found many applications beyond its traditional applications such as image classification and speech recognition into the arena of scientific computing [10–15, 17, 22, 24, 25]. However, to apply the commonly-used DNNs to computational science and engineering problems, we are faced with several challenges. The most prominent issue is that the DNN normally only handles data with low frequency

*Corresponding author. *Email addresses:* liuziqi@csrc.ac.cn (Z. Liu), cai@smu.edu (W. Cai), xuzhiqin@sjtu.edu.cn (Z.-Q. J. Xu)

content well, as shown by a Frequency Principle (F-Principle) that many DNNs learn the low frequency content of the data quickly with a good generalization error, but they are inadequate when high frequency data are involved [21, 27, 28]. The fast convergence behavior of low frequency has been recently studied rigorously in theory in [2, 6, 19, 30]. As a comparison, such a behavior of DNNs is the opposite of that of the popular multi-grid methods (MGM) for solving PDEs such as the Poisson-Boltzmann (PB) equation, where the convergence is achieved first in the high frequency spectrum of the solution due to the smoothing operations employed in the MGM. Considering the potential of DNNs in handling higher dimensional solutions and approximating functions without the need of a structured mesh as in traditional finite element or finite difference method, it is of great value to extend the capability of DNN as a mesh-less PDE solver. Therefore, it is imperative to improve the convergence of DNNs for solutions with fine structures as encountered in the electrostatic potentials of complex molecules.

The electrostatic interaction of bio-molecules with ionic solvents, governed by the Poisson-Boltzmann (PB) equation within the Debye-Huckel theory [3], plays an important role in many applications including drug design and the study of disease. However, due to the complex surface structure of the bio-molecules, usually represented by a bead model, it has been a long outstanding challenging to design efficient numerical method to handle the singular molecular surface, which is either the van der Waals (vdW) surface being the sum of overlapping vdW spheres or the solvent accessible surface (SAS) generated by rolling a small ball on the vdW surface [18], and the complex distribution of the electrostatic potential over the molecular surfaces. Tradition finite element [1] and finite difference methods [29] have faced difficulties in the costly mesh generation and expensive solution of the discretized linear system. Therefore, in this paper, we will propose and investigate multi-scale DNNs, termed MscaleDNN, with the goal of approximating both low and high frequency information of a solution uniformly and developing a mesh-less solver for PDEs such as the PB equations in domains with complex and singular geometries.

Different learning behaviors among different frequencies are common. Leveraging this difference in designing neural network structure can benefit the learning process. In the field of computer vision, a series of works, such as image recovery [9], super-resolution [20], or classification [26], have improved the learning performance, including the generalization and training speed, by utilizing the learning difference of different image frequencies. However, it should be noted that the frequency used in the computer vision tasks, is different from the response frequency of a mapping from the input (e.g., image) to the output (e.g., label), and the former refers to the frequency within an input (i.e. an image) with respect to the spatial locations inside the image. In this work, we address different response frequencies of the mapping from the input to the output. As demonstrated in the previous work [28], the low response frequency is learned much faster than the high frequency. The main idea of the MscaleDNN is to find a way to convert the learning or approximation of high frequency data to that of a low frequency one. Similar idea has been attempted in a previous work in the development of a phase

shift DNN (PhaseDNN) [4], where the high frequency component of the data was given a phase shift downward to a low frequency spectrum. The learning of the shifted data can be achieved with a small size DNN quickly, which was then shifted back (i.e., upward in frequency) to give approximation to the original high frequency data. The PhaseDNN has been shown to be very effective to handle highly oscillatory data from solutions of high frequency Helmholtz equations and functions of small dimensions. However, due to the number of phase shifts employed along each coordinate direction independently, the PhaseDNN will result in many small DNNs and a considerable computational cost even for three dimensional problems. Here, we will consider a different approach to achieve the conversion of high frequency to lower one, namely, with a radial partition of the Fourier space, a scaling down operation will be used to convert higher frequency spectrum to a low frequency one before the learning is carried out with a small-sized DNN. As the scaling operation only needs to be done along the radial direction in the Fourier space, this approach is easy to be implemented and gives an overall small number of DNNs, thus reducing the computational cost. In addition, borrowing the multi-resolution concept of wavelet approximation theory using compact scaling and wavelet functions [8], we will modify the traditional global activation functions to ones with compact support. The compact support of the activation functions with sufficient smoothness will give a localization in the frequency domain where the scaling operation will effectively produce DNNs to approximate different frequency contents of a PDE solution. As a previous study shows [23] that DNNs can approximate an intrinsically low dimensional function defined in a high dimensional space without the curse of dimensionality in terms of neuron number, provided it also has a sparse wavelet representation. The proposed compact supported activation functions, similar to scaling and wavelet functions in the wavelet theory, will show their scale resolution capability in the MscaleDNNs.

Two types of MscaleDNN architectures are proposed, investigated, and compared for their performances. After various experiments, we demonstrate that MscaleDNNs solves elliptic PDEs much faster and can achieve a much smaller generalization error, compared with normal fully connected networks with similar overall size. We will apply MscaleDNNs to solve variable coefficient elliptic equations, including those solutions with a broad range of frequencies and over different types of domains such as a ring-shaped domain and a cubic domain with multiple holes. Also, to test the potential of MscaleDNN for finding Poisson-Boltzmann electrostatic solvation energy in biomolecules, we apply MscaleDNN to solve elliptic equation with geometric singularities, such as cusps and self-intersecting surfaces in a molecular surface. These extensive experiments clearly demonstrate that the MscaleDNN is an efficient and easy-to-implement mesh-less PDE solver in complex domains.

The rest of the paper is organized as follows. In section 2, we will introduce frequency scaling to generate a MscaleDNN representation. Section 3 will present MscaleDNN structures with compact support activation functions. Section 4 will present a minimization approach through the Ritz energy for finding the solutions of elliptic PDEs and a minimization approach through a least squared error for fitting functions. In section 5,

we use two test problems to show the effectiveness of the proposed MscaleDNN over a normal fully connected DNN of same size. Next, numerical results of the solution of complex elliptic PDEs with complex domains by the proposed MscaleDNN will be given in Section 6. Finally, Section 7 gives a conclusion and some discussion for further work.

2 Frequency scaled DNNs and compact activation functions

In this section, we will first present a naive idea of how to use a frequency scaling in Fourier wave number space to reduce a high frequency learning problems for a function to a low frequency learning one for the DNN and will also point out the difficulties it may encounter as a practical algorithm.

Consider a band-limited function $f(\boldsymbol{x})$ $\boldsymbol{x}\in\mathbb{R}^d$, whose Fourier transform $\widehat{f}(\mathbf{k})$ has a compact support, i.e.,

$$\operatorname{supp}\widehat{f}(\mathbf{k})\subset B(K_{\max})=\{\mathbf{k}\in\mathbb{R}^d,\ |\mathbf{k}|\leq K_{\max}\}. \tag{2.1}$$

We will first partition the domain $B(K_{\max})$ as union of M concentric annulus with uniform or non-uniform width, e.g., for the case of uniform K_0-width

$$A_i=\{\mathbf{k}\in\mathbb{R}^d,\ (i-1)K_0\leq|\mathbf{k}|\leq iK_0\},\quad K_0=K_{\max}/M,\quad 1\leq i\leq M \tag{2.2}$$

so that

$$B(K_{\max})=\bigcup_{i=1}^{M}A_i. \tag{2.3}$$

Now, we can decompose the function $\widehat{f}(\mathbf{k})$ as follows

$$\widehat{f}(\mathbf{k})=\sum_{i=1}^{M}\chi_{A_i}(\mathbf{k})\widehat{f}(\mathbf{k})\triangleq\sum_{i=1}^{M}\widehat{f}_i(\mathbf{k}), \tag{2.4}$$

where χ_{A_i} is the indicator function of the set A_i and

$$\operatorname{supp}\widehat{f}_i(\mathbf{k})\subset A_i. \tag{2.5}$$

The decomposition in the Fourier space give a corresponding one in the physical space

$$f(\boldsymbol{x})=\sum_{i=1}^{M}f_i(\boldsymbol{x}), \tag{2.6}$$

where

$$f_i(\boldsymbol{x})=\mathcal{F}^{-1}[\widehat{f}_i(\mathbf{k})](\boldsymbol{x})=f(\boldsymbol{x})*\chi_{A_i}^{\vee}(\boldsymbol{x}), \tag{2.7}$$

and the inverse Fourier transform of $\chi_{A_i}(\mathbf{k})$ is called the frequency selection kernel [4] and can be computed analytically using Bessel functions

$$\chi^{\vee}_{A_i}(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}} \int_{A_i} e^{i\mathbf{k}\circ\mathbf{x}} dk. \tag{2.8}$$

From (2.5), we can apply a simple down-scaling to convert the high frequency region A_i to a low frequency region. Namely, we define a scaled version of $\widehat{f_i}(\mathbf{k})$ as

$$\widehat{f_i}^{(\text{scale})}(\mathbf{k}) = \widehat{f_i}(\alpha_i \mathbf{k}), \qquad \alpha_i > 1, \tag{2.9}$$

and, correspondingly in the physical space

$$f_i^{(\text{scale})}(\boldsymbol{x}) = f_i\Big(\frac{1}{\alpha_i}\boldsymbol{x}\Big), \tag{2.10}$$

or

$$f_i(\boldsymbol{x}) = f_i^{(\text{scale})}(\alpha_i \boldsymbol{x}). \tag{2.11}$$

We can see the low frequency spectrum of the scaled function $\widehat{f_i}^{(\text{scale})}(\mathbf{k})$ if α_i is chosen large enough, i.e.,

$$\text{supp}\,\widehat{f_i}^{(\text{scale})}(\mathbf{k}) \subset \Big\{\mathbf{k}\in\mathbb{R}^d,\ \frac{(i-1)K_0}{\alpha_i} \le |\mathbf{k}| \le \frac{iK_0}{\alpha_i}\Big\}. \tag{2.12}$$

Using the F-Principle of common DNNs [27], with iK_0/α_i being small, we can train a DNN $f_{\theta^{n_i}}(\boldsymbol{x})$, with θ^{n_i} denoting the DNN parameters, to learn $f_i^{(\text{scale})}(\boldsymbol{x})$ quickly

$$f_i^{(\text{scale})}(\boldsymbol{x}) \sim f_{\theta^{n_i}}(\boldsymbol{x}), \tag{2.13}$$

which gives an approximation to $f_i(\boldsymbol{x})$ immediately

$$f_i(\boldsymbol{x}) \sim f_{\theta^{n_i}}(\alpha_i \boldsymbol{x}) \tag{2.14}$$

and to $f(\boldsymbol{x})$ as well

$$f(\boldsymbol{x}) \sim \sum_{i=1}^{M} f_{\theta^{n_i}}(\alpha_i \boldsymbol{x}). \tag{2.15}$$

The difficulty of the above procedure used directly for approximating function and even more for finding a PDE solution is the need to compute the convolution in (2.7), which is computationally expensive for scattered data in the space, especially in higher dimensional problems. However, this framework will lay the structure for the multiscale DNN to be proposed next.

3 MscaleDNN structures

3.1 Activation function with compact support

In order to produce scale separation and identification capability of a MscaleDNN, we borrow the idea of compact scaling function in the wavelet theory [8], and consider the activation functions with compact support as well. Compared with the normal activation function $\mathrm{ReLU}(x)=\max\{0,x\}$, we will see activation functions with compact support are more effective in MscaleDNNs. Two possible activation functions are defined as follows

$$\mathrm{sReLU}(x)=\mathrm{ReLU}(-(x-1))\times\mathrm{ReLU}(x)=(x)_+(1-x)_+, \tag{3.1}$$

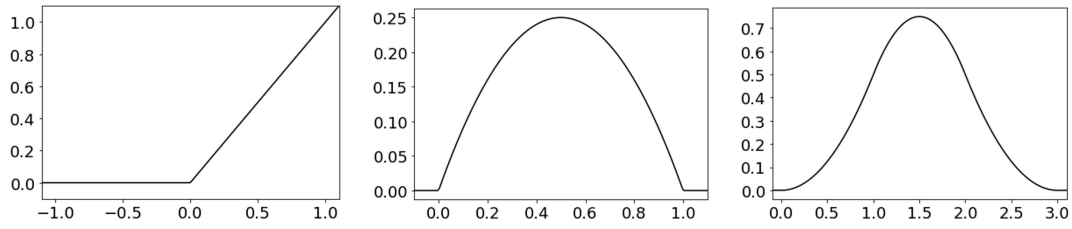
and the quadratic B-spline with first continuous derivative

$$\phi(x)=(x-0)_+^2-3(x-1)_+^2+3(x-2)_+^2-(x-3)_+^2, \tag{3.2}$$

where $x_+=\max\{x,0\}=\mathrm{ReLU}(x)$, and the latter has an alternative form,

$$\phi(x)=\mathrm{ReLU}(x)^2-3\mathrm{ReLU}(x-1)^2+3\mathrm{ReLU}(x-2)^2-\mathrm{ReLU}(x-3)^2. \tag{3.3}$$

All three activation functions are illustrated in spatial domain in Fig. 1 and the Fourier transforms of both $\mathrm{sReLU}(x)$ and $\phi(x)$ are illustrated in Fig. 2.



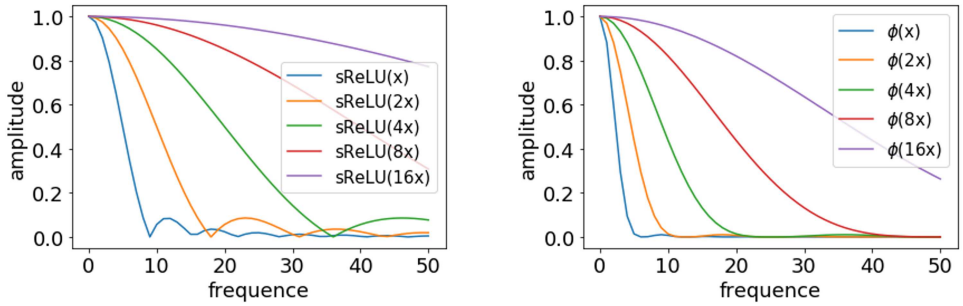
(a) ReLU (b) sReLU (c) ϕ

Figure 1: Activation functions in spatial domain.

3.2 Two MscaleDNN structures

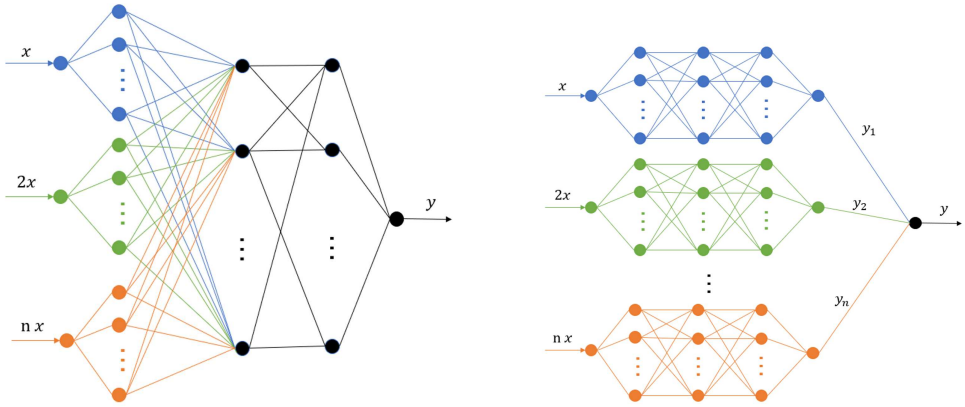
While the procedure leading to (2.15) is not practical for numerical approximation in high dimension, it does suggest a plausible form of function space for finding the solution more quickly with DNN functions. We can use a series of a_i ranging from 1 to a large number to produce a MscaleDNN structure to achieve our goal in speeding up the convergence for solution with a wide range of frequencies with uniform accuracy in frequencies. For this purpose, we propose the following two multi-scale structures.

MscaleDNN-1. For the first kind, we separate the neuron in the first hidden-layer into to N parts. The neuron in the i-th part receives input $a_i\boldsymbol{x}$, that is, its output is $\sigma(a_i\boldsymbol{w}\cdot$



(a) sReLU (b) ϕ

Figure 2: Activation functions in frequency domain, normalized by the maximum of each case.



(a) MscaleDNN-1 (b) MscaleDNN-2

Figure 3: Illustration of two MscaleDNN structures.

$\boldsymbol{x}+b)$, where $\boldsymbol{w}$, $\boldsymbol{x}$, b are weight, input, and bias parameters, respectively. A complete MscaleDNNs takes the following form

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}^{[L-1]}\sigma\circ(\cdots(\boldsymbol{W}^{[1]}\sigma\circ(\boldsymbol{W}^{[0]}(\boldsymbol{K}\odot\boldsymbol{x})+\boldsymbol{b}^{[0]})+\boldsymbol{b}^{[1]})\cdots)+\boldsymbol{b}^{[L-1]}, \tag{3.4}$$

where $\boldsymbol{x}\in\mathbb{R}^d$, $\boldsymbol{W}^{[l]}\in\mathbb{R}^{m_{l+1}\times m_l}$, m_l is the neuron number of l-th hidden layer, $m_0=d$, $\boldsymbol{b}^{[l]}\in\mathbb{R}^{m_{l+1}}$, σ is a scalar function and "$\circ$" means entry-wise operation, $\odot$ is the Hadamard product and

$$\boldsymbol{K} = (\underbrace{a_1,a_1,\cdots,a_1}_{\text{1st part}},\underbrace{a_2,a_2,\cdots,a_2}_{\text{2nd part}},a_3,\cdots,a_{i-1},\underbrace{a_i,a_i,\cdots,a_i}_{i\text{th part}},\cdots,\underbrace{a_N,a_N\cdots,a_N}_{N\text{th part}})^T, \tag{3.5}$$

where $a_i=i$ or $a_i=2^{i-1}$.

We refer to this structure as Multi-scale DNN-1 (MscaleDNN-1) of the form in Eq. (3.4), as depicted in Fig. 3(a).

MscaleDNN-2. A second kind of multi-scale DNN is given in Fig. 3(b), as a sum of N subnetworks, in which each scale input goes through a subnetwork. In MscaleDNN-2, weight matrices from $W^{[1]}$ to $W^{[L-1]}$ are block diagonal. Again, we could select the scale coefficient $a_i=i$ or $a_i=2^{i-1}$.

For comparison studies, we will define a "**normal**" network as an one fully connected DNN with no multi-scale features. We would perform extensive numerical experiments to examine the effectiveness of different settings and use an efficient one to solve complex problems. All models are trained by Adam [16] with learning rate 0.001.

4 MscaleDNN for approximations and elliptic PDE's solutions

In this section, we will address two problems, i.e., fitting functions and solving PDEs such as the PB equations, to show the effectiveness of MscaleDNNs in the following sections.

4.1 Mean squared error training for fitting functions

A DNN, denoted by $f_\theta(\boldsymbol{x})$, will be trained with the mean squared error (MSE) loss to fit a target function $f^*(\boldsymbol{x})$. The loss function is defined as

$$\min L(f_\theta)=\int_\Omega |f^*(\boldsymbol{x})-f_\theta(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x}, \tag{4.1}$$

where $f_\theta(\boldsymbol{x})$ is a neural network with parameter θ.

In our training process, the training data are sampled from $f(\boldsymbol{x})$ at each training epoch, the loss at each epoch is

$$L_S(f_\theta)=\frac{1}{n}\sum_{x\in S}|f^*(\boldsymbol{x})-f_\theta(\boldsymbol{x})|^2, \tag{4.2}$$

where n is the sample size in S.

The above training process requires all information of the target function, which indicates such a training process is not of much practical use. We conduct this study to examine the ability of a DNN in fitting high-frequency functions given sufficient information of the target function.

4.2 A Ritz variational method for Poisson-Boltzmann equations

Let us consider the following elliptic Poisson-Boltzmann equation,

$$-\nabla(\epsilon(\boldsymbol{x})\nabla u(\boldsymbol{x}))+\kappa(\boldsymbol{x})u(\boldsymbol{x})=f(\boldsymbol{x}),\quad \boldsymbol{x}\in\Omega\subset\mathbb{R}^d, \tag{4.3}$$

where $\epsilon(\boldsymbol{x})$ is the dielectric constant and $\kappa(\boldsymbol{x})$ the inverse Debye-Huckel length of an ionic solvent. For a typical solvation problem of a solute such as a bio-molecule in ionic solvent,

the dielectric constant will be a discontinuous function across the solute-solvent interface Γ where the following transmission condition will be imposed,

$$[u](\boldsymbol{x})=0, \quad \boldsymbol{x}\in\Gamma, \tag{4.4}$$

$$\left[\epsilon\frac{\partial u}{\partial n}\right](\boldsymbol{x})=0, \quad \boldsymbol{x}\in\Gamma, \tag{4.5}$$

where $[\,]$ denotes the jump of the quantity inside the square bracket and, for simplicity, an approximate homogeneous boundary condition on $\partial\Omega$ is used for this study, i.e.

$$u|_{\partial\Omega}=0. \tag{4.6}$$

We will apply the deep Ritz method as proposed in [11], which produces a variational solution $u(\boldsymbol{x})$ of Eqs. (4.3), (4.4) and (4.5) through the following minimization problem

$$u=\arg\min_{v\in H_0^1(\Omega)} J(v), \tag{4.7}$$

where the energy functional is defined as

$$J(v)=\int_\Omega \frac{1}{2}\left(\epsilon(\boldsymbol{x})|\nabla v(\boldsymbol{x})|^2+\kappa(\boldsymbol{x})v(\boldsymbol{x})^2\right)d\boldsymbol{x}-\int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})d\boldsymbol{x}\triangleq\int_\Omega E(v(\boldsymbol{x}))\mathrm{d}\boldsymbol{x}. \tag{4.8}$$

We use the MscaleDNN $u_\theta(\boldsymbol{x})$ to represent trial functions $v(\boldsymbol{x})$ in the above variational problem, where θ is the DNN parameter set. Then, the MscaleDNN solution is

$$\boldsymbol{\theta}_*=\arg\min_\theta J(u_\theta(\boldsymbol{x})). \tag{4.9}$$

The minimizer $\boldsymbol{\theta}_*$ can be found by a stochastic gradient decent (SGD) method,

$$\boldsymbol{\theta}^{(n+1)}=\boldsymbol{\theta}^{(n)}+\eta\nabla_{\boldsymbol{\theta}}J(u_\theta(\boldsymbol{x})). \tag{4.10}$$

The integral in Eq. (4.8) will only be sampled at some random points $\boldsymbol{x}_i$, $i=1,\cdots,n$ at each training step (see (2.11) in [11]), namely,

$$\nabla_{\boldsymbol{\theta}}J(u_\theta(\boldsymbol{x}))\sim\nabla_{\boldsymbol{\theta}}\frac{1}{n}\sum_{i=1}^{n}E(u_\theta(\boldsymbol{x}_i)). \tag{4.11}$$

At convergence $\boldsymbol{\theta}^{(n)}\to\boldsymbol{\theta}_*$, we obtain a MscaleDNN solution $u_{\theta_*}(\boldsymbol{x})$.

Variational functional for non-homogeneous Dirichlet boundary conditions

To derive the functional for (4.3) with a non-homogeneous boundary condition

$$u|_{\partial\Omega}=g, \tag{4.12}$$

we will construct a spatial extension function $\widetilde{g}\in C^2(\Omega)$, such that

$$\widetilde{g}|_{\partial\Omega}=g,\quad \frac{\partial\widetilde{g}}{\partial n}\Big|_{\partial\Omega}=0,\quad \sup(\widetilde{g})\cap\Gamma=\emptyset, \tag{4.13}$$

and consider the function

$$w=u-\widetilde{g}, \tag{4.14}$$

which satisfies Eq. (4.3) with a new right hand side

$$\widetilde{f}=f+\epsilon(x)\Delta\widetilde{g}-\kappa(x)\widetilde{g} \tag{4.15}$$

with an homogeneous boundary condition on $\partial\Omega$, and can be found as the minimizer of the following minimization problem

$$w=\arg\min_{v\in H_0^1(\Omega)} J(v), \tag{4.16}$$

where

$$J(v)=J(v;\widetilde{f})=\frac{1}{2}\int_\Omega \left(|\epsilon(x)\nabla v(x)|^2+\kappa(x)v(x)^2\right)dx-\int_\Omega \widetilde{f}v(x)dx. \tag{4.17}$$

Now consider the set

$$V_g=\{\varpi\in H^1(\Omega)\,|\,\varpi=\widetilde{g}+v,\ v\in H_0^1(\Omega)\}=\widetilde{g}\oplus H_0^1(\Omega)\subset H^1(\Omega). \tag{4.18}$$

Using the definition of V_g in (4.18) and $\widetilde{f}$ in (4.15), we can show that for piecewise constant $\epsilon(x)$,

$$\begin{aligned}
J(v)&=\frac{1}{2}\int_\Omega\left(|\epsilon(x)(\nabla\varpi-\nabla\widetilde{g})|^2+\kappa(x)(\varpi-\widetilde{g})^2\right)dx-\int_\Omega(f+\epsilon(x)\Delta\widetilde{g}-\kappa(x)\widetilde{g})(\varpi-\widetilde{g})dx\\
&=\left(\frac{1}{2}\int_\Omega\left(|\epsilon(x)\nabla\varpi|^2+\kappa(x)\varpi^2\right)dx-\int_\Omega f\varpi dx\right)-\int_\Omega(\epsilon(x)\nabla\varpi\nabla\widetilde{g}+\kappa(x)\varpi\widetilde{g})dx\\
&\quad+\frac{1}{2}\int_\Omega\left(|\epsilon(x)\nabla\widetilde{g}|^2+\kappa(x)\widetilde{g}^2\right)dx-\int_\Omega(\epsilon(x)\Delta\widetilde{g}-\kappa(x)\widetilde{g})\varpi dx\\
&\quad+\int_\Omega f\widetilde{g}dx+\int_\Omega(\epsilon(x)\Delta\widetilde{g}-\kappa(x)\widetilde{g})\widetilde{g}dx\\
&=J(\varpi;f)-\int_\Omega\epsilon\nabla\varpi\nabla\widetilde{g}dx-\int_\Omega\epsilon\Delta\widetilde{g}\varpi dx+C(f,\widetilde{g})\\
&=J(\varpi;f)+\int_{\partial\Omega}\epsilon\frac{\partial\widetilde{g}}{\partial n}\varpi dx+C(f,\widetilde{g})\\
&=J(\varpi;f)+C(f,\widetilde{g}),
\end{aligned}$$

where the term $C(f,\widetilde{g})$ is considered as a constant during the minimization process. Therefore, we have

$$\arg\min_{v\in H_0^1(\Omega)} J(v,\widetilde{f}) = \arg\min_{\varpi\in V_g} J(\varpi,f), \tag{4.19}$$

where

$$J(\varpi) = J(\varpi,f) = \frac{1}{2}\int_\Omega \left(|\epsilon(x)\nabla\varpi|^2+\kappa(x)\varpi^2\right)dx - \int_\Omega f\varpi dx. \tag{4.20}$$

In practice, a penalty term can be added in the functional to enforce the boundary condition, namely

$$w = \arg\min_{\varpi\in H^1(\Omega)} J(\varpi)+\beta||\varpi-g||^2. \tag{4.21}$$

In our numerical tests, the Ritz loss function is taken as

$$\begin{aligned} L_{\text{ritz}}(u_\theta) = &\frac{1}{n}\sum_{\boldsymbol{x}\in S}(\epsilon(\boldsymbol{x})|\nabla u_\theta(\boldsymbol{x})|^2/2+\kappa(\boldsymbol{x})u_\theta(\boldsymbol{x})^2/2-f(\boldsymbol{x})u_\theta(\boldsymbol{x})) \\ &+\beta\frac{1}{\tilde{n}}\sum_{\boldsymbol{x}\in\tilde{S}}(u_\theta(\boldsymbol{x})-g(\boldsymbol{x}))^2, \end{aligned} \tag{4.22}$$

where $u_\theta(\boldsymbol{x})$ is the DNN output, S is the sample set from Ω and n is the sample size, $\tilde{n}$ indicates the number of sample set $\tilde{S}$ from $\partial\Omega$. We choose $\beta=1000$ for all experiments.

To see the learning accuracy, we also compute the L^2 error between $u_\theta(\boldsymbol{x})$ and $u_{\text{true}}(\boldsymbol{x})$ on test data points $S_t=\{\boldsymbol{x}_i\}_{i=1}^{n_t}$ inside the domain,

$$\text{error}(u_\theta(\boldsymbol{x}),u_{\text{true}}(\boldsymbol{x})) = \left(\frac{1}{n_t}\sum_{\boldsymbol{x}\in S_t}(u_\theta(\boldsymbol{x})-u_{\text{true}}(\boldsymbol{x}))^2\right)^{1/2}. \tag{4.23}$$

5 Effectiveness of various MscaleDNN settings

In this section, we will show that MscaleDNNs outperform normal fully-connected DNNs (indicated by "normal" in the numerical results) in various settings, namely, the loss function of MscaleDNN decays faster to smaller values than that of normal fully-connected DNNs.It will also reflect smaller errors for the solutions for the MscaleDNN. First, we will carry out two test problems. Second, we will demonstrate that compact supported activation functions of sReLU(x) and $\phi(x)$ are much better than the commonly used ReLU(x). Third, we use activation functions $\phi(x)$ to show MscaleDNN structures are better than normal fully connected one. Finally, we examine the effects of various scale selections.

5.1 Two test problems

To understand the performance of different MscaleDNNs and their parameters, here we consider one- and two- dimensional problems in fitting functions and solving PDEs, and problems in 3-D in complex domains will be considered in the next section.

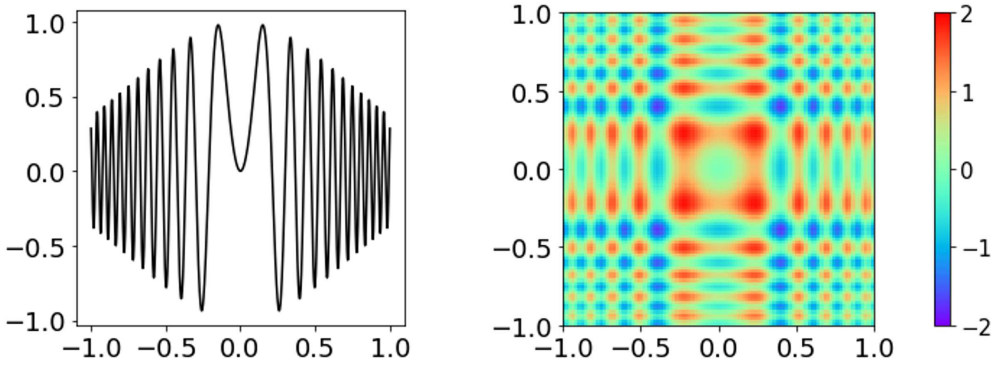
Test problem 1: Fitting problem. The target function for the fitting problem is $F:[-1,1]^d\to\mathbb{R}$

$$F(\boldsymbol{x})=\sum_{j=1}^{d}g(x_j)\quad x_j\in[-1,1], \tag{5.1}$$

where $\boldsymbol{x}=(x_1,\cdots,x_d)$,

$$g(x)=e^{-x^2}\sin(\mu x^2).$$

In the case of $d=1$, we choose $\mu=70$ while for the case of $d=2$, $\mu=30$. The functions of $d=1$ and $d=2$ are shown in Fig. 4. 5000 training data at each epoch and 500 test data are randomly sampled from $[-1,1]^d$. All DNNs are trained by the Adam optimizer with learning rate 0.001.



(a) $d=1$ (b) $d=2$

Figure 4: Test Problem 1: target functions for fitting problems.

Test problem 2: Solving PB equations. We will solve the elliptic equation (4.3) with $\epsilon=1$ and a constant $\kappa(\boldsymbol{x})=\lambda^2$ in a domain $\Omega=[-1,1]^d$ and the right hand side

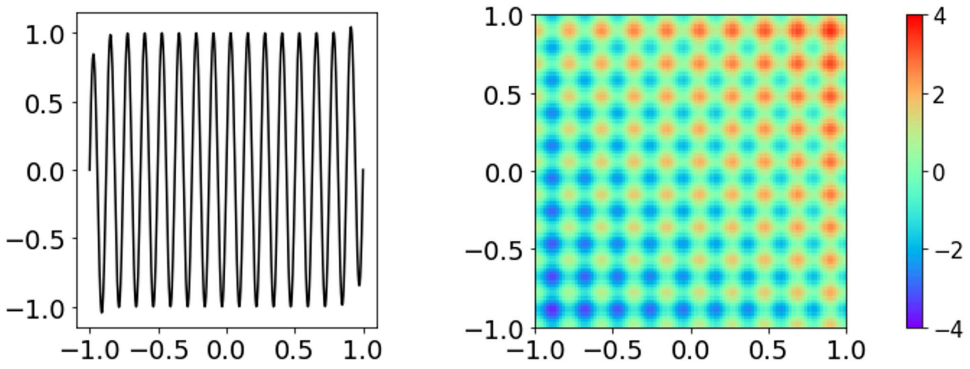
$$f(\boldsymbol{x})=\sum_{i=1}^{d}(\lambda^2+\mu^2)\sin(\mu x_i),$$

which gives a PB equation with the following exact solution,

$$u(\boldsymbol{x})=\sum_{i=1}^{d}-\frac{\sin\mu}{\sinh\lambda}\sinh(\lambda x_i)+\sin(\mu x_i)$$

with corresponding boundary condition given by the exact solution.

For $d=1$, we choose $\lambda=20$, $\mu=50$. For $d=2$, we choose $\lambda=2$, $\mu=30$. The exact solutions for $d=1$ and $d=2$ are shown in Fig. 5. DNNs are trained by Adam optimizer



(a) $d=1$ (b) $d=2$

Figure 5: Test problem 2: exact solutions of 1-D and 2-D PB equation.

with learning rate 0.001. 5000 training data at each epoch are randomly sampled from Ω. We choose the penalty coefficient for boundary as $\beta=1000$. The number of boundary data randomly sampled from $\partial\Omega$ is 400 for $d=1$ and 4000 for $d=2$.

5.2 Different activation functions

We use the following three network structures to examine the effectiveness of different activation functions by solving one-dimensional fitting and PDE problems described above:

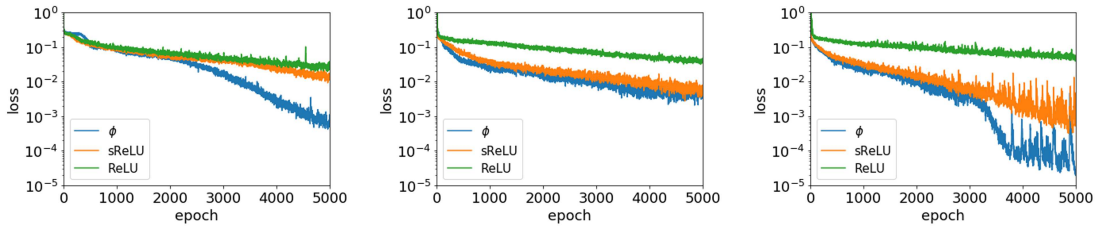
1. fully-connected DNN with size **1-900-900-900-1** (**normal**);

2. MscaleDNN-1 with size **1-900-900-900-1** and scale coefficients of $\{1,2,4,8,16,32\}$ (**MscaleDNN-1(32)**);

3. MscaleDNN-2 with six subnetworks with size **1-150-150-150-1** and scale coefficients of $\{1,2,4,8,16,32\}$ (**MscaleDNN-2(32)**).

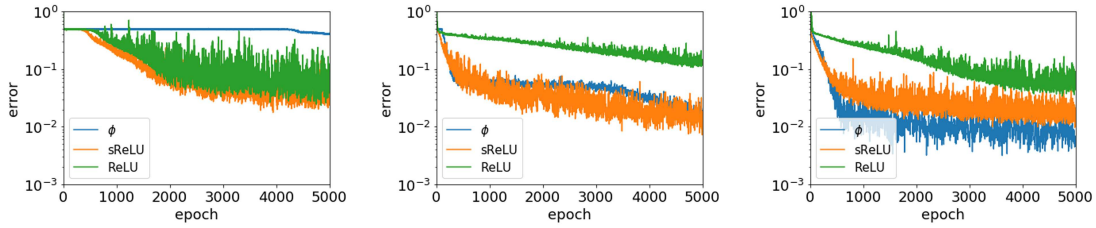
In Fig. 8, we increase the number of total epoch to 50000. The results are similar. Therefore, several thousand epochs are enough to compare the performance of networks.

We use three different activation functions, i.e., ReLU, sReLU, ϕ for the above structures. For normal network structure in the fitting problem, as shown in Fig. 6(a), ϕ (blue) performs much better than other two activation functions. However, with a normal network structure to solve the PDE, as shown in Fig. 7(a), ϕ (blue) performs much worse than other two activation functions. The results indicate all three activation function are not stable in a normal fully connected structure. On the other hand, as shown in Figs. 6(b,c), and 7(b,c), for both MscaleDNN structures, the performance of compact supported activation functions, sReLU (orange) and ϕ (blue), are much better than that of ReLU (green) for both test problems.



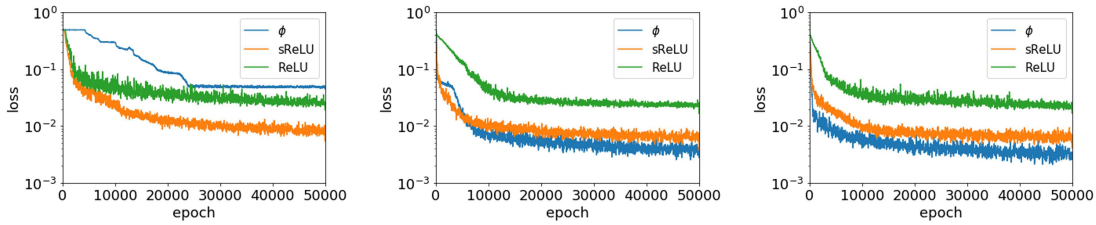
(a) normal (b) MscaleDNN-1 (c) MscaleDNN-2

Figure 6: Different activation functions in 1-D fitting problems.



(a) normal (b) MscaleDNN-1 (c) MscaleDNN-2

Figure 7: Different activation functions in a 1-D PB equation.



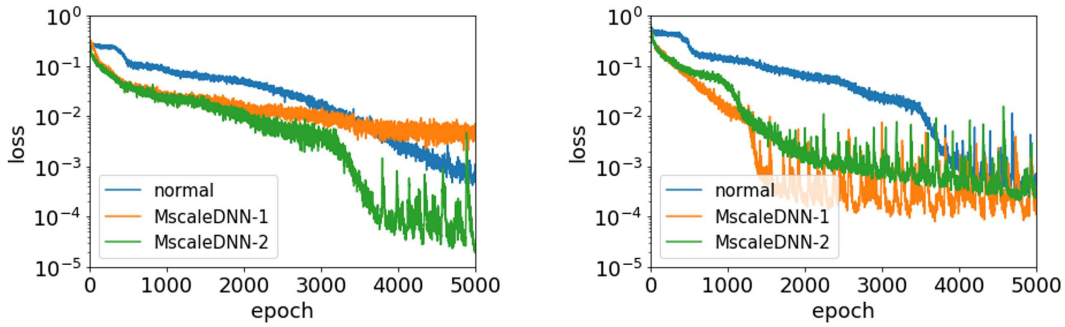
(a) normal (b) MscaleDNN-1 (c) MscaleDNN-2

Figure 8: Different activation functions in a 1-D PB equation.

5.3 Different network structures

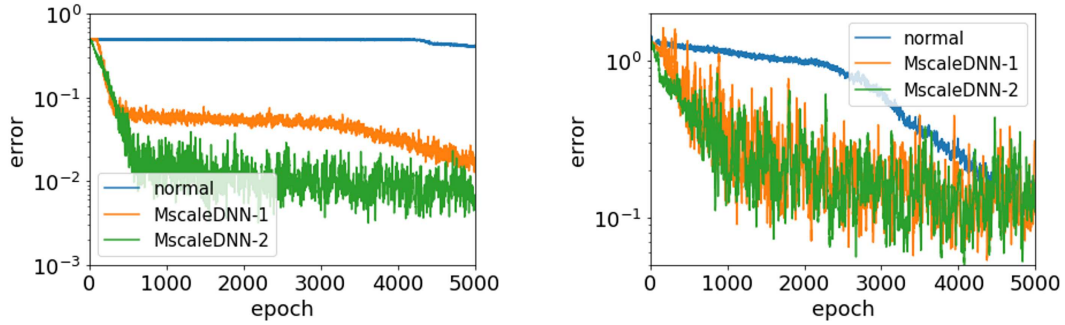
In this subsection, we examine the effectiveness of the following different network structures with the activation function of $\phi(x)$:

1. fully-connected DNN with size **1-900-900-900-1** (**normal**);
2. MscaleDNN-1 with size **1-900-900-900-1** and scale coefficients of $\{1,2,4,8,16,32\}$ (**MscaleDNN-1(32)**);
3. MscaleDNN-2 with six subnetworks with size **1-150-150-150-1** and scale coefficients of $\{1,2,4,8,16,32\}$ (**MscaleDNN-2(32)**).



(a) 1d function (b) 2d function

Figure 9: Different network structures in fitting problems.



(a) 1d equation (b) 2d equation

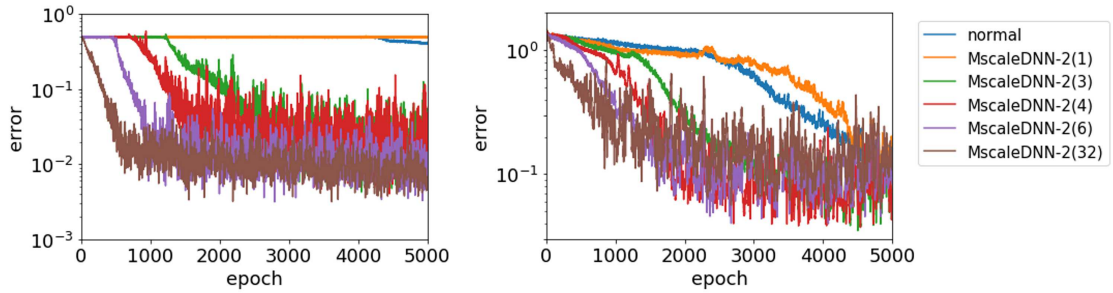
Figure 10: Different network structures in PDE problems.

As shown in Figs. 9 and 10, both MscaleDNN structures are better than normal structures in both problems. Two different MscaleDNN structures have similar performance in both test problems. As MscaleDNN-2 performs better than MscaleDNN-1 and also has much less connections compared with MscaleDNN-1 and a dynamic adaptive strategy of adding and removing scales can also be implemented, in the following we will use MscaleDNN-2 for further numerical experiments.

5.4 Different scale selections in MscaleDNNs

In this subsection, we will test different scales for the activation function in MscaleDNNs:

1. fully-connected DNN with size **1-900-900-900-1** (**normal**);
2. MscaleDNN-2 with six subnetworks with size **1-150-150-150-1** and scale coefficients of $\{1,1,1,1,1,1\}$ (**MscaleDNN-2(1)**);
3. MscaleDNN-2 with three subnetworks with size **1-300-300-300-1** and scale coefficients of $\{1,2,3\}$ (**MscaleDNN-2(3)**);



(a) 1d equation (b) 2d equation

Figure 11: Different scale options in 1-D and 2-D PB equations.

4. MscaleDNN-2 with three subnetworks with size **1-300-300-300-1** and scale coefficients of $\{1,2,4\}$ (**MscaleDNN-2(4)**);

5. MscaleDNN-2 with six subnetworks with size **1-150-150-150-1** and scale coefficients of $\{1,2,3,4,5,6\}$ (**MscaleDNN-2(6)**);

6. MscaleDNN-2 with six subnetworks with size **1-150-150-150-1** and scale coefficients of $\{1,2,4,8,16,32\}$ (**MscaleDNN-2(32)**).

As shown in Fig. 11, MscaleDNNs almost perform consistently better than normal DNNs. Note that with larger-range scale, MscaleDNN solves the problem faster. With all scales as 1, the performance of DNN structure (MscaleDNN-2(1)) is much worse than those with multiscales in solving elliptic PDEs. Therefore, with the subnetwork structures with different scales, the MscaleDNN is able to achieve a faster convergence. These experiments show that MscaleDNNs with proper scales are more efficient in solving PDE problems and the selection of the scales are not too sensitive.

With these numerical experiments, we have demonstrated that MscaleDNN is much more efficient to solve elliptic PDEs and the preferred network is MscaleDNN-2 with the compact support function $\phi(x)$, which will be used for the rest of the paper for solving Poisson and PB equations in complex and/or singular domains.

6 MscaleDNNs for Poisson and Poisson-Boltzmann equations in complex and singular domains

In this section, we apply MscaleDNNs with activation function $\phi(x)$ to solve complex elliptic equations, including cases with a broad range of frequencies, variable coefficients, a ring-shaped domain, and a cubic domain with multiple holes. Finally, we apply the MscaleDNN to solve PB equations with geometric singularities, such as cusps and self-intersecting surfaces, which comes from a typical bead-model of bio-molecule. Through

these experiments, we convincingly demonstrate that MscaleDNNs are an efficient and easy-implemented mesh-less method to solve complex elliptic PDEs.

6.1 Poisson equation in complex domains

6.1.1 Broad range of frequencies

Consider the Poisson equation in $\Omega=[-1,1]^d$,

$$-\Delta u(\boldsymbol{x})=f(\boldsymbol{x}), \tag{6.1}$$

where

$$f(\boldsymbol{x})=\sum_{i=1}^{d}4\mu^2x_i^2\sin(\mu x_i^2)-2\mu\cos(\mu x_i^2). \tag{6.2}$$

The equation has an exact solution as

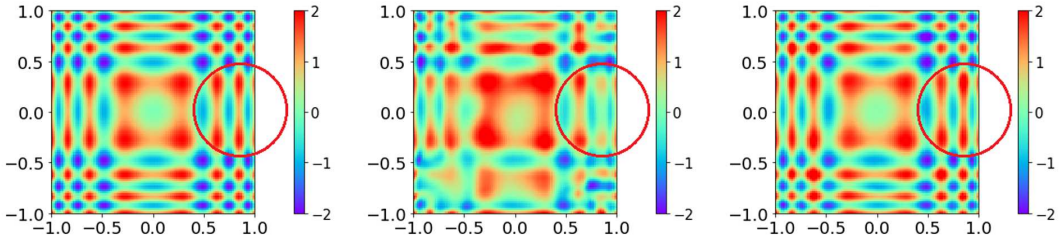
$$u(\boldsymbol{x})=\sum_{i=1}^{d}\sin(\mu x_i^2), \tag{6.3}$$

which will also provide the boundary condition in problem (6.1).

In each training epoch, we sample 5000 points inside the domain and 4000 points from the boundary. We examine the following two structures:

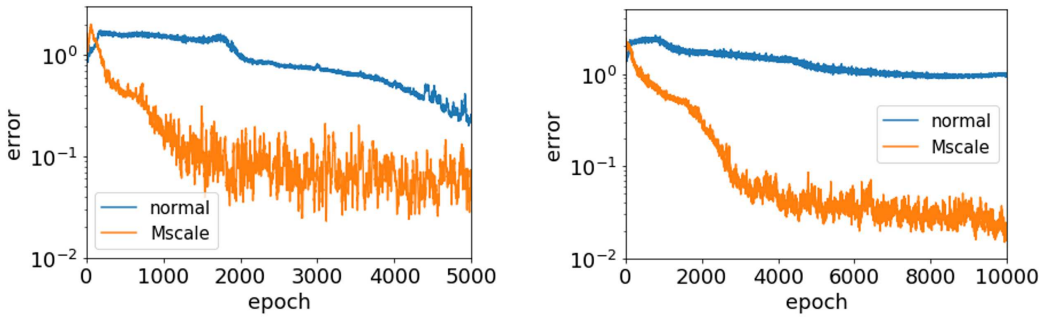
1. a fully-connected DNN with size **1-1000-1000-1000-1** (**normal**);
2. a MscaleDNN-2 with five subnetworks with size **1-200-200-200-1**, and scale coefficients $\{1,2,4,8,16\}$ (**Mscale**).

This problem does not have a fixed frequency but a broad range of frequencies. A commonly-used fully connected DNN will not be able to solve this problem. For $\mu=15$, the exact solution for the two-dimensional case of problem (6.1) is shown in Fig. 12(a) as a highly oscillated function. The solution, obtained by the normal DNN in Fig. 12(b),



(a) exact (b) normal (c) Mscale

Figure 12: Two-dimensional case for problem (6.1). As an example, the MscaleDNN well captures the oscillation in the red marked circle while the normal fully connected network fails.



(a) 2d Poisson equation (b) 3d Poisson equation

Figure 13: Error vs. epoch for problems with broad range of frequencies.

fails to capture the oscillate structure, while the solution obtained by the MscaleDNN in Fig. 12(c) captures well the different-scale oscillations. For example of the area marked by the red circle, the expected oscillation almost disappears in the solution of the normal networks while MscaleDNN solutions resolve the oscillations well. Similar behavior differences occur for the oscillations at four corners.

The errors of the two-dimensional and the three-dimensional problems are shown in Fig. 13(a) and (b), respectively. In both cases, MscaleDNNs solve problems much faster to lower errors.

6.1.2 A ring-shaped domain

Consider the Poisson equation (6.1) in a ring-shaped domain Ω with its center at $(0,0)$ and inner radius 1 and outer radius 3 with a source term

$$f(\boldsymbol{x})=\mu^2 J_0(\mu|\boldsymbol{x}-\boldsymbol{x}_0|), \tag{6.4}$$

where J_0 is the Bessel function. The exact solution is given by

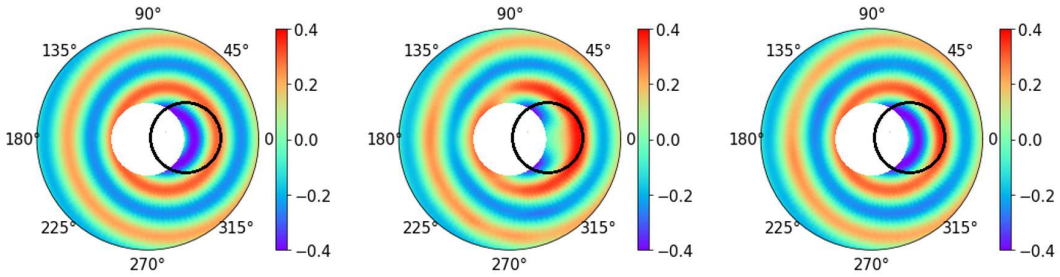
$$u(\boldsymbol{x})=J_0(\mu|\boldsymbol{x}-\boldsymbol{x}_0|). \tag{6.5}$$

Again, the boundary condition is given by the exact solution $u(\boldsymbol{x})$. We choose $\boldsymbol{x}_0=(0.5,0)$ and solve the equation with $\mu=5$ and $\mu=10$.

In each training epoch, we sample 5000 points inside the domain and 4000 points from the boundary. We examine the following two structures:

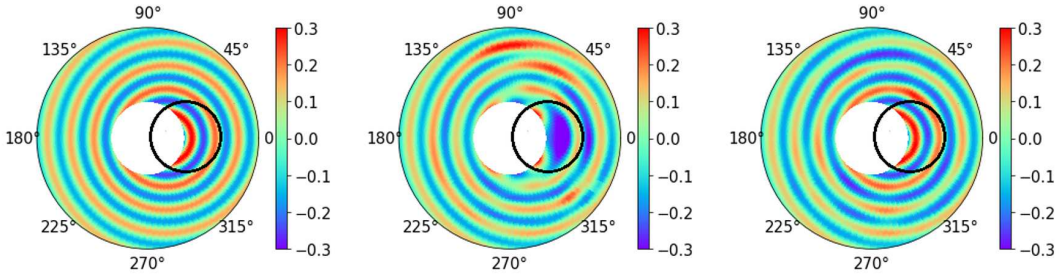
1. a fully-connected DNN with size **1-500-500-500-1** (**normal**);
2. a MscaleDNN-2 with five subnetworks with size **1-100-100-100-1** and scale coefficients $\{1,2,4,8,16\}$ (**Mscale**).

The exact solutions and numerical solutions obtained by normal and MscaleDNNs are shown in Fig. 14 ($\mu=5$) and Fig. 15 ($\mu=10$). To highlight the superior performance



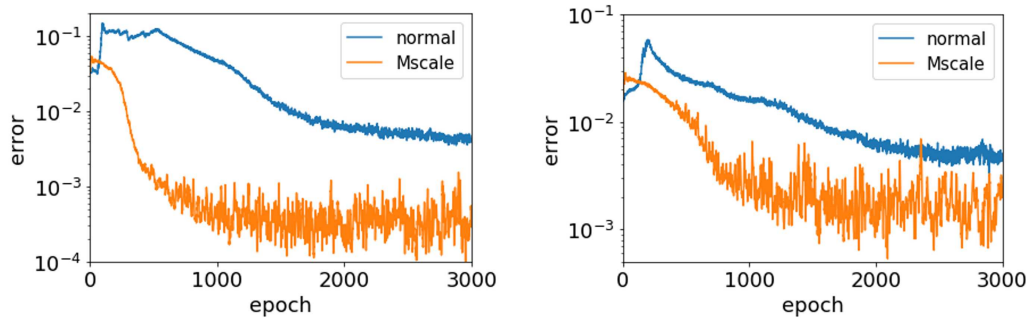
(a) exact (b) normal (c) Mscale

Figure 14: Exact and numerical solutions for the equation in a ring-shaped domain with $\mu=5$. The black circle is for illustration purpose only.



(a) exact (b) normal (c) Mscale

Figure 15: Exact and numerical solution for the equation in ring-shaped domain with $\mu=10$. The black circle is for illustration purpose only.



(a) $\mu=5$ (b) $\mu=10$

Figure 16: Error vs. epoch for the Poisson equation in ring-shaped domain.

of the MscaleDNNs, areas in the figures marked by the black circle show the region of the solution with the largest amplitude, the normal networks completely fail to capture the oscillations while the MscaleDNNs faithfully captures them in both cases. Again, as shown in Fig. 16, MscaleDNN solves both problems with a much better accuracy.

6.1.3 A square domain with a few holes

Domain one. The centers for three circle holes are $(-0.5,-0.5)$, $(0.5,0.5)$, and $(0.5,-0.5)$, with radii of 0.1, 0.2, and 0.2, respectively. In each epoch, we randomly sample 3000 on outer boundary, 800 points on the boundary of each big hole and 400 points on the boundary of the small hole.

Domain two. The centers for three circle holes are $(-0.6,-0.6)$, $(0.3,-0.3)$ and $(0.6,0.6)$, with radii of 0.3, 0.6, 0.3, respectively. The boundary of the elliptic hole is described by $16(x+0.5)^2+64(y-0.5)^2=1$. The sample sizes at each epoch are 2400, 1100, 550, and 400 for the outer boundary, the boundary of the big circle hole, the boundary of each small circle hole, and the boundary of the elliptic hole, respectively.

We solve the Poisson equation (6.1) with the source term as

$$f(\boldsymbol{x})=2\mu^2\sin\mu x_1\sin\mu x_2,\quad \mu=7\pi. \tag{6.6}$$

The exact solution is

$$u(\boldsymbol{x})=\sin\mu x_1\sin\mu x_2, \tag{6.7}$$

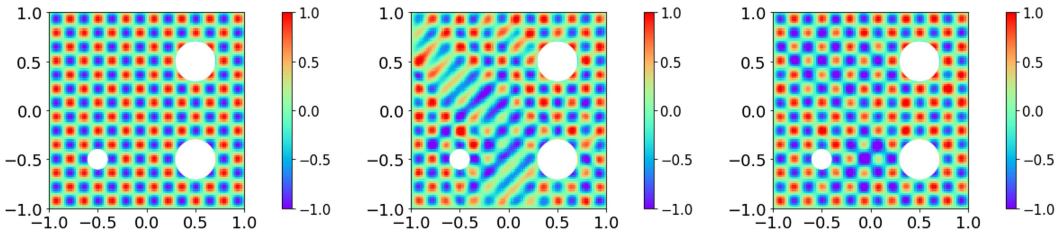
which also provides the boundary condition. In each training epoch, we sample 5000 points inside the domain with the following two DNN structures:

1. a fully-connected DNN with size **1-1000-1000-1000-1** (**normal**);

2. a MscaleDNN-2 with five subnetworks with size **1-200-200-200-1**, and scale coefficients of $\{1,2,4,8,16\}$ (**Mscale**).

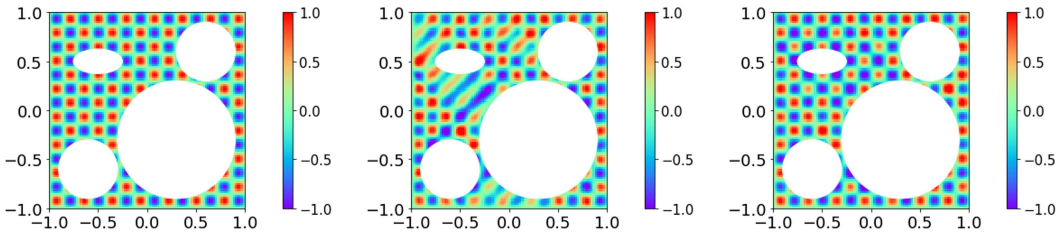
As shown in Fig. 19. MscaleDNNs solve both problems much faster to lower errors.

Compared with the exact solutions in Fig. 17(a) and Fig. 18(a), normal DNN fails to resolve the magnitudes of many oscillations as shown in Fig. 17(b) and Fig. 18(b) while MscaleDNNs capture each oscillation of the true solutions accurately as shown in Fig. 17(c) and Fig. 18(c).



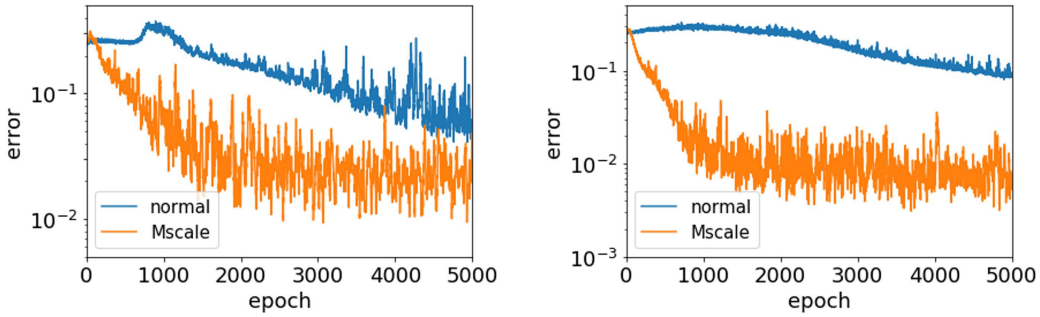
(a) exact (b) normal (c) Mscale

Figure 17: Exact and numerical solution for the Poisson equation in domain 1.



(a) exact (b) normal (c) Mscale

Figure 18: Exact and numerical solution for the Poisson equation in domain 2.



(a) domain 1 (b) domain 2

Figure 19: Error vs. epoch for the Poisson equation in square domains with few holes.

6.1.4 A square domain with many holes

To verify the capability of the MscaleDNN for complex domains, we consider a three dimensional cube $[-1,1]^3$ with 125 holes inside removed as shown in Fig. 20, and the holes are centered at a uniform mesh, i.e., $\{-0.8,-0.4,0,0.4,0.8\}^3$, with radii randomly sampled from a uniform distribution in $[0,0.15]$. The sample sizes for training DNNs at each training epoch are 2500 for the outer boundary and 1500 for the inner holes (12 points for each hole).

Again, consider the Poisson equation with $f(x)$ and the Dirichlet boundary condition given by the exact solution $u(\boldsymbol{x})$ for the following three cases:

1. Example 1: $u(\boldsymbol{x}) = \sin\mu x_1 \sin\mu x_2 \sin\mu x_3$.
2. Example 2: $u(\boldsymbol{x}) = e^{\sin\mu x_1+\sin\mu x_2+\sin\mu x_3}$.
3. Example 3: $u(\boldsymbol{x}) = e^{\sin\mu x_1 \sin\mu x_2 \sin\mu x_3}$.

The difficulty of this problem consists of the complex holes and oscillatory exact solutions with $\mu = 7\pi$. In each training epoch, we sample 5000 points inside the domain, and compare the following two structures:

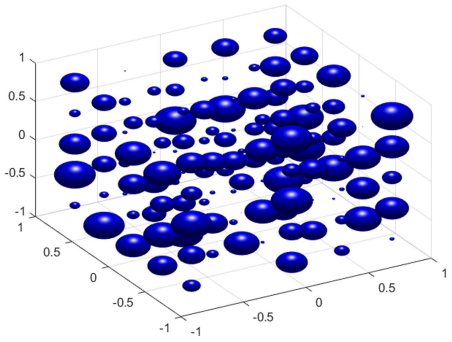
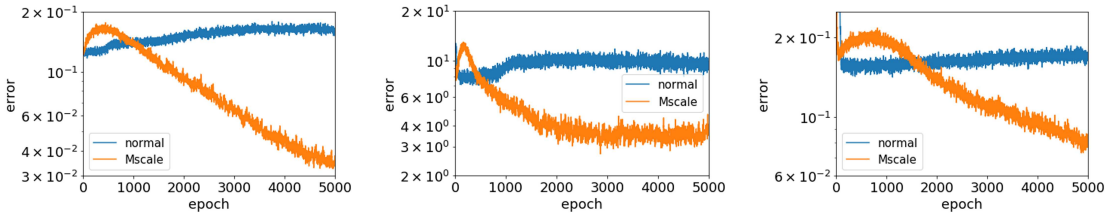


Figure 20: Holes of domain for the problem.



(a) Example 1 (b) Example 2 (c) Example 3

Figure 21: Error vs. epoch for the PDEs in square domain with many holes.

1. a fully-connected DNN with size **1-1000-1000-1000-1** (**normal**);
2. a MscaleDNN-2 with five subnetworks with size **1-200-200-200-1**, and scale coefficients of $\{1,2,4,8,16\}$ (**Mscale**).

As shown in Fig. 21 for all three cases, the normal fully-connected structures do not converge for such complex problems at all while MscaleDNNs can solve the problem with much smaller errors.

6.2 Poisson-Boltzmann equations with domain and source singularities

6.2.1 Variable coefficients

Consider the PB equation (4.3) in $\Omega=[-1,1]^3$ with

$$f(\boldsymbol{x})=(\mu_1^2+\mu_2^2+\mu_3^2+x_1^2+2x_2^2+3x_3^2)\sin(\mu_1 x_1)\sin(\mu_2 x_2)\sin(\mu_3 x_3), \tag{6.8}$$

and

$$\kappa(\boldsymbol{x})=(x_1^2+2x_2^2+3x_3^2), \tag{6.9}$$

which has an exact solution as

$$u(\boldsymbol{x})=\sin(\mu_1 x_1)\sin(\mu_2 x_2)\sin(\mu_3 x_3). \tag{6.10}$$

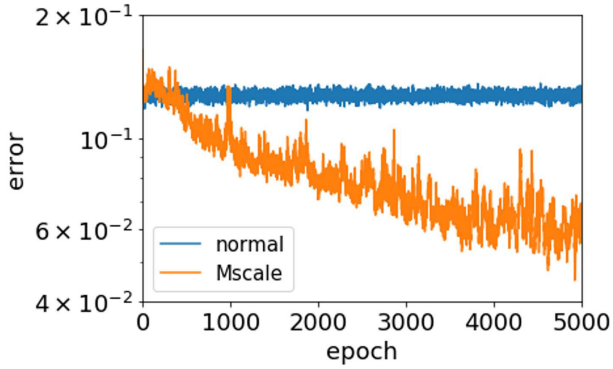


Figure 22: Error vs. epoch for variable coefficient PB equation.

The boundary condition is given by the exact solution $u(\boldsymbol{x})$. We choose $\mu_1=15$, $\mu_2=20$, $\mu_3=25$. In each training epoch, we sample 5000 points inside the domain and 4000 points from the boundary. We compare the following two DNN structures:

1. a fully-connected DNN with size **1-900-900-900-1** (**normal**);
2. a MscaleDNN-2 with six subnetworks with size **1-150-150-150-1** and scale coefficients $\{1,2,4,8,16,32\}$ (**Mscale**).

As shown in Fig. 22, during the training process, the error of the MscaleDNN decays significantly, while the error of the normal DNN almost keeps unchanged. Therefore, MscaleDNN solves the problem much faster with a much better accuracy.

6.2.2 Geometric singularities

In this subsection, we consider the PB equation (4.3) in a domain with geometric singularities and jump condition on interior interfaces, which arises from the simulation of solvation of bio-molecules. Consider an open bounded domain $\Omega_1\subset\mathbb{R}^3$, which divides $\mathbb{R}^3$ into two disjoint open subdomains by the surface $\Gamma=\partial\Omega_1$. Ω_1 is identified as the bio-molecule, and $\Omega_2=\mathbb{R}^3\setminus\Omega_1$ is the solvent region. The exact solution $u(x)$ is also divided into two parts, $u_1(x)$ is defined in Ω_1 and $u_2(x)$ in Ω_2. The solution will also satisfy the transmission condition (4.4), (4.5) along the interface Γ and a decaying condition at the ∞, i.e.

$$\lim_{|\boldsymbol{x}|\to\infty} u_2(\boldsymbol{x})=0. \tag{6.11}$$

To deal with the unbounded domain, we truncate the solution domain to a large ball or cube, denoted by Ω satisfying $\Omega_1\subset\Omega$ and we re-define $\Omega_2=\Omega\setminus\Omega_1$ and set an approximate condition $u_2=0$ on the boundary of the ball (Fig. 23 (left)) and such a crude boundary condition will surely introduce error to the PDEs solution. Higher order boundary conditions have been studied extensively, and as we are more interested in the performance of the DNNs near the interior interface, we will not ponder over this issue here.

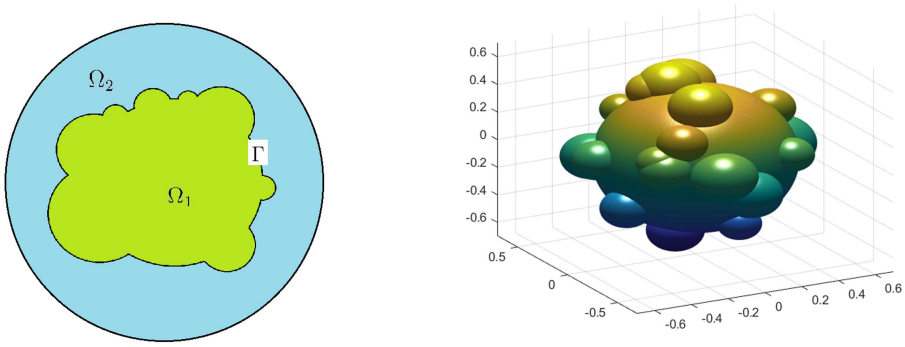


Figure 23: Solution domain: (left) truncation of computation domain, (right) geometric singularity.

The domain with geometric singularities is constructed as follows. We choose a big ball with a center at $(0,0,0)$ and a radius of 0.5. 20 points are randomly selected on the surface of the big ball as the centers of small balls. Radiuses of the small balls are randomly sampled from $[0.1,0.2]$. Ω_1 is the union of these balls and the big ball. The shape of Ω_1 is illustrated in Fig. 23 (right). The intersections among balls cause geometric singularities, such as kinks, which poses major challenges for obtaining mesh generation for traditional finite element and boundary element methods and accurate solution procedures.

Following two examples are considered. In both examples, coefficients $\epsilon(\boldsymbol{x})$ and $\kappa(\boldsymbol{x})$ are chosen as piece-wise constant. Singular sources for the PB equations, which can occur from the point charges inside bio-molecules or ions in the solvents, will be considered later. These point charge sources, modeled by Dirac delta function, will create point singularity in the solution, which can be removed by subtracting a singular solution [7].

Example 1. The exact solution is

$$u(\boldsymbol{x})=\frac{e^{\sin\mu x_1+\sin\mu x_2+\sin\mu x_3}}{|\boldsymbol{x}|^2+1}(|\boldsymbol{x}|^2-1) \tag{6.12}$$

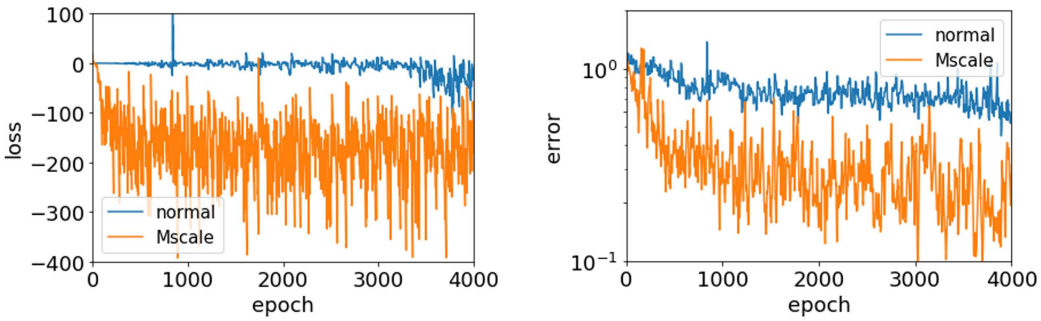
with coefficients for the PB equation as

$$\mu=15,\ \epsilon(\boldsymbol{x})=1,\ \kappa(\boldsymbol{x})=1\ \text{ for }\ \boldsymbol{x}\in\Omega_1,\ \ \epsilon(\boldsymbol{x})=1,\ \kappa(\boldsymbol{x})=5\ \text{ for }\ \boldsymbol{x}\in\Omega_2. \tag{6.13}$$

The whole domain is truncated by a ball with center at $(0,0,0)$ and a radius 1 with zero boundary condition on the sphere.

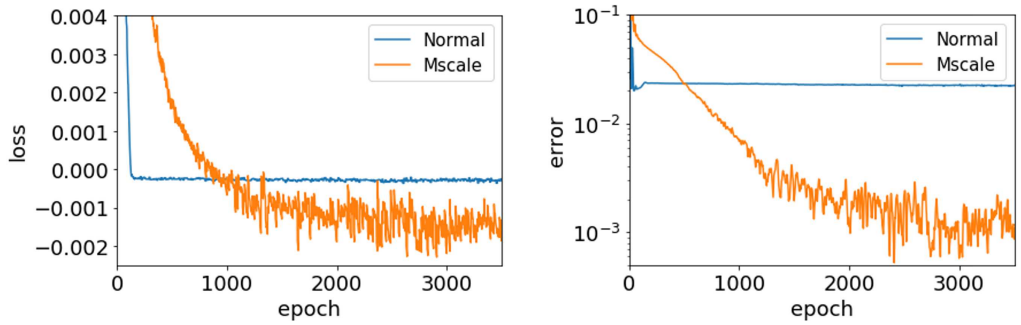
Example 2. We choose

$$f(\boldsymbol{x})=\frac{e^{\sin\mu x_1+\sin\mu x_2+\sin\mu x_3}}{|\boldsymbol{x}|^2+1}(|\boldsymbol{x}|^2-1) \tag{6.14}$$



(a) loss (b) relative error

Figure 24: Loss and relative error vs. epoch for the PB equation in a domain with geometric singularities (Example 1).



(a) loss (b) relative error

Figure 25: Loss and relative error vs. epoch for the PB equations in a domain with geometric singularities (Example 2).

with coefficients

$$\mu = 20,\ \epsilon(\boldsymbol{x}) = 1 \ \text{ for } \ \boldsymbol{x} \in \Omega_1,\ \ \epsilon(\boldsymbol{x}) = 80 \ \text{ for } \ \boldsymbol{x} \in \Omega_2,\ \ \kappa(\boldsymbol{x}) = 1. \tag{6.15}$$

In this case, the computational domain is obtained with a truncation by a cube $[-1,1]^3$ and the reference solution is calculated by finite difference method (FDM) with a sufficient fine mesh ensuring enough accuracy.

In Example 1, in each training epoch, we sample 5000 points inside the domain Ω and 4000 points on boundary $\partial\Omega$. In Example 2, we sample 6000 points inside the domain Ω, 3000 points on boundary $\partial\Omega$. We train MscaleDNNs with the Ritz loss function in (4.22). Note that the continuity condition in (4.4) is satisfied since we use a single network to fit the whole domain Ω; The natural condition in (4.5) is also automatically satisfied due to the use of the Ritz loss.

We examine the following two structures:

1. a fully-connected DNN with size **1-1000-1000-1000-1** (**normal**);

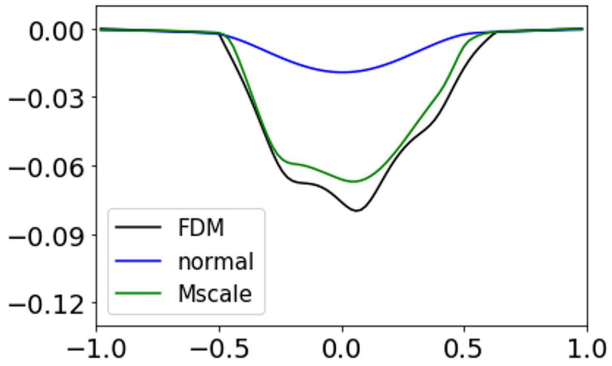


Figure 26: Numerical solutions of Example 2 on line $x_1 = x_3 = 0$.

2. a MscaleDNN-2 with five subnetworks with size **1-200-200-200-1**, and scale coefficients of $\{1,2,4,8,16\}$ (**Mscale**).

Since the value of the exact solution is small, we show the relative L^2 error for both cases. As in practice, the exact solution is unknown, therefore, we also show the training loss for both examples, which could be used as a possible criteria to terminate the training. For Example 1 as shown in Fig. 24, the training loss in Fig. 24(a) and the error in Fig. 24(b) have similar trends, that is, the MscaleDNN converge faster to smaller values, compared with the normal DNN. For Example 2 shown in Fig. 25, the MscaleDNN shows a similar advantage over the normal DNN. These examples indicate that with by just monitoring the training loss, MscaleDNN solves the PB equations with non-smooth solution over singular domains much faster and with better accuracy.

For illustration, we show a cross section of the solution in the second example. The reference solution is obtained by the FDM. Numerical solutions on the line $x_1 = x_3 = 0$ obtained by FDM ($h = 0.02$), normal DNN (5000 epochs) and MscaleDNN (5000 epochs) are shown in Fig. 26. The output of the normal fully connected network gives a wrong solution in the interior of the singular domain while the MscaleDNN gives a satisfactory approximation to the reference solution.

6.2.3 Source and geometric singularities

In this subsection, we consider the PB equation (4.3) with singular sources, that is,

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} q_k \delta(\boldsymbol{x} - \mathbf{s}_k), \tag{6.16}$$

where $\delta(x)$ is Dirac delta function, q_k and $\mathbf{s}_k$ represent the charge and position of one nuclei in the bio-molecule, respectively. We assume that the distance between nucleus and the molecule interface is bigger than a constant R_0, that is, $\Omega_0 = \{\boldsymbol{x} : \exists k, |\boldsymbol{x} - \mathbf{s}_k| < R_0\} \subset \Omega_1$. In Fig. 27, the blue part represents the solvent domain Ω_2, the green part represents

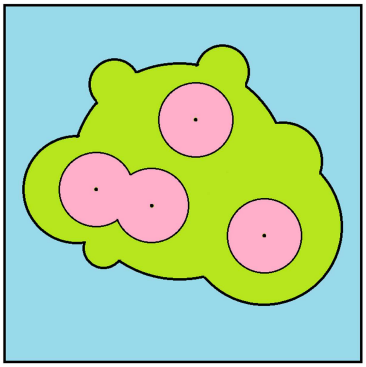


Figure 27: Spherical truncation of the physical domain.

the biomolecular domain $\Omega_1\backslash\Omega_0$, and the pink part represents Ω_0, which contains all charges.

To deal with singularities, we define

$$\bar{u}(\boldsymbol{x})=\sum_{k=1}^{K}q_kG(\boldsymbol{x}-\mathbf{s}_k)m(\boldsymbol{x}-\mathbf{s}_k), \tag{6.17}$$

where

$$G(\boldsymbol{x})=\frac{1}{4\pi\epsilon_1}\frac{e^{-\frac{\kappa_1}{\sqrt{\epsilon_1}}|\boldsymbol{x}|}}{|\boldsymbol{x}|} \tag{6.18}$$

and the mollifier function

$$\begin{cases} m(\boldsymbol{x})=1-\left(\frac{|\boldsymbol{x}|}{R_0}\right)^3\left(4-3\frac{|\boldsymbol{x}|}{R_0}\right), & |\boldsymbol{x}|<R_0,\\ m(\boldsymbol{x})=0. & |\boldsymbol{x}|>R_0. \end{cases} \tag{6.19}$$

By above definitions, it can be verified easily that $\bar{u}(\boldsymbol{x})$ satisfies

$$\begin{cases} -\Delta\bar{u}(\boldsymbol{x})+\kappa^2\bar{u}(\boldsymbol{x})=\sum_{k=1}^{K}q_k\delta(\boldsymbol{x}-\mathbf{s}_k)+\sum_{k=1}^{K}q_kF(|\boldsymbol{x}-\mathbf{s}_k|), & \boldsymbol{x}\in\Omega_0,\\ \bar{u}=\frac{\partial\bar{u}}{\partial n}=0, & \boldsymbol{x}\in\partial\Omega_0,\\ \bar{u}(\boldsymbol{x})=0, & \boldsymbol{x}\in\Omega_0^c. \end{cases} \tag{6.20}$$

Next, we define

$$w(x)=u(x)-\bar{u}(x)\chi_{\Omega_0}(x), \tag{6.21}$$

which will satisfy the following equations without singularities

$$-\epsilon(x)\triangle w(x)+\kappa^2(x)w(x)=f(x)\chi_{\Omega_0}(x), \tag{6.22}$$

where

$$f(x)=-\sum_{k=1}^{K}q_kF(|x-\mathbf{s}_k|), \tag{6.23}$$

$$\begin{cases} F(r)=\frac{3e^{-\frac{\kappa_1}{\sqrt{\epsilon_1}}r}}{\pi R_0^4}(2R_0-3r+2\frac{\kappa_1}{\sqrt{\epsilon_1}}r^2-2R_0\frac{\kappa_1}{\sqrt{\epsilon_1}}r), & r<R_0, \\ F(r)=0, & r>R_0. \end{cases} \tag{6.24}$$

We will present the numerical results for Eq. (6.22).

Example 1. In the first example, we choose $\Omega=[-1,1]^3$. Ω_1 is a ball with center $(0,0,0)$ and radius $R=0.7$. Parameters are chosen as

$$\mathbf{s}=(0,0,0),\ \ q=1,\ \ R_0=0.5,\ \ \epsilon(\boldsymbol{x})=1\ \text{ for }\ \boldsymbol{x}\in\Omega_1,\ \ \epsilon(\boldsymbol{x})=80\ \text{ for }\ \boldsymbol{x}\in\Omega_2,\ \kappa(\boldsymbol{x})=0.$$

The exact solution is

$$u(x)=\frac{1}{4\pi|x|\epsilon_1}-\Big(\frac{1}{\epsilon_1}-\frac{1}{\epsilon_2}\Big)\frac{1}{4\pi R},\quad \boldsymbol{x}\in\Omega_1;\qquad u(x)=\frac{1}{4\pi|x|\epsilon_2},\quad \boldsymbol{x}\in\Omega_2, \tag{6.25}$$

and, correspondingly

$$\begin{cases} w(x)=\frac{1}{4\pi\epsilon_1|x|}\Big(\frac{|x|}{R_0}\Big)^3\Big(4-3\frac{|x|}{R_0}\Big)-(\frac{1}{\epsilon_1}-\frac{1}{\epsilon_2})\frac{1}{4\pi R}, & |x|<R_0, \\ w(x)=\frac{1}{4\pi|x|\epsilon_1}-(\frac{1}{\epsilon_1}-\frac{1}{\epsilon_2})\frac{1}{4\pi R}, & R_0<|x|<R, \\ w(x)=\frac{1}{4\pi|x|\epsilon_2}, & |x|>R. \end{cases}$$

Example 2. In the second example, we choose $\Omega=[-1,1]^3$. The domain is constructed as follows. We choose a large ball with center $(0,0,0)$ and radius 0.7. 20 points are randomly selected on the surface of the large ball as the centers of small balls. Radii of the small balls are randomly sampled from $[0.1,0.3]$. Ω_1 is the union of these balls.

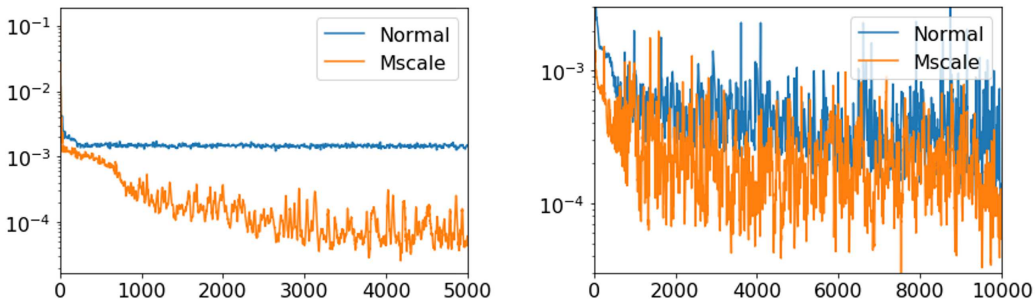
The singular source term in (6.16) is constructed as follows. The position of each charge is randomly selected in the ball with center $(0,0,0)$ and radius 0.5 and the quantity of charges is from $[-0.5,0.5]$. We choose $R_0=0.2$. Parameters are chosen as

$$\epsilon(\boldsymbol{x})=1\ \text{ for }\ \boldsymbol{x}\in\Omega_1,\ \ \epsilon(\boldsymbol{x})=80\ \text{ for }\ \boldsymbol{x}\in\Omega_2,\ \kappa(\boldsymbol{x})=0.$$

The reference solution is again calculated by a FDM with a very fine mesh.

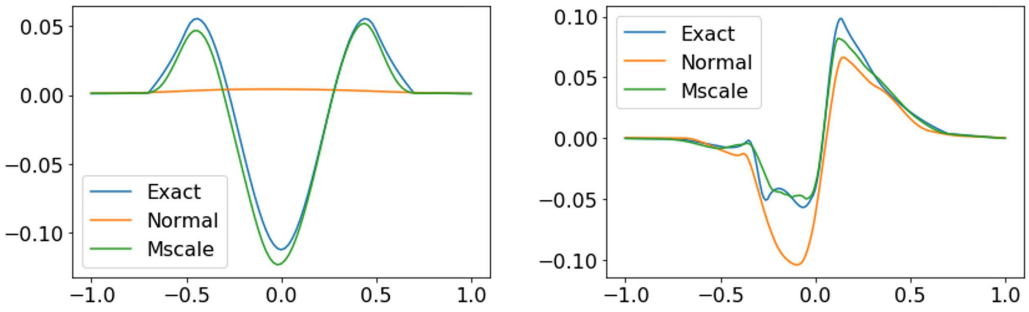
DNN results. In each training epoch, we sample 6000 points inside the domain Ω and 3000 points on boundary $\partial\Omega$. In Example 1, we examine the following two structures:

1. fully-connected DNN with size **1-1000-1000-1000-1** (**normal**);



(a) Example 1 (b) Example 2

Figure 28: Error vs. epoch for the PDEs in domain with geometric and source singularities.



(a) Example 1 (b) Example 2

Figure 29: Numerical solutions on line $x_2=x_3=0$.

2. MscaleDNN-2 with five subnetworks with size **1-200-200-200-1**, and scale coefficients of $\{1,2,4,8,16\}$ (**Mscale**).

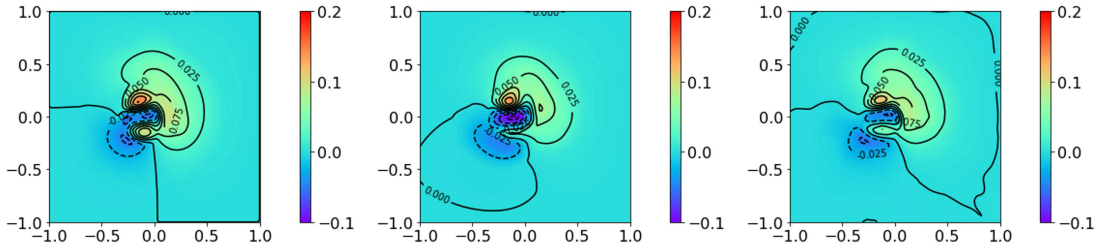
In Example 2, the equation is more complex than before, we need more neurons to approximate the complex solution. In Example 2, we examine the following two structures with boundary penalty $\beta=100$:

1. fully-connected DNN with size **1-1500-1000-1000-500-1** (**normal**);

2. MscaleDNN-2 with five subnetworks with size **1-300-200-200-100-1**, and scale coefficients of $\{1,2,4,8,16\}$ (**Mscale**).

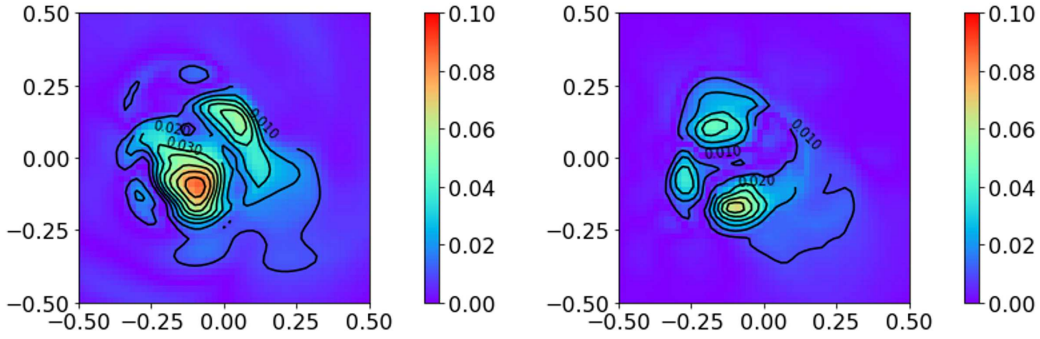
As shown in Fig. 28, the errors of the MscaleDNN decays much faster and achieves much smaller errors after training for both examples.

The numerical solutions on the line $x_2=x_3=0$ obtained by the FDM ($h=0.01$), normal DNN (10000 epochs) and MscaleDNN (10000 epochs) are shown in Fig. 29. The output of the normal fully connected network can not capture the peaks in exact solution very well.



(a) FDM (b) Normal (c) Mscale

Figure 30: Numerical solutions of Example 2 on plane $x_3=0$.



(a) Normal (b) Mscale

Figure 31: Errors of Example 2 on plane $x_3=0$.

For the second example, the numerical solutions and errors on the surface $x_3=0$ around the bio-molecule obtained by FDM ($h=0.01$), normal DNN (10000 epochs) and MscaleDNN (10000 epochs) are shown in Fig. 30 and Fig. 31.

7 Conclusion and future work

In this paper, we have introduced a new kind of multi-scale DNNs, using a frequency domain scaling technique and compactly supported activation functions, to generate a multi-scale capability for finding the solutions of elliptic PDEs with rich frequency contents. By using a radial scaling in the Fourier domain of the solutions, the MscaleDNN is shown to be an efficient mesh-less and easy-to-implement method for PDEs on complex and singular domains, for which solvers by finite element and finite difference methods may be costly due to the need of mesh generations and solution of large linear systems.

For future work, we will also explore the idea of activation function with the mother wavelet properties as proposed in [5], which should give further frequency localization and separation capability in the MscaleDNNs. Applications of the MscaleDNN to large scale computational engineering problems will be carried out, especially, in comparison

with finite element and finite difference methods. More importantly, an area to be explored is to apply the MscaleDNN to high dimensional PDEs such as Schrodinger equations for many body quantum systems, issues of high dimensional sampling and low dimensional structure of solutions will be studied.

Acknowledgments

W.C. is supported by US National Science Foundation (Grant No. DMS-1950471). Z.X. is supported by National Key R&D Program of China (2019YFA0709503), Shanghai Sailing Program, Natural Science Foundation of Shanghai (20ZR1429000), and HPC of School of Mathematical Sciences at Shanghai Jiao Tong University.

References

[1] N. A. Baker, D. Sept, S. Joseph, M. J. Holst, and J. A. McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proceedings of the National Academy of Sciences*, 98(18):10037–41, 2001.

[2] R. Basri, D. Jacobs, Y. Kasten, and S. Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *arXiv preprint arXiv:1906.00425*, 2019.

[3] W. Cai. *Computational Methods for Electromagnetic Phenomena, electrostatics in solvation, scatterings, and electron transport*. Cambirdge University Press, 2013.

[4] W. Cai, X. Li, and L. Liu. A phase shift deep neural network for high frequency approximation and wave problems. *to appear in SIAM J. Scientific Computing, arXiv:1909.11759*, 2019.

[5] W. Cai and Z.-Q. J. Xu. Multi-scale deep neural networks for solving high dimensional pdes. *Arxiv preprint, arXiv:1910.11710*, 2019.

[6] Y. Cao, Z. Fang, Y. Wu, D.-X. Zhou, and Q. Gu. Towards Understanding the Spectral Bias of Deep Learning. *arXiv:1912.01198 [cs, stat]*, 2020.

[7] I.-L. Chern, J.-G. Liu, and W.-C. Wang. Accurate evaluation of electrostatics for macromolecules in solution. *Meth. Appl. Anal.*, 10:309–328, 2003.

[8] I. Daubechies. *Ten lectures on wavelets*, volume 61. Siam, 1992.

[9] M. Deng, S. Li, and G. Barbastathis. Learning to synthesize: splitting and recombining low and high spatial frequencies for image recovery. *arXiv preprint arXiv:1811.07945*, 2018.

[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. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[11] W. E and B. 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.

[12] A. Hamilton, T. Tran, M. B. Mckay, B. Quiring, and P. S. Vassilevski. Dnn approximation of nonlinear finite element equations. Technical report, Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), 2019.

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

[14] J. Han, L. Zhang, R. Car, et al. Deep potential: A general representation of a many-body potential energy surface. *Communications in Computational Physics*, 23(3):629–639, 2018.

[15] J. He, L. Li, J. Xu, and C. Zheng. Relu deep neural networks and linear finite elements. *arXiv preprint arXiv:1807.03973*, 2018.
[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[17] Y. Liao and P. Ming. Deep nitsche method: Deep ritz method with essential boundary conditions. *arXiv preprint arXiv:1912.01309*, 2019.
[18] S. Lindskog. Structure and mechanism of carbonic anhydrase. *Pharmacol. Therapeut.*, 74:1–20, 1997.
[19] T. Luo, Z. Ma, Z.-Q. J. Xu, and Y. Zhang. Theory of the frequency principle for general deep neural networks. *arXiv preprint arXiv:1906.09235*, 2019.
[20] J. Pan, S. Liu, D. Sun, J. Zhang, Y. Liu, J. Ren, Z. Li, J. Tang, H. Lu, Y.-W. Tai, et al. Learning dual convolutional neural networks for low-level vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3070–3079, 2018.
[21] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the Spectral Bias of Neural Networks. In *International Conference on Machine Learning*, pages 5301–5310, 2019.
[22] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
[23] U. Shaham, A. Cloninger, and R. R. Coifman. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537–557, 2018.
[24] C. M. Strofer, J.-L. Wu, H. Xiao, and E. Paterson. Data-driven, physics-based feature extraction from fluid flow fields using convolutional neural networks. *Communications in Computational Physics*, 25(3):625–650, 2019.
[25] Z. Wang and Z. Zhang. A mesh-free method for interface problems using the deep learning approach. *Journal of Computational Physics*, 400:108963, 2020.
[26] C.-Y. Wu, R. Girshick, K. He, C. Feichtenhofer, and P. Krahenbuhl. A multigrid method for efficiently training video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 153–162, 2020.
[27] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *Accepted by Communications in Computational Physics, arXiv:1901.06523*, 2019.
[28] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao. Training Behavior of Deep Neural Network in Frequency Domain. In *Neural Information Processing*, Lecture Notes in Computer Science, pages 264–274, 2019.
[29] W. Geng, S. Yu, and G. Wei. Treatment of geometric singularities in implicit solvent models. *J. Chem. Phys.*, 126:244108, 2007.
[30] Y. Zhang, Z.-Q. J. Xu, T. Luo, and Z. Ma. Explicitizing an implicit bias of the frequency principle in two-layer neural networks. *arXiv preprint arXiv:1905.10264*, 2019.