ON SPECTRAL BIAS REDUCTION OF MULTI-SCALE NEURAL NETWORKS FOR REGRESSION PROBLEMS [*]

BO WANG[†], HENG YUAN[‡], LIZUO LIU[§], WENZHONG ZHANG[¶], AND WEI CAI[‖]

Abstract. In this paper, we derive diffusion equation models in the spectral domain for the evolution of training errors of two-layer multi-scale deep neural networks (MscaleDNN) [6, 30], designed to reduce the spectral bias of fully connected deep neural networks in approximating oscillatory functions. The diffusion models are obtained from the spectral form of the error equation of the MscaleDNN, derived with a neural tangent kernel approach and gradient descent training and a sine activation function, assuming a vanishing learning rate and infinite network width and domain size. The involved diffusion coefficients are shown to have larger supports if more scales are used in the MscaleDNN, and thus, the proposed diffusion equation models in the frequency domain explain the MscaleDNN's spectral bias reduction capability. Numerical results of the diffusion models for a two-layer MscaleDNN training match with the error evolution of actual gradient descent training with a reasonably large network width, thus validating the effectiveness of the diffusion models. Meanwhile, the numerical results for MscaleDNN show error decay over a wide frequency range and confirm the advantage of using the MscaleDNN in approximating functions with a wide range of frequencies.

Key words. multi-scale deep neural network, spectral bias, diffusion equation, gradient descent method.

AMS subject classifications. 35Q68, 65N35, 65T99,65K10

1. Introduction.

Deep learning algorithms have achieved great success in computer vision [20, 42, 44], natural language processing [53, 35, 22] and many other areas. Their computational power with the help of graphics processing units (GPUs) and capability of handling high dimensional problems have led the computational community to investigate their potentials in applied mathematics research. As a result, a new research field known as scientific machine learning has became active in the past few years.

One important task in scientific machine learning is to use deep neural networks (DNNs) to approximate functions or solutions of partial differential equations (PDEs). The idea of using neural networks to solve PDEs goes back to the 1990's [9, 21]. In general, four categories of deep PDE solvers have been investigated. The first category is to use deep neural networks to improve classical numerical methods [14, 17, 46]. In the second category, the solution operators between infinite-dimensional spaces are approximated by neural networks [1, 27, 26]. In the third category, the deep neural networks are utilized to approximate the solutions of PDEs directly such as the physics-informed neural networks (PINNs) [29, 38, 39], the deep Ritz method [11, 34, 28, 18], and Garkerkin methods with weak adversarial networks (WAN) [54, 7].

[*] October 16, 2023, submitted to SIAM J. Numerical Analysis

[†] LCSM, Ministry of Education, School of Mathematics and Statistics, Hunan Normal University, Changsha, Hunan 410081, P. R. China. The author acknowledges the financial support provided by NSFC (grant 12022104,12371394).

[‡] LCSM, Ministry of Education, School of Mathematics and Statistics, Hunan Normal University, Changsha, Hunan 410081, P. R. China.

[§] Department of Mathematics, Southern Methodist University, Dallas, TX 75275.

[¶] Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, Jiangsu 215000, China. The author acknowledges the financial support provided by NSFC (grant 12201603)

[‖] Corresponding author, Department of Mathematics, Southern Methodist University, Dallas, TX 75275 (cai@smu.edu). The author acknowledges the financial support provided by the US National Science Foundation grant DMS-2207449.

Lastly, Feynman-Kac formula approaches utilize the connection between linear and nonlinear PDEs and (backward) stochastic differential equations to construct loss functions for the learning algorithms [10, 16, 15, 3, 56, 4].

Despite their many successes for a wide range of applications, recent studies on the convergence of the deep learning algorithms in theories and practical computations show that standard fully connected DNNs have difficulties in learning high frequency functions, a phenomenon referred as "spectral bias" [37] or "F-Principle" [51, 52] in the literature. To overcome this bias, several strategies have been introduced to design neural networks with better frequency resolution, producing promising results. For solving PDEs, a multi-scale DNN (MscaleDNN) [6, 30, 47, 31, 25, 55], which consists of a series of parallel fully connected sub-neural networks receiving scaled inputs, has been proposed to learn highly oscillating solutions. Each individual scaled sub-network in the MscaleDNN is designed to approximate a segment of frequency content of the target function, and the effect of the scaling is to convert a specific range of high frequency content to a lower one so that the learning can be accomplished much faster. It was also proposed in [6, 30] that the MscaleDNN should use activation functions with a localized frequency profile, such as the sine function and compact supported functions (hat functions and B-splines, etc). In a related work for image and 3D shape reconstruction, the Fourier feature networks, which in fact can be obtained from the MscaleDNN with a sine activation function in their first layer, use the sinusoidal mapping on their inputs and dramatically improve the performance of learning [33, 57, 43].

To analyze the convergence of deep learning algorithms, the neural tangent kernel (NTK), introduced in [19], has been a very effective tool to study the evolution of DNNs in function spaces during training [2, 23, 12, 32, 36], and the eigenvector space for the NTK provides many information on the convergence of the DNNs. The convergence and spectral bias of the standard fully connected models and the improved performance of the afore-mentioned Fourier feature embedded neural networks can be explained by using the NTK. In fact, the NTK theory suggests that standard fully connected DNN have a kernel with a rapid frequency falloff, which prevents them from being able to represent the high-frequency contents of target functions effectively. Fourier feature embedded neural networks have been designed to modify the Fourier spectrum of the NTK so that a faster training convergence for high frequency components can be achieved [33, 57, 43, 48].

Most of the convergence analysis so far has been done in the physical domain [43, 32, 36, 40, 23]. Some explicit formulas of NTKs have been reported for two layers neural networks with the ReLU activation function [49, 8, 40, 2, 50, 45]. The behaviors of the NTK are usually obtained by analyzing the eigenvalues of the corresponding Gram matrix [43, 36]. In this paper, in order to illuminate the mechanism behind the observed reduced spectral bias in the convergence of the MscaleDNN in approximating highly oscillatory functions and PDE solutions [6, 30, 47], we will derive an error diffusion model, using the NTK approach, in the spectral domain for a two-layered MscaleDNN with a sine activation function for the case of vanishing learning rate and infinite network width and domain size. Our contribution is three folds: i) we prove that the gradient descent training is equivalent to a diffusion problem in the Fourier spectral domain; ii) the diffusion coefficients can be determined by the Fourier transform of the NTK; iii) the MscaleDNNs with more scales result in diffusion coefficients with larger value and support in the frequency domain. Therefore, our theoretical results provide clear a mathematical explanation why the MscaleDNN can learn much faster over a wider range of frequency. Also, due to the connection between the MscaleDNN and

Fourier feature network [43], the presented theory can be applied to the latter, as well.

The rest of the paper is organized as follows. In Section 2, a brief review of the MscaleDNN is given. Section 3 derives the diffusion equation models for the training error of high dimensional fitting problem. Analysis of the spectral bias reduction of a two layer MscaleDNN will be done by solving the error diffusion equation models using a Hermite spectral method in Section 4. The numerical results show that the MscaleDNN leads to faster convergence over wider range of frequencies when the number of scales is increased. Finally, section 5 gives a conclusion and future work.

2. A review of the multi-scale DNN (MscaleDNN).

The frequency bias behavior of the deep learning algorithms [37, 51] has inspired the development and usage of the MscaleDNN in various applications. The MscaleDNN is simply a combination of several fully connected DNNs with different scales on their inputs. It is very convenient to replace a fully connected DNN by a MscaleDNN with equal number of total neurons in a deep learning algorithm while much better results can be expected. The main idea of the MscaleDNN is to do a radial scaling in the frequency domain such that the learning is performed on functions of scaled-down frequency ranges [30, 47, 55].

To illustrate the idea, let us consider the DNN approximation of a given band-limited target function $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^d$, whose Fourier transform

$$\hat{f}(\boldsymbol{\xi}) := \mathcal{F}[f](\boldsymbol{\xi}) = \int_{\mathbb{R}^d} f(\boldsymbol{x}) e^{-\mathrm{i}2\pi\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{x}} \mathrm{d}\boldsymbol{x}, \tag{2.1}$$

has a compact support, i.e.,

$$\mathrm{supp}\hat{f}(\boldsymbol{\xi}) \subset B_K(\mathbf{0}) := \{\boldsymbol{\xi} \in \mathbb{R}^d, |\boldsymbol{\xi}| \leq K\}. \tag{2.2}$$

Note that the hyper-sphere $B_K(\mathbf{0})$ in the frequency domain can be partitioned into a union of $s+1$ concentric annulus with uniform or non-uniform radial dimension, e.g., for the case of uniform radial dimension,

$$B_K(\mathbf{0}) = \bigcup_{j=0}^{s} A_j, \quad A_j := \Big\{\boldsymbol{\xi} \in \mathbb{R}^d, \frac{jK}{s+1} \leq |\boldsymbol{\xi}| < \frac{(j+1)K}{s+1}\Big\}. \tag{2.3}$$

Then, the target function in the frequency domain has a decomposition

$$\hat{f}(\boldsymbol{\xi}) = \sum_{j=0}^{s} I_{A_j}(\boldsymbol{\xi})\hat{f}(\boldsymbol{\xi}) := \sum_{j=0}^{s} \hat{f}_j(\boldsymbol{\xi}), \tag{2.4}$$

where $I_{A_j}(\boldsymbol{\xi})$ is the indicator function of the set A_j. From its definition, the component $\hat{f}_j(\boldsymbol{\xi})$ has a $\mathrm{supp}\hat{f}_j(\boldsymbol{\xi}) \subset A_j$, for $j = 0, 1, \cdots, s$. A corresponding decomposition of (2.4) in the physical domain is given by

$$f(\boldsymbol{x}) = \sum_{j=0}^{s} f_j(\boldsymbol{x}), \tag{2.5}$$

with $f_j(\boldsymbol{x})$ being the inverse Fourier transform

$$f_j(\boldsymbol{x}) = \mathcal{F}^{-1}[\hat{f}_j](\boldsymbol{x}) := \int_{\mathbb{R}^d} \hat{f}_j(\boldsymbol{\xi}) e^{\mathrm{i}2\pi\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{x}} \mathrm{d}\boldsymbol{\xi}. \tag{2.6}$$

With the decomposition (2.4), an appropriate scaling can be used to transform the component $\hat{f}_j(\boldsymbol{\xi})$ from the high frequency region A_j to a low frequency region A_j/α_j. The scaled version of $\hat{f}_j(\boldsymbol{\xi})$ is defined as

$$\hat{f}_j^{(\text{scale})}(\boldsymbol{\xi}) = \hat{f}_j(\alpha_j \boldsymbol{\xi}), \tag{2.7}$$

where $\alpha_j > 1$ is an appropriate scaling factor for A_j. By the identity

$$\mathcal{F}[g(a\boldsymbol{x})](\boldsymbol{\xi}) = \left(\frac{1}{|a|}\right)^d \mathcal{F}[g]\left(\frac{\boldsymbol{\xi}}{a}\right), \tag{2.8}$$

the scaling (2.7) in the frequency domain leads to

$$f_j^{(\text{scale})}(\boldsymbol{x}) := \mathcal{F}^{-1}[\hat{f}_j^{(\text{scale})}](\boldsymbol{x}) = \frac{1}{\alpha_j^d} f_j\left(\frac{\boldsymbol{x}}{\alpha_j}\right), \tag{2.9}$$

or equivalently

$$f_j(\boldsymbol{x}) = \alpha_j^d f_j^{(\text{scale})}(\alpha_j \boldsymbol{x}). \tag{2.10}$$

By choosing an appropriate scale α_j, we are able to make the Fourier spectrum of $\hat{f}_j^{\text{scale}}(\boldsymbol{\xi})$ into a lower frequency range, i.e.,

$$\text{supp} \hat{f}_j^{\text{scale}}(\boldsymbol{\xi}) \subset \left\{ \boldsymbol{\xi} \in \mathbb{R}^d, \frac{jK}{(s+1)\alpha_j} \leq |\boldsymbol{\xi}| < \frac{(j+1)K}{(s+1)\alpha_j} \right\}. \tag{2.11}$$

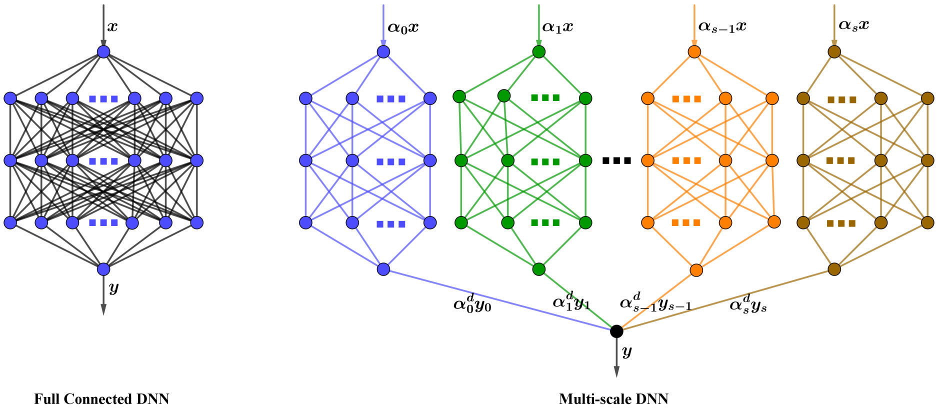

$\boldsymbol{x}$
$\boldsymbol{y}$
$\boldsymbol{\alpha_0 x}$
$\boldsymbol{\alpha_1 x}$
$\boldsymbol{\alpha_{s-1} x}$
$\boldsymbol{\alpha_s x}$
$\boldsymbol{\alpha_0^d y_0}$
$\boldsymbol{\alpha_1^d y_1}$
$\boldsymbol{\alpha_{s-1}^d y_{s-1}}$
$\boldsymbol{\alpha_s^d y_s}$
$\boldsymbol{y}$
Full Connected DNN
Multi-scale DNN

Fig. 2.1: Sketches of a fully connected DNN and a Multi-scale DNN.

As a result of the spectral bias of DNN, a fully connected DNN $f(\boldsymbol{x}; \boldsymbol{\theta}_j)$ with parameters $\boldsymbol{\theta}_j$ can be trained to learn $f_j^{(\text{scale})}(\boldsymbol{x})$ very fast if $(j+1)K/((s+1)\alpha_j)$ is small enough. Therefore, the decomposition (2.5) and scaling formula (2.10) implies that a deep learning algorithm using a neural network in the form

$$\mathcal{N}_s(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{j=0}^{s} \alpha_j^d f(\alpha_j \boldsymbol{x}; \boldsymbol{\theta}_j) \tag{2.12}$$

can be expected to have a more uniform convergence and less spectral bias, i.e., frequency uniform approximation to any band-limited function $f(\boldsymbol{x})$. Deep neural networks defined by (2.12) are named as the MscaleDNN and a schematic comparison between fully connected DNN and MscaleDNN is shown in Fig. 2.1.

Previous work presented in [6, 30, 47] have shown that the MscaleDNN can reduce spectral bias significantly in learning highly oscillatory functions, however, mathematical analysis on the mechanism has not been presented in the literature. The following analysis will build a foundation for the MscaleDNN and provide a strategy to manipulate the neural networks.

3. Error diffusion equation model of a two-layer MscaleDNN. In this section, the convergence of a machine learning algorithm for d-dimensional regression problems with two layers multi-scale neural networks is analyzed. We will show that the evolution of the error can be modeled by a diffusion equation in the Fourier frequency domain as the width of the network goes to infinity and learning rate approaches to zero.

Consider a regression problem with an objective function $y = f(\boldsymbol{x})$ defined in a bounded domain $\Omega \subset \mathbb{R}^d$. The machine learning algorithm with a neural network denoted by $\mathcal{N}(\boldsymbol{x}, \boldsymbol{\theta})$ and mean square loss

$$L(\boldsymbol{\theta}) = \frac{1}{2}\int_{\Omega} |\mathcal{N}(\boldsymbol{x}, \boldsymbol{\theta}) - f(\boldsymbol{x})|^2 d\boldsymbol{x}, \tag{3.1}$$

will be discussed in the following analysis.

The gradient descent dynamics based on the loss functional (3.1) is

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \tau \nabla L(\boldsymbol{\theta}^{(k)}), \tag{3.2}$$

where τ is the learning rate. By regarding τ as the time step size, the continuum limit dynamics at $\tau \to 0$ is

$$\frac{\mathrm{d}\boldsymbol{\theta}(t)}{\mathrm{d}t} = -\nabla L(\boldsymbol{\theta}(t)). \tag{3.3}$$

With the mean square loss function (3.1) and the chain rule of differentiation, we obtain

$$\begin{aligned}
\partial_t \mathcal{N}(\boldsymbol{x}, \boldsymbol{\theta}) =& [\nabla_{\boldsymbol{\theta}} \mathcal{N}(\boldsymbol{x}, \boldsymbol{\theta})]^{\mathrm{T}} \frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} \\
=& -\int_{\Omega} (\nabla_{\theta} \mathcal{N}(\boldsymbol{x}, \boldsymbol{\theta}))^{\mathrm{T}} \nabla_{\boldsymbol{\theta}} \mathcal{N}(\boldsymbol{x}', \theta)(\mathcal{N}(\boldsymbol{x}', \theta) - f(\boldsymbol{x}')) d\boldsymbol{x}' \\
:=& -\int_{\Omega} \Theta(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})(\mathcal{N}(\boldsymbol{x}', \boldsymbol{\theta}) - f(\boldsymbol{x}')) d\boldsymbol{x}',
\end{aligned} \tag{3.4}$$

for the dynamics of the network function $\mathcal{N}(\boldsymbol{x}, \theta)$, where

$$\Theta(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = (\nabla_{\theta} \mathcal{N}(\boldsymbol{x}, \boldsymbol{\theta}))^{\mathrm{T}} \nabla_{\boldsymbol{\theta}} \mathcal{N}(\boldsymbol{x}', \theta), \tag{3.5}$$

is the neural tangent kernel (NTK) proposed in [19].

A multi-scale neural network with one hidden layer (see. Fig. 2.1 (right)) is given as

$$\mathcal{N}_s(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{\sqrt{N}} \sum_{j=0}^{s} \alpha_j^d \sum_{k=1}^{q} \sigma(\boldsymbol{\theta}_{jq+k}^{\mathrm{T}} \alpha_j \boldsymbol{x} + b_{jq+k}), \quad \boldsymbol{x} \in \Omega := [-1, 1]^d, \tag{3.6}$$

where $s+1$ is the number of scales, $\{\alpha_j\}_{j=0}^{s}$ are the scaling factors, q is the number of neurons for each scale, $N=(s+1)q$ is the total number of neurons in the hidden layer. Apparently, the network includes the standard fully connected neural network with one hidden layer as a special case of $s=0$. For this two-layer multi-scale neural network, a direct calculation gives its NTK

(3.7)
$$\Theta_s(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta}) = \sum_{j=0}^{s} \frac{\alpha_j^{2d}(\alpha_j^2 \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'+1)}{N} \sum_{k=1}^{q} \sigma'(\boldsymbol{\theta}_{jq+k}^{\mathrm{T}}\alpha_p \boldsymbol{x} + b_{jq+k})\sigma'(\boldsymbol{\theta}_{jq+k}^{\mathrm{T}}\alpha_j \boldsymbol{x}' + b_{jq+k}).$$

Setting the activation function

$$\sigma(x) = \sin(x) \tag{3.8}$$

and assuming all the parameters $\{\theta_{jk}\}$ in $\boldsymbol{\theta}_j = (\theta_{j1},\theta_{j2},\cdots,\theta_{jd})^{\mathrm{T}}$, $\{b_j\}$ are independent random variables of normal distribution, then, by the law of large numbers and identity

$$(2\pi)^{-\frac{d+1}{2}} \int_{\mathbb{R}^{d+1}} e^{\mathrm{i}(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}+yb)} e^{-\frac{|\boldsymbol{\theta}|^2+b^2}{2}} d\boldsymbol{\theta} db = e^{-\frac{|\boldsymbol{x}|^2+y^2}{2}}, \quad \forall \boldsymbol{x}\in\mathbb{R}^d, \;\; y\in\mathbb{R},$$

we have

$$\begin{aligned} \lim_{q\to\infty} \Theta_s(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta}) &= \sum_{j=0}^{s} \frac{\alpha_j^{2d}(\alpha_j^2 \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'+1)}{s+1} \mathbb{E}(\cos(\boldsymbol{\theta}_1^{\mathrm{T}}\alpha_j\boldsymbol{x}+b_1)\cos(\boldsymbol{\theta}_1^{\mathrm{T}}\alpha_j\boldsymbol{x}'+b_1)) \\ &= \sum_{j=0}^{s} \frac{\alpha_j^{2d}(\alpha_j^2 \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'+1)}{2(s+1)} \left[e^{-2} e^{-\frac{\alpha_j^2|\boldsymbol{x}+\boldsymbol{x}'|^2}{2}} + e^{-\frac{\alpha_j^2|\boldsymbol{x}-\boldsymbol{x}'|^2}{2}} \right]. \end{aligned} \tag{3.9}$$

Apparently, $\{\boldsymbol{\theta}_1, b_1\}$ can replaced by the parameters of any neuron in the hidden layer. According to the analysis in [19], the NTK will be static during the training assuming the width of the neural network tends to infinity. In addition, the limit NTK is also a convolution kernel as presented in [8, 40]. Suppose $\boldsymbol{x}$ and $\boldsymbol{x}'$ are located on the unit sphere, i.e., $|\boldsymbol{x}| = |\boldsymbol{x}'| = 1$, then the limit NTK is a function of the angle β between $\boldsymbol{x}$ and $\boldsymbol{x}'$. The NTKs of some multi-scale neural networks with finite width are compared with their infinite width limit in Fig. 3.1. We can see that the NTK (3.7) has an limit given above as $q\to\infty$. In order to validate the static property of the limit NTK,

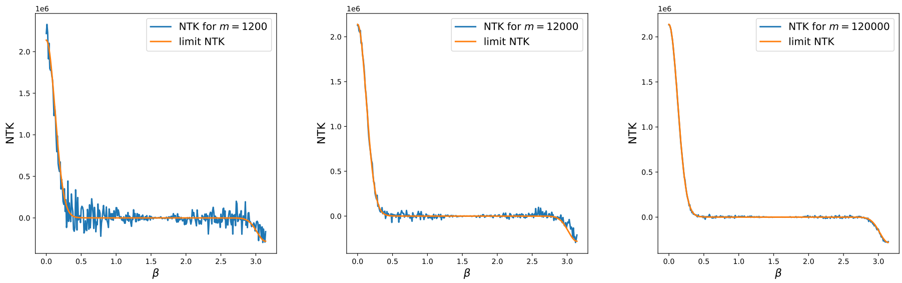


Fig. 3.1: The NTKs in (3.7) and the limit NTK in (3.9) ($d=3, s=3$).

we train a multi-scale neural network with $s = 3$, $N = 12000$ to fit a 3-dimensional function in the domain $[-1, 1]^3$. The scaling parameters α_p are set to be 2^p. The NTKs of the multi-scale neural network after training $1000, 2000, 5000$ epochs are compared with the limit NTK in Fig. 3.2. The results clearly show that the NTK is static during training.

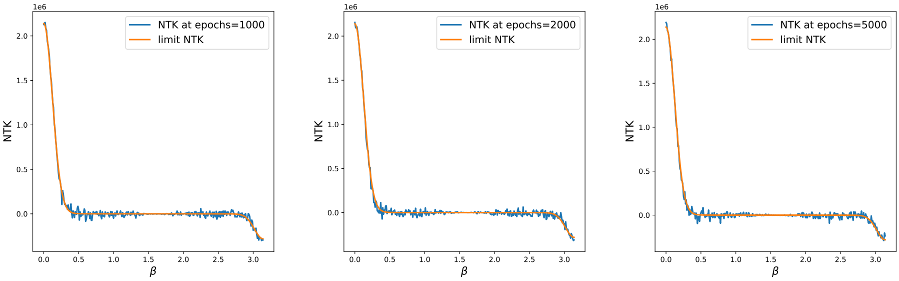


Fig. 3.2: The NTKs in (3.7) with $d = 3, s = 3, N = 12000$ during training.

Consequently, as the width of the network goes to infinity, the dynamics of the gradient descent learning (3.4) tends to

$$\begin{aligned} &\partial_t(\mathcal{N}_s(\boldsymbol{x},\boldsymbol{\theta}) - f(\boldsymbol{x})) \\ &= -\sum_{j=0}^{s} \frac{\alpha_j^{2(d+1)}\boldsymbol{x}^{\mathrm{T}}}{2(s+1)} \int_\Omega \left[e^{-2}\mathcal{G}_j(\boldsymbol{x}+\boldsymbol{x}') + \mathcal{G}_j(\boldsymbol{x}-\boldsymbol{x}')\right]\boldsymbol{x}'(\mathcal{N}_s(\boldsymbol{x}',\boldsymbol{\theta}) - f(\boldsymbol{x}'))\mathrm{d}\boldsymbol{x}' \\ &\quad -\sum_{j=0}^{s} \frac{\alpha_j^{2d}}{2(s+1)} \int_\Omega \left[e^{-2}\mathcal{G}_j(\boldsymbol{x}+\boldsymbol{x}') + \mathcal{G}_j(\boldsymbol{x}-\boldsymbol{x}')\right](\mathcal{N}_s(\boldsymbol{x}',\boldsymbol{\theta}) - f(\boldsymbol{x}'))\mathrm{d}\boldsymbol{x}', \end{aligned} \tag{3.10}$$

where

$$\mathcal{G}_j(\boldsymbol{x}) := e^{-\alpha_j^2|\boldsymbol{x}|^2/2}, \quad \boldsymbol{x} \in \mathbb{R}^d, \tag{3.11}$$

is the scaled Gaussian function.

Next, we define a zero extension of the error function by

$$\eta(\boldsymbol{x},\boldsymbol{\theta}) = \begin{cases} 0, & \boldsymbol{x} \notin \Omega, \\ \mathcal{N}_s(\boldsymbol{x},\boldsymbol{\theta}) - f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \end{cases} \tag{3.12}$$

then, the dynamic system (3.10) can be rewritten as

$$\begin{aligned} &\partial_t \eta(\boldsymbol{x},\boldsymbol{\theta}) I_\Omega(\boldsymbol{x}) \\ &= -\sum_{j=0}^{s} \frac{I_\Omega(\boldsymbol{x})\alpha_j^{2(d+1)}\boldsymbol{x}^{\mathrm{T}}}{2(s+1)} \int_{\mathbb{R}^d} \left[e^{-2}\mathcal{G}_j(\boldsymbol{x}+\boldsymbol{x}') + \mathcal{G}_j(\boldsymbol{x}-\boldsymbol{x}')\right]\boldsymbol{x}'\eta(\boldsymbol{x}',\boldsymbol{\theta})\mathrm{d}\boldsymbol{x}' \\ &\quad -\sum_{j=0}^{s} \frac{I_\Omega(\boldsymbol{x})\alpha_j^{2d}}{2(s+1)} \int_{\mathbb{R}^d} \left[e^{-2}\mathcal{G}_j(\boldsymbol{x}+\boldsymbol{x}') + \mathcal{G}_j(\boldsymbol{x}-\boldsymbol{x}')\right]\eta(\boldsymbol{x}',\boldsymbol{\theta})\mathrm{d}\boldsymbol{x}', \end{aligned} \tag{3.13}$$

where an indicator function $I_\Omega(\boldsymbol{x})$ is used to extend the equation to the whole space.

Existing works on DNN convergence analysis employ a discrete version of (3.13) in the physical space by analyzing the eigenvalues of the Gram matrix [43, 32, 36, 40, 23]. However, to get a precise information on the spectral bias phenomena, it is more natural to study the convergence behavior in the Fourier domain as follows.

Given any $g(\boldsymbol{x}) \in L^1(\mathbb{R}^d)$, the Fourier transform defined in (2.1) has the following identities

$$\mathcal{F}[\nabla g](\boldsymbol{\xi}) = 2\pi \mathrm{i}\boldsymbol{\xi}\mathcal{F}[g](\boldsymbol{\xi}), \quad \nabla \hat{g}(\boldsymbol{\xi}) = -2\pi\mathrm{i}\mathcal{F}[\boldsymbol{x}g(\boldsymbol{x})](\boldsymbol{\xi}), \quad \forall \boldsymbol{x}g(\boldsymbol{x}) \in (L^1(\mathbb{R}^d))^d, \tag{3.14}$$

and

$$\mathcal{F}[e^{-|\boldsymbol{x}|^2}](\xi) = \pi^{\frac{d}{2}}e^{-\pi^2|\boldsymbol{\xi}|^2}, \quad \mathcal{F}[g(a\boldsymbol{x})](\boldsymbol{\xi}) = \Big(\frac{1}{|a|}\Big)^d \mathcal{F}[g]\Big(\frac{\boldsymbol{\xi}}{a}\Big). \tag{3.15}$$

In addition, given two functions $h(\boldsymbol{x})$, $g(\boldsymbol{x})$, their cross-correlation and convolution are defined as

$$h \star g := \int_{\mathbb{R}^d} \overline{h(\boldsymbol{x}')} g(\boldsymbol{x}+\boldsymbol{x}')\mathrm{d}\boldsymbol{x}', \quad h * g := \int_{\mathbb{R}^d} h(\boldsymbol{x}')g(\boldsymbol{x}-\boldsymbol{x}')\mathrm{d}\boldsymbol{x}', \tag{3.16}$$

and we have teh following identities,

$$\widehat{h\star g}(\boldsymbol{\xi}) = \overline{\hat{h}(\boldsymbol{\xi})}\hat{g}(\boldsymbol{\xi}), \quad \widehat{h * g}(\boldsymbol{\xi}) = \hat{h}(\boldsymbol{\xi})\hat{g}(\boldsymbol{\xi}), \quad \widehat{fg}(\boldsymbol{\xi}) = f * g(\boldsymbol{\xi}). \tag{3.17}$$

Taking Fourier transform (2.1) on both sides of (3.13) with respect to $\boldsymbol{x}$ and then applying (3.14)-(3.17) to rearrange the terms gives a integral-differential equation

$$\begin{aligned}\frac{\partial \hat{\eta}(\boldsymbol{\xi},\boldsymbol{\theta}(t)) * \hat{I}_\Omega(\boldsymbol{\xi})}{\partial t} = \Bigg[&\nabla_{\boldsymbol{\xi}} \cdot \Big[\sum_{j=0}^{s} \frac{\alpha_j^{2(d+1)}\widehat{\mathcal{G}}_j(\boldsymbol{\xi})}{8\pi^2(s+1)}\Big(\nabla_{\boldsymbol{\xi}}\hat{\eta}(\boldsymbol{\xi},\boldsymbol{\theta}(t)) - e^{-2}\nabla_{\boldsymbol{\xi}}\overline{\hat{\eta}(\boldsymbol{\xi},\boldsymbol{\theta}(t))}\Big)\Big] \\ &- \sum_{j=0}^{s}\frac{\alpha_j^{2d}\widehat{\mathcal{G}}_j(\boldsymbol{\xi})}{2(s+1)}[e^{-2}\bar{\hat{\eta}}(\boldsymbol{\xi},\boldsymbol{\theta}(t)) + \hat{\eta}(\boldsymbol{\xi},\boldsymbol{\theta}(t))]\Bigg] * \hat{I}_\Omega(\boldsymbol{\xi}),\end{aligned} \tag{3.18}$$

where

$$\widehat{\mathcal{G}}_j(\boldsymbol{\xi}) = (2\pi)^{\frac{d}{2}}\alpha_j^{-d}e^{-\frac{2\pi^2|\boldsymbol{\xi}|^2}{\alpha_j^2}}. \tag{3.19}$$

The convolution with $\hat{I}_\Omega(\boldsymbol{\xi})$ makes the model too complicate to analyze. Nevertheless, if we consider the limit of infinite large domain, i.e., $\Omega \to \mathbb{R}^d$, the limit of $\hat{I}_\Omega(\boldsymbol{\xi})$ is the Dirac delta function $\delta(\xi)$ and then (3.18) simplifies to

$$\begin{aligned}\frac{\partial \hat{\eta}(\boldsymbol{\xi},\boldsymbol{\theta}(t))}{\partial t} =& \nabla_{\boldsymbol{\xi}} \cdot \Big[\sum_{j=0}^{s} \frac{\alpha_j^{2(d+1)}\widehat{\mathcal{G}}_j(\boldsymbol{\xi})}{8\pi^2(s+1)}\Big(\nabla_{\boldsymbol{\xi}}\hat{\eta}(\boldsymbol{\xi},\boldsymbol{\theta}(t)) - e^{-2}\nabla_{\boldsymbol{\xi}}\overline{\hat{\eta}(\boldsymbol{\xi},\boldsymbol{\theta}(t))}\Big)\Big] \\ &- \sum_{j=0}^{s}\frac{\alpha_j^{2d}\widehat{\mathcal{G}}_j(\boldsymbol{\xi})}{2(s+1)}[e^{-2}\bar{\hat{\eta}}(\boldsymbol{\xi},\boldsymbol{\theta}(t)) + \hat{\eta}(\boldsymbol{\xi},\boldsymbol{\theta}(t))].\end{aligned} \tag{3.20}$$

Define

$$A_s^{\pm}(\boldsymbol{\xi}) = \frac{1 \pm e^{-2}}{8\pi^2(s+1)}\sum_{j=0}^{s}\alpha_j^{2(d+1)}\widehat{\mathcal{G}}_j(\boldsymbol{\xi}), \quad B_s^{\pm}(\boldsymbol{\xi}) = \frac{1 \pm e^{-2}}{2(s+1)}\sum_{j=0}^{s}\alpha_j^{2d}\widehat{\mathcal{G}}_j(\boldsymbol{\xi}), \tag{3.21}$$

and denote by $\hat{\eta}^{\pm}(\boldsymbol{\xi}, \boldsymbol{\theta}(t))$ the real and imaginary parts of $\hat{\eta}(\boldsymbol{\xi}, \boldsymbol{\theta}(t))$, i.e.,

$$\hat{\eta}(\boldsymbol{\xi}, \boldsymbol{\theta}(t)) = \hat{\eta}^{+}(\boldsymbol{\xi}, \boldsymbol{\theta}(t)) + \mathrm{i}\hat{\eta}^{-}(\boldsymbol{\xi}, \boldsymbol{\theta}(t)).$$

(**Diffusion Model**) As the coefficients in (3.20) are real valued functions, we can rewrite (3.20) into two independent equations

$$\partial_t \hat{\eta}^{\pm}(\boldsymbol{\xi}, t) = \nabla_{\boldsymbol{\xi}} \cdot \Big[A_s^{\mp}(\boldsymbol{\xi}) \nabla_{\boldsymbol{\xi}} \hat{\eta}^{\pm}(\boldsymbol{\xi}, t)\Big] - B_s^{\pm}(\boldsymbol{\xi}) \hat{\eta}^{\pm}(\boldsymbol{\xi}, t), \quad \boldsymbol{\xi} \in \mathbb{R}^d, \tag{3.22}$$

with respect to the real and imaginary parts of $\hat{\eta}(\boldsymbol{\xi}, \boldsymbol{\theta}(t))$, respectively.

A simpler diffusion equation can be derived if the bias are set to zero in the network. In fact, a function represented by the network without bias has the form

$$\mathcal{N}_s(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{\sqrt{N}} \sum_{j=0}^{s} \alpha_j^d \sum_{k=1}^{q} \sigma(\boldsymbol{\theta}_{jq+k}^{\mathrm{T}} \alpha_j \boldsymbol{x}), \quad \boldsymbol{x} \in \Omega := [-1, 1]^d, \tag{3.23}$$

and the neural tangent kernel is given by

$$\Theta_s(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = \frac{\boldsymbol{x}^{\mathrm{T}} \boldsymbol{x}'}{N} \sum_{j=0}^{s} \alpha_j^{2(d+1)} \sum_{k=1}^{q} \sigma'(\boldsymbol{\theta}_{jq+k}^{\mathrm{T}} \alpha_j \boldsymbol{x}) \sigma'(\boldsymbol{\theta}_{jq+k}^{\mathrm{T}} \alpha_j \boldsymbol{x}'). \tag{3.24}$$

Setting the activation function $\sigma(x) = \sin(x)$ again, and assuming all the parameters $\{\theta_p\}$ are independent random variables of normal distribution, then, by law of large numbers, we have

$$\begin{aligned} \lim_{q \to \infty} \Theta_s(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) &= \lim_{q \to \infty} \frac{\boldsymbol{x}^{\mathrm{T}} \boldsymbol{x}'}{N} \sum_{j=0}^{s} \alpha_j^{2(d+1)} \sum_{k=1}^{q} \cos(\boldsymbol{\theta}_{jq+k}^{\mathrm{T}} \alpha_j \boldsymbol{x}) \cos(\boldsymbol{\theta}_{jq+k}^{\mathrm{T}} \alpha_j \boldsymbol{x}') \\ &= \frac{\boldsymbol{x}^{\mathrm{T}} \boldsymbol{x}'}{2(s+1)} \sum_{j=0}^{s} \alpha_j^{2(d+1)} \mathbb{E}(\cos(\boldsymbol{\theta}_1^{\mathrm{T}} \alpha_j \boldsymbol{x}) \cos(\boldsymbol{\theta}_1^{\mathrm{T}} \alpha_j \boldsymbol{x}')) \\ &= \frac{\boldsymbol{x}^{\mathrm{T}} \boldsymbol{x}'}{2(s+1)} \sum_{j=0}^{s} \alpha_j^{2(d+1)} \Big[\mathcal{G}_j(\boldsymbol{x} + \boldsymbol{x}') + \mathcal{G}_j(\boldsymbol{x} - \boldsymbol{x}')\Big]. \end{aligned} \tag{3.25}$$

As the width of the network goes to infinity, the dynamics of the gradient descent learning tends to

$$\partial_t \eta(\boldsymbol{x}, \theta) = -\frac{\boldsymbol{x}^{\mathrm{T}}}{2(s+1)} \int_{\Omega} \sum_{j=0}^{s} \alpha_j^{2(d+1))} \Big[\mathcal{G}_j(\boldsymbol{x} + \boldsymbol{x}') + \mathcal{G}_j(\boldsymbol{x} - \boldsymbol{x}')\Big] \boldsymbol{x}' \eta(\boldsymbol{x}', \boldsymbol{\theta}) \mathrm{d}\boldsymbol{x}'. \tag{3.26}$$

Mimicking the derivation for (3.20), we obtain from (3.26) that

$$\begin{aligned} \frac{\partial \hat{\eta}(\boldsymbol{\xi}, \boldsymbol{\theta}(t))}{\partial t} =& \nabla_{\boldsymbol{\xi}} \cdot \Big[\sum_{j=0}^{s} \frac{\alpha_j^{2(d+1))} \widehat{\mathcal{G}}_j(\boldsymbol{\xi})}{8\pi^2(s+1)} \Big(\nabla_{\boldsymbol{\xi}} \hat{\eta}(\boldsymbol{\xi}, \boldsymbol{\theta}(t)) - \nabla_{\boldsymbol{\xi}} \overline{\hat{\eta}(\boldsymbol{\xi}, \boldsymbol{\theta}(t))}\Big)\Big] \\ =& \mathrm{i} \nabla_{\boldsymbol{\xi}} \cdot \Big[\sum_{j=0}^{s} \frac{\alpha_j^{2(d+1))}}{4\pi^2(s+1)} \widehat{\mathcal{G}}_j(\boldsymbol{\xi}) \nabla_{\boldsymbol{\xi}} \hat{\eta}^{-}(\boldsymbol{\xi}, \boldsymbol{\theta}(t))\Big], \end{aligned} \tag{3.27}$$

where $\hat{\eta}^{-}(\boldsymbol{\xi}, \boldsymbol{\theta}(t)) := \mathfrak{Im}\Big\{\hat{\eta}(\boldsymbol{\xi}, \boldsymbol{\theta}(t))\Big\}$. The dynamic system (3.26) in the Fourier frequency domain implies that only the imaginary part of the error evolves during the gradient descent training if a two layer multi-scale neural network with activation function $\sigma(x) = \sin(x)$ and zero bias is used. This conclusion is consistent with the fact that the network function (3.23) can only be used to fit odd functions. Actually, the necessity of non-zero biases in a two layer neural network has been emphasized in [40, 23].

Note that $A_s^{\pm}(\boldsymbol{\xi}), B_s^{\mp}(\boldsymbol{\xi})$ defined in (3.21) are positive functions in $\mathbb{R}^d$. Therefore, the solution of (3.22) has an energy equality

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d} |\hat{\eta}^{\pm}(\boldsymbol{\xi},t)|^2 \mathrm{d}\boldsymbol{\xi} = -2\int_{\mathbb{R}^d}\left[A_s^{\mp}(\boldsymbol{\xi})\left|\nabla_{\boldsymbol{\xi}}\hat{\eta}^{\pm}(\boldsymbol{\xi},t)\right|^2 + B_s^{\pm}(\boldsymbol{\xi})|\hat{\eta}^{\pm}(\boldsymbol{\xi},t)|^2\right]\mathrm{d}\boldsymbol{\xi}, \tag{3.28}$$

which implies that the solution $\hat{\eta}^{\pm}(\boldsymbol{\xi},t) \to 0$ for any $\boldsymbol{\xi} \in \mathbb{R}^d$ as $t \to \infty$. That means the gradient descent learning for a fitting problem with one hidden layer neural network is convergent assuming that the learning rate is sufficiently small and the width of the neural network is sufficiently large. It is clear that the diffusion coefficients $\{A_s^{\pm}(\boldsymbol{\xi}), B_s^{\mp}(\boldsymbol{\xi})\}$ plays a key role in the error decay speed. Several plots of the coefficients $\{A_s^{\mp}(\xi), B_s^{\pm}(\xi)\}$ are given in Fig. 3.3 for different scales. We can see that both $A_s^{\mp}(\xi)$ and $B_s^{\pm}(\xi)$ have larger support and maximum values with an increasing scale s. This implies that larger s will leads to fast error reduction in a wider frequency region.

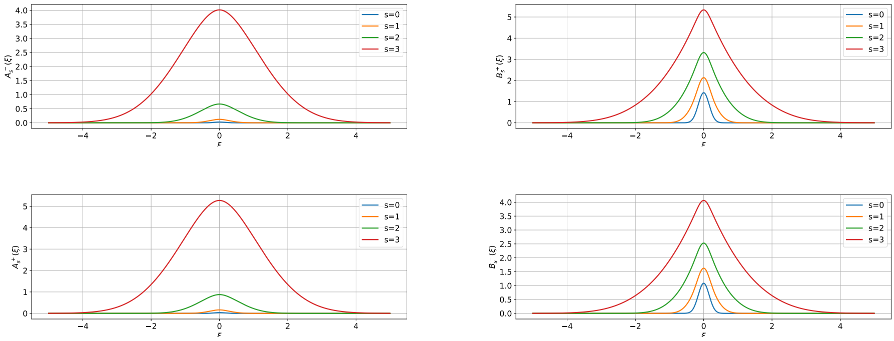


Fig. 3.3: Diffusion coefficients $A_s^{\mp}(\xi)$ (left) and $B_s^{\pm}(\xi)$ (right) with $\alpha_j = 2^j$, $s = 0, 1, 2, 3$.

4. Spectral bias analysis of a two layer MscaleDNN using the diffusion equation model. The analysis in previous section has shown that the error dynamics of the gradient descent learning can be approximately described by the diffusion equations (3.22) in the Fourier spectral domain when the network width and the domain size go to infinity and the learning rate to zero. In this section, we will first propose a Hermite spectral method to obtain highly accurate numerical solutions of the diffusion equation. Some numerical results will be presented to show that the error dynamics predicted by the diffusion model matches well with that of the MscaleDNN during realistic training. Moreover, the results also validate the capability of spectral bias reduction of the MscaleDNNs for wider range of frequencies. For simplicity, we only consider the 1-dimensional case to illustrate the main results.

4.1. Hermite spectral method for the diffusion equation problem. In order to examine quantitatively the decay of the error in the Fourier domain, we will solve numerically the equations in (3.22) with a Hermite spectral method for the ξ-variable of the equations in (3.22) on the unbounded computational domain. For this purpose, we introduce the Hermite functions (cf. [41]) defined by

$$\widehat{H}_n(\xi) = \frac{1}{\pi^{1/4}\sqrt{2^n n!}} e^{-\xi^2/2} H_n(\xi), \quad n \geq 0, \ \ \xi \in \mathbb{R}, \tag{4.1}$$

where $H_n(\xi)$ are Hermite polynomials. The Hermite functions $\widehat{H}_n(\xi)$ are orthogonal

$$(\widehat{H}_n(\xi), \widehat{H}_m(\xi)) = \int_{-\infty}^{+\infty} \widehat{H}_n(\xi)\widehat{H}_m(\xi) dx = \delta_{mn}, \tag{4.2}$$

where δ_{mn} is Kronecker symbol.

We discretize the computational time interval $[0, T]$ into equally-spaced intervals $I_k := [k\Delta t, (k+1)\Delta t]$ for $k = 0, 1, \cdots, N$, where $\Delta t = T/N$. Then, the Hermite spectral method together with backward Euler time discretization is to find approximation

$$\tilde{\eta}^{\pm}_m(\xi) = \sum_{k=0}^{p} \tilde{\eta}^{\pm}_{mk} \widehat{H}_k(\lambda\xi), \tag{4.3}$$

for $\hat{\eta}^{\pm}(\xi, t)$ at time $t_m = m\Delta t$ s.t.,

$$\Big(\frac{\tilde{\eta}^{\pm}_m(\xi) - \tilde{\eta}^{\pm}_{m-1}(\xi)}{\Delta t}, \widehat{H}_n(\lambda\xi) \Big) = -a(\tilde{\eta}^{\pm}_m(\xi), \widehat{H}_n(\lambda\xi)), \tag{4.4}$$

for all $n = 0, 1, \cdots, p$. Here, λ is a scaling parameter to achieve resolution near $\xi = 0$, and the bilinear form $a(\cdot, \cdot)$ is defined as

$$a(\phi(\xi), \psi(\xi)) = \Big(A^{\mp}_s(\xi) \frac{d\phi(\xi)}{d\xi}, \frac{d\psi(\xi)}{d\xi} \Big) - (B^{\pm}_s(\xi)\phi(\xi).\psi(\xi)). \tag{4.5}$$

Next, with the unknown vector denoted by $\boldsymbol{U}^{\pm}_m = (\tilde{\eta}^{\pm}_{m0}, \tilde{\eta}^{\pm}_{m1}, \cdots, \tilde{\eta}^{\pm}_{mp})^{\mathrm{T}}$, the numerical scheme (4.4) gives a linear system

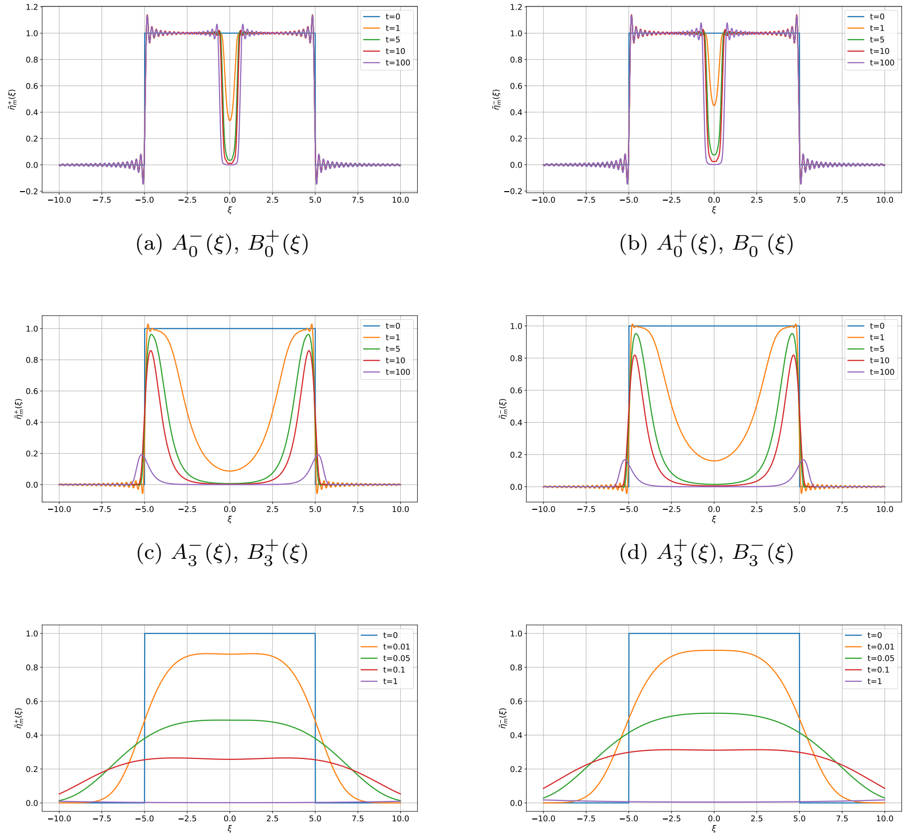
$$\mathbb{D} \frac{\boldsymbol{U}^{\pm}_m - \boldsymbol{U}^{\pm}_{m-1}}{\Delta t} = (\mathbb{K}^{\mp} + \mathbb{M}^{\pm}) \boldsymbol{U}^{\pm}_m, \tag{4.6}$$

where $\mathbb{D} = (D_{nk})$, $\mathbb{K}^{\pm} = (K^{\pm}_{nk})$, $\mathbb{M} = (M^{\pm}_{nk})$ are matrices with entries given by

$$\begin{aligned} & D_{nk} = (\widehat{H}_k(\lambda\xi), \widehat{H}_n(\lambda\xi)) = \frac{1}{\lambda}\delta_{nk}, \quad K^{\pm}_{nk} = -\lambda^2 \Big(A^{\pm}_s(\xi)\widehat{H}'_k(\lambda\xi), \widehat{H}'_n(\lambda\xi) \Big), \\ & M^{\pm}_{nk} = -(B^{\pm}_s(\xi)\widehat{H}_k(\lambda\xi), \widehat{H}_n(\lambda\xi)). \end{aligned} \tag{4.7}$$

By using the recurrence formula of the Hermite functions, formulations for the matrices $\mathbb{K}^{\pm}$, $\mathbb{M}^{\pm}$ can be derived analytically (see the appendix A).

4.2. Spectral bias reduction of a two layer MscaleDNN. Some numerical examples will be presented to show the capability of the diffusion model in predicting the error dynamics of a two layer MscaleDNN. The predicted results will be compared with the training error of the two-layer MscaleDNN with a large network width and sine activation function. The spectral bias reduction phenomena of MscaleDNNs is validated by the numerical solution of the diffusion model.



(a) $A_0^-(\xi)$, $B_0^+(\xi)$

(b) $A_0^+(\xi)$, $B_0^-(\xi)$

(c) $A_3^-(\xi)$, $B_3^+(\xi)$

(d) $A_3^+(\xi)$, $B_3^-(\xi)$

(e) $A_5^-(\xi)$, $B_5^+(\xi)$

(f) $A_5^+(\xi)$, $B_5^-(\xi)$

Fig. 4.1: Frequency domain error decay in time for a regression problem as predicted by the diffusion model (3.22) for three MscaleDNNs of scale ($s = 0, 3, 5$) with corresponding diffusion coefficients $A_s^{\pm}(\xi), B_s^{\pm}(\xi)$.

Test 1 (Decaying behavior predicted by the diffusion model). We first study the decay speed and range of the solution of the diffusion model (3.22). Considering an initial condition for the error function in the frequency domain

$$\hat{\eta}^{\pm}(\xi, 0) = \begin{cases} 1, & |\xi| \leq 5, \\ 0, & |\xi| > 5, \end{cases} \tag{4.8}$$

we will test the diffusion model (3.22) with three sets of coefficients $\{A_s^{\mp}(\xi), B_s^{\pm}(\xi)\}$, $s = 0, 3, 6$. For the numerical discretization of the PDE, we take $p = 100$, $\Delta t = 1.0e-3$ in (4.3). The numerical solutions at different time t are plotted in Fig. 4.1. The numerical results clearly show that the initial error function decays faster over wider frequency ranges with an increasing of s. It is worthy to emphasize that diffusion coefficients $\{A_0^{\mp}(\xi), B_0^{\pm}(\xi)\}$ only produce fast decay in only a small neighborhood of the zero frequency, which corresponds to exactly the spectral bias of a fully connected DNN [37, 51]. These observations are consistent with the performance of the MscaleDNN, which has faster convergence in the approximation of highly oscillated functions.

Test 2 (Validation of error diffusion model with real MscaleDNN training). In this test, we will show that the error dynamics of a finite but wide enough 2-layered multi-scale neural network can be predicted by the diffusion equation model quite well.

We consider a fitting problem with an objective function

$$f(x) = \sin a\pi x + \cos b\pi x, \tag{4.9}$$

on the interval $[-\beta, \beta]$. The Fourier transform of $f(x)$ with zero extension outside $[-\beta, \beta]$ is

$$\hat{f}(\xi) = \frac{\sin[(b+2\xi)\beta\pi]}{(b+2\xi)\pi} + \frac{\sin[(b-2\xi)\beta\pi]}{(b-2\xi)\pi} + \mathrm{i}\left[\frac{\sin[(a+2\xi)\beta\pi]}{(a+2\xi)\pi} - \frac{\sin[(a-2\xi)\beta\pi]}{(a-2\xi)\pi}\right].$$

For the two layers multi-scale neural network, the Fourier transform of $\mathcal{N}_s(x,\theta)$ with zero extension outside $[-\beta, \beta]$ can be calculated as

$$\widehat{\mathcal{N}}_s(\xi,\theta) = \frac{1}{\sqrt{N}}\sum_{j=0}^{s}\alpha_j\sum_{k=1}^{q} S_{j,k}(\xi,\theta) + \frac{1}{\sqrt{N}}\sum_{j=0}^{s}\alpha_j\sum_{k=1}^{q} C_{j,k}(\xi,\theta),$$

where

$$S_{j,k}(\xi,\theta) = \frac{-2\pi\mathrm{i}\xi(e^{2\pi\mathrm{i}\beta\xi}\sin(\alpha_j\theta_{jq+k}\beta - b_{jq+k}) + e^{-2\pi\mathrm{i}\beta\xi}\sin(\alpha_j\theta_{jq+k}\beta + b_{jq+k}))}{\alpha_j^2\theta_{jq+k}^2 - 4\pi^2\xi^2},$$

and

$$C_{j,k}(\xi,\theta) = \frac{\alpha_j\theta_{jq+k}(e^{2\pi\mathrm{i}\beta\xi}\cos(\alpha_j\theta_{jq+k}\beta - b_{jq+k}) - e^{-2\pi\mathrm{i}\beta\xi}\cos(\alpha_j\theta_{jq+k}\beta + b_{jq+k}))}{\alpha_j^2\theta_{jq+k}^2 - 4\pi^2\xi^2}.$$

We will show that the error $\hat{\eta}_{NN}(\xi,\theta) = \widehat{\mathcal{N}}_s(\xi,\theta) - \hat{f}(\xi)$ of the MscaleDNN by the gradient descent learning agrees with that predicted by the diffusion equation (3.22). We take $a = 4.2$, $b = 5.8$, $\beta = 1$ and the initial errors are given by $\eta_{NN}(x,\theta_0) = \mathcal{N}_s(x,\theta_0) - f(x)$ with parameters initialized by sampling from independent random variables of normal distribution. In the gradient descent training for the $\mathcal{N}_s(x,\theta)$, the training data set consists of 2000 uniformly distributed points in $[-\beta, \beta]$ and learning rate $\tau = 1.0e-3$ is adopted. In this example, a two layers neural network with $m = 12,000$, $\alpha_j = 2^j$ and scale $s = 3$ is tested and the training is performed in full batch.

Meanwhile, in the Fourier spectral domain, the diffusion equation (3.22) with initial function $\hat{\eta}(\xi,\theta_0) = \widehat{\mathcal{N}}_s(\xi,\theta_0) - \hat{f}(\xi)$ will be solved with a p-th order the Hermite spectral method introduced above. We take $p = 300$ and $\Delta t = \tau$ in the discretization.

The Fourier transform of $\eta_{NN}(x,\theta(t))$, denoted by

$$\hat{\eta}_{NN}(x,\theta(t)) = \hat{\eta}_{NN}^{+}(\xi,\theta(t)) + \mathrm{i}\eta_{NN}^{-}(\xi,\theta(t))$$

are compared with $\tilde{\eta}_m^{\pm}(\xi)$ at $t = m\Delta t$, see Fig. 4.2. Although many approximations have been used in deriving the diffusion model, the results show that the prediction produced by the diffusion model captured the main features of the error over a long time training process.

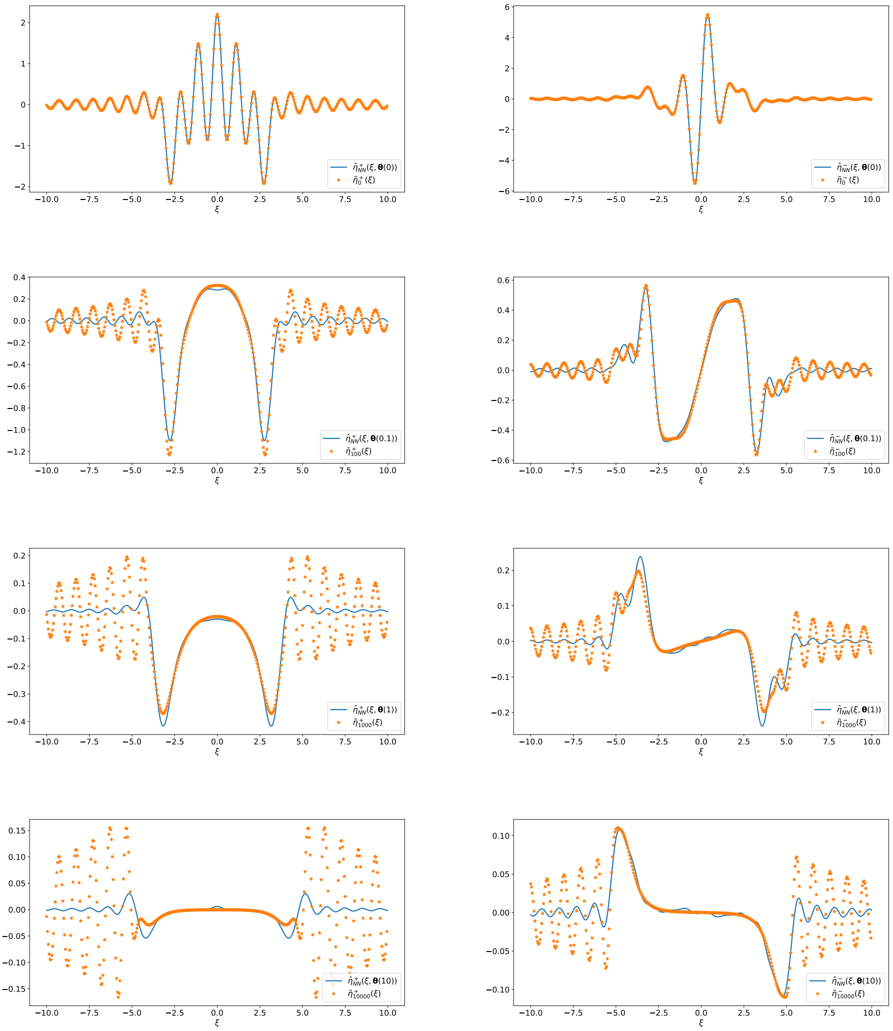


Fig. 4.2: Frequency domain error evolution (left - real part, right - imaginary part) in time of a 3-scale MscaleDNN with a network width $N = 12,000$ (line) vs prediction by diffusion model (3.22) (symbol).

On the other hand, we can also compare the training error with the diffusion model prediction in the physical domain. Using the fact that [13, 24]

$$\mathcal{F}^{-1}[\widehat{H}_k(\xi)](x) = \int_{-\infty}^{+\infty} \widehat{H}_k(\xi) e^{2\mathrm{i}\pi\xi x} \mathrm{d}\xi = \sqrt{2\pi}\mathrm{i}^k \widehat{H}_k(2\pi\xi), \tag{4.10}$$

the Hermite approximation of the error predicted by the diffusion model, i.e.,

$$\tilde{\eta}^{\pm}_m(\xi) = \sum_{k=0}^{p} \tilde{\eta}_{mk} \widehat{H}_k(\lambda\xi), \tag{4.11}$$

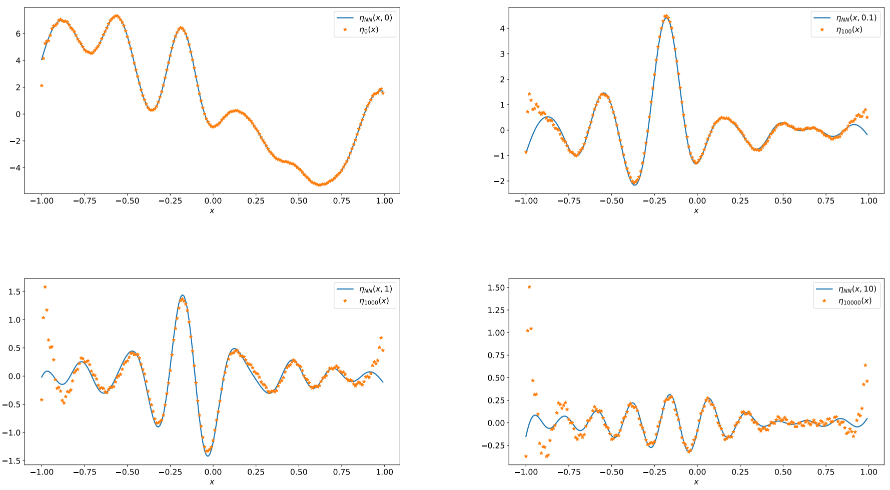


Fig. 4.3: Physical domain error evolution in time of a 3-scale MscaleDNN with a network width $N = 12,000$ (line) vs prediction by diffusion model (3.22) (symbol).

can be analytically transformed back to the physical domain as

(4.12)

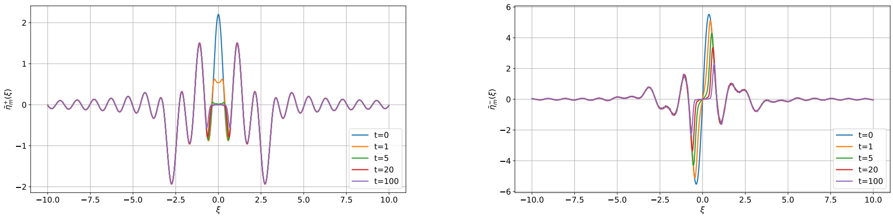
$$\eta_m^{\pm}(x) := \mathcal{F}^{-1}[\tilde{\eta}_m^{\pm}](x) = \sum_{k=0}^{p} \tilde{\eta}_{mk} \int_{-\infty}^{+\infty} \widehat{H}_k(\lambda\xi) e^{2\mathrm{i}\pi x\xi} \mathrm{d}\xi = \frac{\sqrt{2\pi}}{\lambda} \sum_{k=0}^{p} \tilde{\eta}_{mk} \mathrm{i}^k \widehat{H}_k\Big(\frac{2\pi x}{\lambda}\Big).$$

Then, in the physical domain the errors $\eta_{NN}(x, \theta(t_m))$ from the MscaleDNN training and $\eta_m(x) = \eta_m^{+}(x) + \mathrm{i}\eta_m^{-}(x)$ predicted by the diffusion equation can be compared in Fig. 4.3. Clearly, the evolution of the errors matches quite well in physical domain. It is worthy to point out that the fitting domain $\Omega = [-1, 1]$ is not large. However, the diffusion model can still be a satisfactory predictor for the real error through the training of the MScaleDNN with a large enough network width.

Test 3 (Reduction of spectral bias predicted by error diffusion model). With the confirmation of predicting capability of the diffusion equation model (3.22) for the error decay of the MscaleDNN with a large enough network width, we will use the model to demonstrate the spectral bias reduction of MscaleDNNs with increasing scales.

Again, We set the network width at $m = 12000$, and $a = 4.2$, $b = 5.8$ and the initial errors are given by $\hat{\eta}(\xi, \theta_0) = \widehat{\mathcal{N}}_s(\xi, \theta_0) - \hat{f}(\xi)$ with parameters initialized by sampling from independent random variables of normal distribution. In the Hermite spectral method approximation of the diffusion equation (3.22), we take $p = 300$ and $\Delta t = 1.0e - 3$. The numerical solution of the diffusion equations at different time t for a standard fully connected network (FCN) corresponding to coefficients $\{A_0^{\pm}(\xi), B_0^{\mp}\}$ and a 3-scales MscaleDNN corresponding to coefficients $\{A_3^{\pm}(\xi), B_3^{\mp}\}$ are plotted in Fig. 4.2-Fig. 4.3. We can see clearly that FCN with diffusion coefficients $\{A_0^{\pm}(\xi), B_0^{\mp}\}$ only produce decay in a very small neighborhood of the zero frequency while the 3-scale MscaleDNN with coefficients $\{A_3^{\pm}(\xi), B_3^{\mp}\}$ produce much faster decay in a larger frequency interval. Although the initial errors at $t = 0$ are different, $\widehat{\mathcal{N}}_0(\xi, \theta_0) - \hat{f}(\xi)$

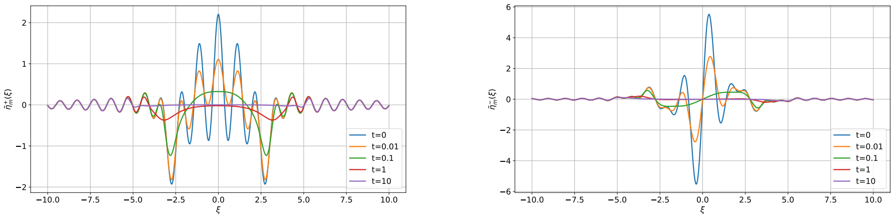
for FCN and $\widehat{\mathcal{N}}_3(\xi,\theta_0)-\hat{f}(\xi)$ for 3-scale MscaleDNN, the numerical results all verify that multi-scale neural networks has better performance in spectral bias reduction compared with the FCN.



(a) real part

(b) imaginary part

Fig. 4.4: Frequency domain error decay in time predicted by (3.22) for a FCN corresponding to coefficients $\{A_0^{\pm}(\xi), B_0^{\mp}\}$.



(a) real part

(b) imaginary part

Fig. 4.5: Frequency domain error decay in time predicted by (3.22) for a 3-scale MscaleDNN corresponding to coefficients $\{A_3^{\pm}(\xi), B_3^{\mp}\}$.

5. Conclusion and future work. In this paper, we investigated the convergence and spectral bias reduction properties of a two-layer multi-scale neural network for regression problems by deriving diffusion equation models in the frequency domain for predicting its error evolution. With the sine activation function, the gradient descent learning of MscaleDNNs leads to the diffusion equation models for the error assuming that the width of the neural network goes to infinity, the learning rate to zero and the fitting domain to the whole space. The diffusion coefficients of the diffusion equations are shown to have wider support in the frequency domain with more scales used in the MscaleDNNs, resulting in a reduction of spectral bias for the MscaleDNNs. This is consistent with the performance of the MscaleDNN with faster convergence in approximating highly oscillated functions from various applications. Moreover, the derived diffusion equation can predict the convergence of the MscaleDNNs learning algorithm even with a finite and reasonably wide network in a finite domain.

The analysis of the MScaleDNNs with more layers, and other popular activation functions, e.g., ReLU, Sigmoid, etc, will be studied following a similar approach of this paper.

Acknowledgments. The first and second authors acknowledges the support provided by NSFC (grant 12022104). W. Z. Zhang acknowledges the support provided by NSFC (grant 12201603). The work of W. Cai is supported by the US National Science Foundation grant DMS-2207449.

Appendix A. Analytic formula for the computation of matrices $\mathbb{K}^{\pm}, \mathbb{M}^{\pm}$. We first recall the recurrence formulas (cf. [41])

$$\begin{aligned} &\widehat{H}_0(x) = \pi^{-1/4}e^{-x^2/2}, \quad \widehat{H}_1(x) = \sqrt{2}\pi^{-1/4}xe^{-x^2/2}, \\ &\widehat{H}_{n+1}(x) = x\sqrt{\frac{2}{n+1}}\widehat{H}_n(x) - \sqrt{\frac{n}{n+1}}\widehat{H}_{n-1}(x) = 0, \quad n \geq 1, \end{aligned} \tag{A.1}$$

$$\begin{aligned} &\widehat{H}_0'(x) = -\frac{\pi^{-1/4}}{2}xe^{-x^2/2} = -\frac{\sqrt{2}}{2}\widehat{H}_1(x), \\ &\widehat{H}_n'(x) = \sqrt{2n}\widehat{H}_{n-1}(x) - x\widehat{H}_n(x) = \sqrt{\frac{n}{2}}\widehat{H}_{n-1}(x) - \sqrt{\frac{n+1}{2}}\widehat{H}_{n+1}(x), \quad n \geq 1, \end{aligned} \tag{A.2}$$

of the Hermite functions $\widehat{H}_n(x)$.

Then, by the recurrence formula (A.2), we have

$$\begin{aligned} &\widehat{H}_0'(x)\widehat{H}_0'(x) = \frac{1}{2}\widehat{H}_1(x)\widehat{H}_1(x), \\ &\widehat{H}_0'(x)\widehat{H}_n'(x) = -\frac{\sqrt{n}}{2}\widehat{H}_1(x)\widehat{H}_{n-1}(x) + \frac{\sqrt{n+1}}{2}\widehat{H}_1(x)\widehat{H}_{n+1}(x), \quad n \geq 1. \end{aligned} \tag{A.3}$$

and

$$\begin{aligned} &\widehat{H}_k'(x)\widehat{H}_n'(x) \\ =&\left[\sqrt{\frac{k}{2}}\widehat{H}_{k-1}(x) - \sqrt{\frac{k+1}{2}}\widehat{H}_{k+1}(x)\right]\left[\sqrt{\frac{n}{2}}\widehat{H}_{n-1}(x) - \sqrt{\frac{n+1}{2}}\widehat{H}_{n+1}(x)\right] \\ =&\frac{\sqrt{nk}}{2}\widehat{H}_{k-1}(x)\widehat{H}_{n-1}(x) - \frac{\sqrt{(n+1)k}}{2}\widehat{H}_{k-1}(x)\widehat{H}_{n+1}(x) \\ &-\frac{\sqrt{n(k+1)}}{2}\widehat{H}_{k+1}(x)\widehat{H}_{n-1}(x) + \frac{\sqrt{(n+1)(k+1)}}{2}\widehat{H}_{k+1}(x)\widehat{H}_{n+1}(x), \end{aligned} \tag{A.4}$$

for all $n, \quad k \geq 1$. Therefore,

$$K_{00}^{\pm} = \frac{1}{2}C_{11}^{\pm}, \quad K_{0n}^{\pm} = K_{n0}^{\pm} = -\frac{\sqrt{n}}{2}C_{1,n-1}^{\pm} + \frac{\sqrt{n+1}}{2}C_{1,n+1}^{\pm}, \quad n \geq 1, \tag{A.5}$$

where $C_{nk}^{\pm} = -\lambda^2\int_{-\infty}^{+\infty}A_s^{\pm}(\xi)\widehat{H}_k(\lambda\xi)\widehat{H}_n(\lambda\xi)d\xi$. Otherwise, for all $n, k \geq 1$,

$$\begin{aligned} K_{nk}^{\pm} =&\frac{\sqrt{n}}{2}\big(\sqrt{k}C_{n-1,k-1}^{\pm} - \sqrt{k+1}C_{n-1,k+1}^{\pm}\big) \\ &- \frac{\sqrt{n+1}}{2}\big(\sqrt{k}C_{n+1,k-1}^{\pm} - \sqrt{k+1}C_{n+1,k+1}^{\pm}\big). \end{aligned} \tag{A.6}$$

Noting that

$$M_{nk}^{\pm} = -\int_{-\infty}^{+\infty}B_s^{\pm}(\xi)\widehat{H}_k(\lambda\xi)\widehat{H}_n(\lambda\xi)d\xi, \tag{A.7}$$

and $A_s^{\pm}(\xi)$, $B_s^{\pm}(\xi)$ are linear combination of Gaussian functions as presented in (3.21), the computation of $C_{nk}^{\pm}$ and $M_{nk}^{\pm}$ can be reduced to compute the weighted inner products

$$
\begin{aligned}
I_{nk}(\tau) &= \int_{-\infty}^{+\infty} \widehat{H}_n(x)\widehat{H}_k(x)e^{-\tau x^2}dx \\
&= \frac{1}{\sqrt{\tau+1}}\int_{-\infty}^{+\infty} \widetilde{H}_n\Big(\frac{y}{\sqrt{\tau+1}}\Big)\widetilde{H}_k\Big(\frac{y}{\sqrt{\tau+1}}\Big)e^{-y^2}dy.
\end{aligned} \tag{A.8}
$$

where $\widetilde{H}_n(x)$ is the normalized Hermite polynomial defined by $\widetilde{H}_n(x) = e^{x^2/2}\widehat{H}_n(x)$. In fact, for $A_s^{\pm}(\xi)$, $B_s^{\pm}(\xi)$ given in (3.21), we have

$$
\begin{aligned}
C_{nk}^{\pm} &= -\frac{(1\pm e^{-2})\lambda}{2(2\pi)^{\frac{3}{2}}(s+1)}\sum_{j=0}^{s}\alpha_j^3 I_{nk}\Big(\frac{2\pi^2}{\alpha_j^2\lambda^2}\Big), \\
M_{nk}^{\pm} &= -\sqrt{\frac{\pi}{2}}\frac{1\pm e^{-2}}{(s+1)\lambda}\sum_{j=0}^{s}\alpha_j I_{nk}\Big(\frac{2\pi^2}{\alpha_j^2\lambda^2}\Big).
\end{aligned} \tag{A.9}
$$

Next, we present formulas for the calculation of the integrals $I_{nk}(\tau)$. Given any scaling factor λ, scaled Hermite polynomial $\widetilde{H}_n(\lambda y)$ can be represented by $\widetilde{H}_n(y)$ as follows

$$
\widetilde{H}_n(\lambda y) = \sum_{k=0}^{n} h_{n,k}(\lambda)\widetilde{H}_k(y), \tag{A.10}
$$

where $\{h_{n,k}(\lambda)\}$ can be calculated via recurrence formulas (A.15). Therefore,

$$
\begin{aligned}
I_{nk}(\tau) &= \frac{1}{\sqrt{\tau+1}}\int_{-\infty}^{\infty} \widetilde{H}_n\Big(\frac{y}{\sqrt{\tau+1}}\Big)\widetilde{H}_k\Big(\frac{y}{\sqrt{\tau+1}}\Big)e^{-y^2}dy \\
&= \frac{1}{\sqrt{\tau+1}}\sum_{i=0}^{n}\sum_{j=0}^{k} h_{n,i}\Big(\frac{1}{\sqrt{\tau+1}}\Big)h_{k,j}\Big(\frac{1}{\sqrt{\tau+1}}\Big)\int_{-\infty}^{\infty}\widetilde{H}_i(y)\widetilde{H}_j(y)e^{-y^2}dy \\
&= \frac{1}{\sqrt{\tau+1}}\sum_{i=0}^{\min\{n,k\}} h_{n,i}\Big(\frac{1}{\sqrt{\tau+1}}\Big)h_{k,i}\Big(\frac{1}{\sqrt{\tau+1}}\Big).
\end{aligned} \tag{A.11}
$$

Next, we derive recurrence formulas for the computation of the coefficients $\{h_{nk}(\lambda)\}$. We drop the explicit dependence on λ without confusion in the following derivation. By the definition of $\widetilde{H}_n(y)$ and the recurrence formula (A.1), we have

$$
\sqrt{2(n+1)}\widetilde{H}_{n+1}(\lambda y) = 2\lambda y\widetilde{H}_n(\lambda y) - \sqrt{2n}\widetilde{H}_{n-1}(\lambda y), \quad n \geq 1. \tag{A.12}
$$

Substituting the expansion (A.10) into (A.12) gives for $n \geq 1$

$$
\sqrt{2(n+1)}\sum_{k=0}^{n+1} h_{n+1,k}(\lambda)\widetilde{H}_k(y) = 2\lambda y\sum_{k=0}^{n} h_{n,k}(\lambda)\widetilde{H}_k(y) - \sqrt{2n}\sum_{k=0}^{n-1} h_{n-1,k}(\lambda)\widetilde{H}_k(y). \tag{A.13}
$$

Noting that

$$
\widetilde{H}_1(y) = \sqrt{2}y\widetilde{H}_0(y), \quad 2y\widetilde{H}_k(y) = \sqrt{2(k+1)}\widetilde{H}_{k+1}(y) + \sqrt{2k}\widetilde{H}_{k-1}(y)\ , \quad k \geq 1, \tag{A.14}
$$

direct calculation from (A.13) gives

$$\begin{aligned}
&2\lambda y\sum_{k=0}^{n} h_{n,k}(\lambda)\widetilde{H}_k(y)\\
=&\lambda\sum_{k=1}^{n} h_{n,k}(\lambda)\left[\sqrt{2(k+1)}\widetilde{H}_{k+1}(y)+\sqrt{2k}\widetilde{H}_{k-1}(y)\right]+2ayh_{n,0}(\lambda)\widetilde{H}_0(y)\\
=&a\sum_{k=0}^{n}\sqrt{2(k+1)}h_{n,k}(\lambda)\widetilde{H}_{k+1}(y)+a\sum_{k=1}^{n}\sqrt{2k}h_{n,k}(\lambda)\widetilde{H}_{k-1}(y)\\
=&a\sum_{k=1}^{n+1}\sqrt{2k}h_{n,k-1}(\lambda)\widetilde{H}_k(y)+a\sum_{k=0}^{n-1}\sqrt{2(k+1)}h_{n,k+1}(\lambda)\widetilde{H}_k(y).
\end{aligned}$$

Therefore, (A.13) can be rearranged into

$$\begin{aligned}
&\sqrt{2(n+1)}\sum_{k=0}^{n+1} h_{n+1,k}\widetilde{H}_k(y)\\
=&\lambda\sum_{k=1}^{n+1}\sqrt{2k}h_{n,k-1}(\lambda)\widetilde{H}_k(y)+\lambda\sum_{k=0}^{n-1}\sqrt{2(k+1)}h_{n,k+1}(\lambda)\widetilde{H}_k(y)\\
&-\sqrt{2n}\sum_{k=0}^{n-1}h_{n-1,k}(\lambda)\widetilde{H}_k(y)\\
=&[\sqrt{2}\lambda h_{n,1}(\lambda)-\sqrt{2n}h_{n-1,0}(\lambda)]\widetilde{H}_0(y)+\lambda\sqrt{2n}h_{n,n-1}(\lambda)\widetilde{H}_n(y)\\
&+\lambda\sqrt{2(n+1)}h_{n,n}(\lambda)\widetilde{H}_{n+1}(y)\\
&+\sum_{k=1}^{n-1}[\lambda\sqrt{2k}h_{n,k-1}(\lambda)+\lambda\sqrt{2(k+1)}h_{n,k+1}(\lambda)-\sqrt{2n}h_{n-1,k}(\lambda)]\widetilde{H}_k(y).
\end{aligned}$$

Matching the coefficients on both sides of the above equation gives us

$$\begin{aligned}
&h_{n+1,0}(\lambda)=\sqrt{\frac{1}{n+1}}\lambda h_{n,1}(\lambda)-\sqrt{\frac{n}{n+1}}h_{n-1,0}(\lambda),\\
&h_{n+1,k}(\lambda)=\lambda\sqrt{\frac{k+1}{n+1}}h_{n,k+1}(\lambda)-\sqrt{\frac{n}{n+1}}h_{n-1,k}(\lambda)+\lambda\sqrt{\frac{k}{n+1}}h_{n,k-1}(\lambda),\\
&\qquad\qquad \text{for}\quad 1\le k\le n-1,\\
&h_{n+1,k}(\lambda)=\lambda\sqrt{\frac{k}{n+1}}h_{n,k-1}(\lambda),\quad k=n,n+1,
\end{aligned}\tag{A.15}$$

for all $n\ge 1$, while the initial values are given by

$$h_{0,0}(\lambda)=1,\quad h_{1,0}(\lambda)=0,\quad h_{1,1}(\lambda)=\lambda.\tag{A.16}$$

By induction, $h_{n,k}(\lambda)$ has explicit formula for all $k=0,1,\cdots,n$

$$h_{n,k}(\lambda)=\begin{cases}0, & n-k=2s+1,\\ \sqrt{\dfrac{n!}{2^{n-k}k!}}\dfrac{1}{s!}\lambda^k(\lambda^2-1)^s, & n-k=2s.\end{cases}\tag{A.17}$$

REFERENCES

[1] A. Anandkumar, K. Azizzadenesheli, K. Bhattacharya, N. Kovachki, Z. Y. Li, B. Liu, and A. Stuart. Neural operator: Graph kernel network for partial differential equations. In *In: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[2] S. Arora, S. Du, W. Hu, Z. Y. Li, and R. Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[3] C. Beck, W. E, and A. Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *J. Nonlinear Sci.*, 29:1563–1619, 2019.

[4] W. Cai, DeepMartNet – A Martingale based deep neural network learning algorithm for eigenvalue/BVP problems and optimal stochastic controls, arXiv preprint arXiv:2307.11942v3, August, 2023.

[5] W. Cai, X. G. Li, and L. Z. Liu. A phase shift deep neural network for high frequency approximation and wave problems. *SIAM J. Sci. Comput.*, 42(5):A3285–A3312, 2020.

[6] W. Cai, Z.Q. Xu. Multi-scale deep neural networks for solving high dimensional PDEs. arXiv preprint arXiv:1910.11710. 2019 Oct 25.

[7] F. Chen, J. Huang, C. Wang, H. Yang. Friedrichs learning: Weak solutions of partial differential equations via deep learning. SIAM Jour. on Scientific Computing. 2023 Jun 30;45(3):A1271-99.

[8] Y. Cho and L. Saul. Kernel methods for deep learning. *NeurIPS*, 22:295–301, 2009.

[9] M. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Commun. Numer. Meth. En.*, 10(3):195–201, 1994.

[10] W. E, J. Han, A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.*, 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. *Commun. Math. Stat.*, 6(1):1–12, 2018.

[12] W. E, C. Ma, L. Wu Machine learning from a continuous viewpoint, i. *Sci. China Math.*, 63(11):2233–2266, 2020.

[13] I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products.* Academic press, 2014.

[14] D. Greenfeld, M. Galun, R. Basri, I. Yavneh, and R. Kimmel. Learning to optimize multigrid pde solvers. In *In: International Conference on Machine Learning*, pages 2415–2423. PMLR, 2019.

[15] J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proc. Nat. Acad. Sci.*, 115(34):8505–8510, 2018.

[16] J. Han and J. H. Long. Convergence of the deep bsde method for coupled fbsdes. *Probab. Uncertain. Quant. Risk*, 5:1–33, 2020.

[17] J. T. Hsieh, S. J. Zhao, S. Eismann, L. Mirabella, and S. Ermon. Learning neural PDE solvers with convergence guarantees. *In: International Conference on Learning Representations*, 2018.

[18] T. H. Hu, B. T. Jin, and Z. Zhou. Solving elliptic problems with singular sources using singularity splitting deep Ritz method. *arXiv preprint arXiv:2209.02931*, 2022.

[19] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Proc. Adv. Neural Inf. Process. Syst.*, 31, 2018.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *In: Advances in Neural Information Processing Systems 25 (NeurIPS 2012)*, pages 1097–1105, 2012.

[21] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Networks*, 9(5):987–1000, 1998.

[22] I. Lauriola, A. Lavelli, and F. Aiolli. An introduction to deep learning in natural language processing: Models, techniques, and tools. *Neurocomputing*, 470:443–456, 2022.

[23] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 32, 2019.

[24] H. Y. Li, R. Q. Liu, and L. L. Wang. Efficient hermite spectral-Galerkin methods for nonlocal diffusion equations in unbounded domains. *Numer. Math-Theory Me.*, 2022.

[25] X. A. Li, Z. Q. John Xu, and L. Zhang. A multi-scale dnn algorithm for nonlinear elliptic equations with multiple scales. *Commun. Comput. Phys.*, 28:1886–1906, 2020.

[26] Z. Y. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *In: International Conference on Learning Representations*, 2021.

[27] Z. Y. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar. Multipole graph neural operator for parametric partial differential equations. *In: Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

[28] Y. Liao and P. Ming. Deep Nitsche method: Deep Ritz method with essential boundary conditions. *arXiv preprint arXiv:1912.01309*, 2019.

[29] Y. J. Liu and C. Yang. Vpvnet: a velocity-pressure-vorticity neural network method for the stokes' equations under reduced regularity. *arXiv preprint arXiv:2112.07131*, 2021.

[30] Z. Q. Liu, W. Cai, and John Z. Q. Xu. Multi-scale deep neural network (MscaleDNN) for solving poisson-boltzmann equation in complex domains. *Commun. Comput. Phys.*, 28(5):1970–2001, 2020.

[31] D. H. Lu, K. Popuri, Gavin W. Ding, R. Balachandar, and M. F. Beg. Multimodal and multiscale deep neural networks for the early diagnosis of alzheimer's disease using structural mr and fdg-pet images. *Sci. Rep.*, 8(1):5697, 2018.

[32] T. Luo, Z. Ma, Z. Q. John Xu, and Y. Zhang. On the exact computation of linear frequency principle dynamics and its generalization. *SIAM J. Math. Data Sci.*, 4(4):1272–1292, 2022.

[33] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99–106, 2021.

[34] J. Müller and M. Zeinhofer. Deep ritz revisited. *arXiv preprint arXiv:1912.03937*, 2019.

[35] D. W. Otter, J. R Medina, and J. K. Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(2):604–624, 2020.

[36] Y. Peng, D. Hu, and Z. Q. John Xu. A non-gradient method for solving elliptic partial differential equations with deep neural networks. *J. Computat. Phys.*, 472:111690, 2023.

[37] 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. PMLR, 2019.

[38] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *J. Comput. Phys.*, 357:125–141, 2018.

[39] 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. *J. Computat. Phys.*, 378:686–707, 2019.

[40] B. Ronen, D. Jacobs, Y. Kasten, and S. Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *NeurIPS*, 32, 2019.

[41] J. Shen, T. Tang, and L.L. Wang. *Spectral Methods: Algorithms, Analysis and Applications*, volume 41 of *Springer Series in Computational Mathematics*. Springer-Verlag, 2011.

[42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *In: Proceedings of the 2015 Int. Conf. on Learning Representations (ICLR)*, 2015.

[43] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *In: Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[44] B. B. Traore, B. Kamsu-Foguem, and F. Tangara. Deep convolution neural network for image recognition. *Ecol. Inform.*, 48:257–268, 2018.

[45] R. Tsuchida, F. Roosta, and M. Gallagher. Invariance of weight distributions in rectified mlps. In *International Conference on Machine Learning*, pages 4995–5004. PMLR, 2018.

[46] K. Um, R. Brand, Y. R. Fei, P. Holl, and N. Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. *In: Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

[47] B. Wang, W. Z. Zhang, and W. Cai. Multi-scale deep neural network (MscaleDNN) methods for oscillatory stokes flows in complex domains. *Commun. Comput. Phys.*, 28(5):2139–2157, 2020.

[48] S. Wang, H. Wang, P. 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. 2021 Oct 1;384:113938.

[49] C. Williams. Computing with infinite networks. *NeurIPS*, 9:295–301, 1996.

[50] B. Xie, Y. Liang, and L. Song. Diverse neural network learns true target functions. In *Artificial Intelligence and Statistics*, pages 1216–1224. PMLR, 2017.

[51] Z. Q. J. Xu, Y. Y. Zhang, T. Luo, Y. Y. Xiao, and Z. Ma. Frequency principle: Fourier analysis

sheds light on deep neural networks. *Commun. Comput. Phys.*, 28(5):1746–1767, 2020.

[52] Z. Q. J. Xu, Y. Zhang, and T. Luo. Overview frequency principle/spectral bias in deep learning. *arXiv preprint arXiv:2201.07395*, 2022.

[53] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. M.*, 13(3):55–75, 2018.

[54] Y. H. Zang, G. Bao, X. J. Ye, and H. M. Zhou. Weak adversarial networks for high-dimensional partial differential equations. *J. Comput. Phys.*, 411:109409, 2020.

[55] L. Zhang, W. Cai, and Z. Q. John Xu. A correction and comments on "multi-scale deep neural network (mscalednn) for solving poisson-boltzmann equation in complex domains. cicp, 28(5):1970–2001,2020". *Commun. Comput. Phys.*, 33(5):1509–1513, 2023.

[56] W. Z. Zhang and W. Cai. Fbsde based neural network algorithms for high-dimensional quasilinear parabolic pdes. *J. Comput. Phys.*, 470:111557, 2022.

[57] E. D. Zhong, T. Bepler, B. Berger, and J. H. Davis. Cryodrgn: reconstruction of heterogeneous cryo-em structures using neural networks. *Nat. Methods*, 18(2):176–185, 2021.