SIAM J. SCI. COMPUT.
Vol. 42, No. 5, pp. A3285–A3312

A PHASE SHIFT DEEP NEURAL NETWORK FOR HIGH FREQUENCY APPROXIMATION AND WAVE PROBLEMS*

WEI CAI†, XIAOGUANG LI‡, AND LIZUO LIU†

Abstract. In this paper, we propose a phase shift deep neural network (PhaseDNN), which provides a uniform wideband convergence in approximating high frequency functions and solutions of wave equations. The PhaseDNN makes use of the fact that common deep neural networks (DNNs) often achieve convergence in the low frequency range first, and constructs a series of moderately sized DNNs trained for selected high frequency ranges. With the help of phase shifts in the frequency domain, each of the DNNs will be trained to approximate the function's specific high frequency range at the speed of learning for low frequency. As a result, the proposed PhaseDNN is able to convert high frequency learning to low frequency learning, allowing a uniform learning of wideband functions. The PhaseDNN is then applied to learn solutions of high frequency wave problems in inhomogeneous media through the least squares residual of either differential or integral equations. Numerical results have demonstrated the capability of the PhaseDNN as a meshless method in general domains in learning high frequency functions and oscillatory solutions of interior and exterior Helmholtz problems.

Key words. neural network, phase shift, high frequency waves, Helmholtz equations

AMS subject classifications. 11L03, 35Q60, 35Q68

DOI. 10.1137/19M1310050

1. Introduction.

Deep neural networks (DNNs) have shown greater potential in approximating high dimensional functions than traditional approximations based on Lagrangian interpolations or spectral methods. Recently, it has been found [20, 21, 16] that some common neural networks (NNs), including fully connected and convolution neural networks (CNNs) with tanh and ReLU activation functions, demonstrate a frequency-dependent convergence behavior. Namely, the DNNs during the training are able to approximate the low frequency components of the targeted functions first, before higher frequency components. This phenomenon is defined as the F-principle of DNNs [20]. The stalling of DNN convergence in the later stage of training could be mostly related to learning the high frequency components of the data. The F-principle behavior of DNNs is the opposite of that of the traditional multigrid method (MGM) [2] in approximating the solutions of PDEs where the convergence occurs first in the higher frequency end of the spectrum, as a result of smoothing operators employed in the MGM. The MGM takes advantage of this fast high frequency error reduction in the smoothing iteration cycles and restricts the original solution on a fine grid to a coarser grid, and then it continues the smoothing iteration on the coarse grid to reduce the higher frequency end of the spectrum in the context of the coarse grid. This downward restriction can be continued until errors over all frequencies are reduced by

*Submitted to the journal's Methods and Algorithms for Scientific Computing section January 2, 2020; accepted for publication (in revised form) July 7, 2020; published electronically October 20, 2020.

https://doi.org/10.1137/19M1310050

Funding: This work was supported by the U.S. Army Research Office (grant W911NF-17-1-0368). The work of X. G. Li is supported by the Construct Program of the Key Discipline in Hunan Province, China.

†Department of Mathematics, Southern Methodist University, Dallas, TX 75275 USA (cai@smu.edu, lizuol@mail.smu.edu).

‡Corresponding and joint first author. MOE-LCSM, School of Mathematics and Statistics, Hunan Normal University, Changsha, Hunan 410081, People's Republic of China (lixiaoguang@hunnu.edu.cn).

a small number of iterations on each level of the coarse grids.

There are many scientific computing problems which involve high frequency solutions in complex domains, such as high frequency wave equations in inhomogeneous media, arising from electromagnetic wave propagation in turbid media, rough surface scattering, seismic waves, and geophysical problems. Finding efficient solutions, especially in random environments, poses great computational challenges due to the highly oscillatory natures of the solutions. To compute the high frequency waves in inhomogeneous media, high order methods such as spectral methods for the differential equations or wideband fast multipole methods [7], [6], [19] for the integral equations are often used. These traditional numerical methods for solving PDEs are well understood and can provide very accurate results for various scientific and engineering applications. However, one of the well-known issues for these types of methods is the cost of mesh generation in order to discretize differential or integral operators. In this paper, we will develop meshless DNN-based numerical methods to handle high frequency functions and solutions of high frequency wave equations in inhomogeneous media in complex domains. These DNN-based methods may not produce as highly accurate numerical results as finite element methods, but they do have the advantage of easy implementation and avoiding the tremendous mesh generation cost while still giving satisfactory accuracy for many engineering applications. To improve the capability of usual DNNs for learning highly oscillatory functions in the physical spatial variables, we propose a phase shift DNN (PhaseDNN) with wideband learning capabilities in error reductions in the approximation for all frequencies of the targeted function by taking advantage of the faster convergence in the low frequencies of the DNN during its training. To learn a function of specific frequency range, we employ a phase shift in the k-space to translate its frequency to a range $|k| < K_0$; then the phase-shifted function with a low frequency content can be learned by common DNNs with a small number of training epochs. The resulting series of DNNs with phase shifts will make a phase shift deep neural network (PhaseDNN).

To achieve uniform wideband approximation of a general function, we can implement the PhaseDNN in a parallel manner where original data is decomposed into data of a specific frequency range, which, after a proper phase shift, is learned quickly. This approach can be implemented in a parallel manner; however, frequency extractions of the original training data have to be done using convolutions with a frequency selection kernel numerically, which could become very expensive or not accurate for scattered training data. Alternatively, we can implement the PhaseDNN in a nonparallel manner where original data consisting of information from all ranges of frequencies are learned together with phase shifts included in the makeup of the PhaseDNN, resulting in a coupled PhaseDNN. Although the coupled PhaseDNN lacks parallelism, it avoids the costly convolutions used in the parallel PhaseDNN to extract the frequency component from the original training data. This feature will be shown to be important when higher dimensional data are involved in the training. Thanks to this property, the coupled PhaseDNN can be used to solve high frequency wave problems where we seek solutions in a space of PhaseDNNs by minimizing the residuals of the differential or integral equation in a least squares approach. We should note that the idea of using frequency shifts to speed up the convergence of the DNN is similar to an approach in the wave-ray MGM [15] for high frequency problems where phase factors set at some frequency lattices together with smooth slowly variant amplitude functions are used in the wave-ray MGM framework. More recently, Fang et al. [11] introduced a ray-finite element method based on geometric optics where phase functions are adaptively learned to improve the effectiveness of the finite element method for high frequency

waves. Also, the proposed coupled PhaseDNN can be viewed as a Fourier-expansion–like neural network acting as a spectral method for general domains without a mesh. Neural networks which mimic other types of approximations such as wavelets [8] have been studied in [3], [5], [10].

The rest of the paper is organized as follows. In section 2, we will review the fast low frequency convergence for neural networks and present the parallel version phase shift deep neural network (PhaseDNN). Based on the properties of the PhaseDNN, a coupled PhaseDNN is introduced in section 3 to reduce the cost of learning in training the DNN for approximations. Then, the coupled PhaseDNN is used to find the solutions of wave problems in inhomogeneous media using either differential equation or integral equation formulations. Section 4 contains various numerical results of the PhaseDNN for approximations and solutions of wave problems. A conclusion and discussion for future work will be given in section 5. Appendix A will include an analysis of the equivalence between the parallel PhaseDNN and coupled PhaseDNN, providing the theoretical basis for the accurate results of the coupled PhaseDNN.

2. A parallel phase shift DNN (PhaseDNN) for high frequency approximation. A deep neural network (DNN) is a sequential alternative composition of linear functions and nonlinear activation functions. Given $m, n \geq 1$, let $\Theta(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^m$ be a linear function with the form $\Theta(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{W} = (w_{ij}) \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ are called weights and biases, respectively. The nonlinear activation function $\sigma(u) : \mathbb{R} \to \mathbb{R}$. By applying $\sigma(u)$ componentwisely, we can extend the activation function to $\sigma(u) : \mathbb{R}^n \to \mathbb{R}^n$. A DNN with $L+1$ layers can be expressed in a compact form as

$$\begin{aligned} T(\boldsymbol{x}) &= T^L(\boldsymbol{x}), \\ T^l(\boldsymbol{x}) &= [\Theta^l \circ \sigma](T^{l-1}(\boldsymbol{x})), \quad l = 1, 2, \dots L, \end{aligned} \tag{2.1}$$

with $T^0(\boldsymbol{x}) = \Theta^0(\boldsymbol{x})$, or equivalently,

$$T(\boldsymbol{x}) = \Theta^L \circ \sigma \circ \Theta^{L-1} \circ \sigma \cdots \circ \Theta^1 \circ \sigma \circ \Theta^0(\boldsymbol{x}). \tag{2.2}$$

Here, $\Theta^l(\boldsymbol{x}) = \boldsymbol{W}^l\boldsymbol{x} + \boldsymbol{b}^l : \mathbb{R}^{n^l} \to \mathbb{R}^{n^{l+1}}$ are linear functions. This DNN is also said to have L hidden layers, and its lth layer has n^l neurons.

In approximating a function $f(x)$ by a DNN through training, we minimize the least squares loss function

$$L(\boldsymbol{W}^0, \boldsymbol{b}^1, \boldsymbol{W}^1, \boldsymbol{b}^1, \dots, \boldsymbol{W}^L, \boldsymbol{b}^L) = \|f(\boldsymbol{x}) - T(\boldsymbol{x})\|_2^2 = \int_{-\infty}^{+\infty} |f(\boldsymbol{x}) - T(\boldsymbol{x})|^2 \, \mathrm{d}x. \tag{2.3}$$

For simplicity, we denote all the parameters in DNN by a parameter vector θ, i.e.,

$$\theta = (\boldsymbol{W}^0_{11}, \dots, \boldsymbol{W}^0_{n^0 n^1}, \boldsymbol{b}^0_1 \dots \boldsymbol{b}^0_{n^1}, \boldsymbol{W}^1_{11}, \dots, \boldsymbol{W}^1_{n^1 n^2}, \boldsymbol{b}^1_1 \dots \boldsymbol{b}^1_{n^2} \dots) \in \mathbb{R}^p.$$

Here, $p = (n^0 + 1) \times n^1 + (n^1 + 1) \times n^2 + (n^2 + 1) \times n^3 + \dots (n^L + 1)$ is the total number of the parameters. Numerically, with N training data $\{x_1, x_2, \dots, x_N\}$, the numerical loss function is defined as

$$L_N(\theta) = \sum_{i=1}^{N} |f(x_i) - T(x_i, \theta)|^2 . \tag{2.4}$$

We can study the loss function in the frequency space and first define the Fourier transform and its inverse of a function $f(\boldsymbol{x})$ by

$$\mathcal{F}[f](k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(x) e^{-ikx}\, \mathrm{d}x, \qquad \mathcal{F}^{-1}[\hat{f}](x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \hat{f}(k) e^{ikx}\, \mathrm{d}k. \tag{2.5}$$

Assuming the Fourier transforms of $f(x)$ and $T(x,\theta)$ exist, by Parseval's equality, we have

$$L(\theta) = \int_{-\infty}^{+\infty} |f(x) - T(x)|^2\, \mathrm{d}x = \int_{-\infty}^{+\infty} |\hat{f}(x) - \hat{T}(x)|^2\, \mathrm{d}k. \tag{2.6}$$

Let $L(k,\theta) = |\hat{f}(x) - \hat{T}(x)|^2$ denote the k-frequency component of $L(\theta)$. The following result was obtained in [21].

THEOREM 1. *Consider a DNN of one hidden layer with a* tanh *activation function. For any frequencies k_1 and k_2 such that $|\hat{f}(k_1)| > 0$, $|\hat{f}(k_2)| > 0$, and $|k_2| > |k_1| > 0$, there exist positive constants c and C such that for sufficiently small δ, we have*

$$\frac{\mu\left(\left\{\boldsymbol{W}^0 : \left|\frac{\partial L(k_1)}{\partial \theta_j}\right| > \left|\frac{\partial L(k_2)}{\partial \theta_j}\right| \text{ for all } j = 1, 2, \ldots, p\right\} \cap B_\delta\right)}{\mu(B_\delta)} > 1 - C\exp(-c/\delta), \tag{2.7}$$

where B_δ is a ball with radius δ centered at the origin of the $\boldsymbol{W}^0$-parameter space and $\mu(\cdot)$ is the Lebesgue measure.

Theorem 1 states that when the gradient decent method is applied to the loss function $L(\theta)$, for most of the $\boldsymbol{W}^0$-parameter space, the low frequency component of the loss function decays faster than the high frequency component. Although the result is only proved for a DNN with one hidden layer, this phenomenon has also been observed in higher dimensional experiments for DNNs with different depths and structures [21], [18]. Similar results have been proved for the ReLU network [18]. Therefore, as in [4], to speed up the learning of higher frequency contents of a target function $f(x)$, we can employ a phase shift technique to translate the higher frequency spectrum $\hat{f}(k)$ to a frequency range of $[-K_0, K_0]$ with some small frequency K_0. Such a shift in frequency is a simple phase factor multiplication on the training data in the physical space.

2.1. Frequency selection kernel $\boldsymbol{\phi_j^\vee(x)}$. For a given frequency increment Δk, say, $\Delta k = 2K_0$, let us assume that for some integer $M > 0$,

$$\operatorname{supp} \hat{f}(k) \subset [-M\Delta k, M\Delta k].$$

We first construct a mesh for the interval $[-M\Delta k, M\Delta k]$ by

$$\omega_j = j\Delta k, j = -M, \ldots, M. \tag{2.8}$$

Then, we introduce a partition of unit (POU) $\{\phi_j(k)\}_{j=-M}^{M}$ for the interval $[-M\Delta k, M\Delta k]$ associated with the mesh as

$$1 = \sum_{j=-M}^{M} \phi_j(k), \ k \in [-M\Delta k, M\Delta k]. \tag{2.9}$$

The simplest choice of $\phi_j(k)$ is $\phi_j(k) = \phi(\frac{k-\omega_j}{\Delta k})$, and $\phi(k)$ is just the characteristic function of $[-\frac{1}{2}, \frac{1}{2}]$, i.e., $\phi(k) = \chi_{[-\frac{1}{2},\frac{1}{2}]}(k)$. The inverse Fourier transform $\mathcal{F}^{-1}$ of $\phi(k)$, indicated by $\vee$, is $\phi^{\vee}(x) = \frac{1}{\sqrt{2\pi}} \frac{\sin \frac{x}{2}}{\frac{x}{2}}$.

With the POU in (2.9), we can decompose the target function $f(x)$ in the Fourier space as follows:

$$\hat{f}(k) = \sum_{j=-M}^{M} \phi_j(k)\hat{f}(k) \triangleq \sum_{j=-M}^{M} \hat{f}_j(k), \tag{2.10}$$

which will give a corresponding decomposition in x-space as

$$f(x) = \sum_{j=-M}^{M} f_j(x), \tag{2.11}$$

where

$$f_j(x) = \mathcal{F}^{-1}[\hat{f}_j](x).$$

The decomposition (2.11) involves $2M+1$ functions $f_j(x)$, whose frequency spectrum is limited to $[\omega_j - \frac{\Delta k}{2}, \omega_j + \frac{\Delta k}{2}]$. Therefore, a simple phase shift could translate its spectrum to $[-\Delta k/2, \Delta k/2]$, and it could be learned quickly by a relatively small DNN $T_j(x)$ with a few training epochs.

Specifically, as the support of $\hat{f}_j(k)$ is $[\omega_j - \frac{\Delta k}{2}, \omega_j + \frac{\Delta k}{2}]$, then $\hat{f}_j(k+\omega_j)$ is supported in $[-\Delta k/2, \Delta k/2]$, and its inverse Fourier transform $\mathcal{F}^{-1}[\hat{f}_j(k+\omega_j)]$, denoted as

$$f_j^{\text{shift}}(x) = \mathcal{F}^{-1}\left[\hat{f}_j(k-\omega_j)\right](x), \tag{2.12}$$

can be learned quickly by a DNN $T_j(x,\theta)$ by minimizing a loss function

$$L_j(\theta) = \int_{-\infty}^{\infty} |f_j^{\text{shift}}(x) - T_j(x,\theta)|^2 \, \mathrm{d}x \tag{2.13}$$

in an n_0-epoch of training.

Moreover, we know that

$$f_j^{\text{shift}}(x_i) = e^{-i\omega_j x_i} f_j(x_i), \quad 1 \le i \le N, \tag{2.14}$$

which provides the training data for $f_j^{\text{shift}}(x)$. Equation (2.14) shows that once $f_j^{\text{shift}}(x)$ is learned, $f_j(x)$ is also learned by removing the phase factor:

$$f_j(x) \approx e^{i\omega_j x} T_j(x, \theta^{(n_0)}). \tag{2.15}$$

Now with all $f_j(x)$ for $-M \le j \le M$ learned after n_0 steps of training each, we have an approximation to $f(x)$ over all frequency ranges $[-M\Delta k, M\Delta k]$ as follows:

$$f(x) \approx \sum_{j=-M}^{M} e^{i\omega_j x} T_j(x, \theta^{(n_0)}), \tag{2.16}$$

where $\theta^{(n_0)}$ is the value of parameters after n_0 steps of training.

2.2. Training data for parallel phase shift DNN (PhaseDNN) algorithm. In practice, we only know the value of $f(x)$ at some locations, which will be used to train the PhaseDNN; namely, our goal is to learn the function $f(x)$ using a training data set

$$\{x_i, f_i = f(x_i)\}_{i=1}^N. \tag{2.17}$$

In order to apply the decomposition (2.11) to $f(x)$, when carrying out the subtraining problem (2.13), we need to compute the training data for $f_j^{\text{shift}}(x)$ based on original training data (2.17). This procedure can be done in x-space through the following convolution:

$$\begin{aligned} f_j^{\text{shift}}(x_i) &= e^{-i\omega_j x_i} f_j(x_i) = e^{-i\omega_j x_i} \phi_j^\vee * f(x_i) = \int_{-\infty}^{\infty} \phi_j^\vee(x_i - s) f(s) ds \\ &\approx \frac{2\delta}{N_s} \sum_{x_s \in (x_i - \delta, x_i + \delta)} e^{-i\omega_j x_i} \phi_j^\vee(x_i - x_s) f(x_s), \end{aligned} \tag{2.18}$$

where δ is chosen such that the kernel function $|\phi^\vee(k)|$ is small enough outside $(-\delta, \delta)$.

Note that the generation of training data for $f_j^{\text{shift}}(x_i)$ and the subsequent training of each DNN $T_j(x, \theta)$ to approximate $f_j^{\text{shift}}(x)$ can be done in parallel. For this reason, this approach will be termed a parallel PhaseDNN, which will consist of the following steps: (1) select the phase frequency ω_j; (2) for each j, construct the training data $f_j^{\text{shift}}(x_i)$; (3) train all DNN $T_j(x, \theta)$; and (4) combine all individual DNNs $T_j(x, \theta)$ with a shift backward to get an approximation for the function $f(x)$.

3. A coupled PhaseDNN.

3.1. Approximating functions. In the previous section, we use the frequency selection kernel $\phi_j^\vee(x)$ to decompose the training data into different frequency components; each of them, after being phase-shifted, can be represented by a small DNN. This method can be implemented in parallel. This strategy is required to use convolution in (2.18) to construct the training data for each small DNN. In principle, the computation cost of this part can be reduced to $O(N \log(N))$ by using the FFT; here N is the number of samples, provided that the distribution of the data is close to uniform and covers the whole domain where the approximation is sought. However, for randomly distributed and scattered samples with many regions missing data, the FFT technique will not be applicable or efficient, and computing the convolution directly via a matrix multiplication thus requires a storage and computation of $O(N^2)$. As a result, this convolution process strongly restricts the performance of PhaseDNN for higher dimensions and larger data sets.

To avoid this problem, based on the construction of the parallel PhaseDNN (2.16), we would like to consider a coupled weighted phase-shifted DNN as an ansatz for a coupled PhaseDNN,

$$T(x) = \sum_{m=1}^{M} e^{i\omega_m x} T_m(x), \tag{3.1}$$

to approximate $f(x), x \in \mathbb{R}^d$, where $T_m(x)$ are relatively small complex-valued DNNs, i.e., $T_m(x) = T_m^{(real)}(x) + iT_m^{(imag)}(x)$. $T_m^{(real)}(x)$ and $T_m^{(imag)}(x)$ are two independent DNNs. $\{\omega_m\}_{m=1}^M$ are frequencies of interest in the target function.

We will minimize the following least squares loss function:

$$L(\theta) = \int_{-\infty}^{+\infty} |f(x) - T(x)|^2 \, \mathrm{d}x, \tag{3.2}$$

or, numerically,

$$L_N(\theta) = \sum_{i=1}^{N} |f(x_i) - T(x_i)|^2 = \sum_{i=1}^{N} \left| f(x_i) - \sum_{m=1}^{M} e^{i\omega_m x_i} T_m(x_i) \right|^2. \tag{3.3}$$

Remark 1. This method is similar to an expansion with Fourier modes of selected frequency with variable coefficients defined by DNNs. When $f(x)$ is a real function, it is equivalent to use real "Fourier" series rather than complex "Fourier" series. Namely, we will consider the sine and cosine expansions

$$T(x) = \sum_{m=1}^{M} A_m \cos(\omega_m x) + B_m \sin(\omega_m x) \tag{3.4}$$

to approximate $f(x)$, where A_m, B_m are DNNs while $\omega = 0$ will always be included.

It can be shown that under the condition that the weight of the input layer for each T_m is small, the coupled PhaseDNN is equivalent to the parallel PhaseDNN, and an analysis of this fact is given in Appendix A. In practical applications, the condition that the weight of the input layer for each T_m is small holds at the beginning of training, since we always use small random values to initialize the network. As a matter of fact, to encourage this condition in the training process, we can add a weight regularization in the loss function, namely,

$$L_N^R(\theta) = \sum_{i=1}^{N} |f(x_i) - T(x_i)|^2 + \beta \sum_{m,l} \left\| \boldsymbol{W}^{m,l} \right\|_F^2, \tag{3.5}$$

where x_i are training data, $\boldsymbol{W}^{m,l}$ is the weight matrix of the lth layer of sub DNN T_m, and β is a regularization parameter. This weight regularization can also restrain some training disasters such as the gradient blowing up [12].

Compared with the previous approach using the phase selecting kernel, the main advantage of the coupled PhaseDNN is that there is no need for computing convolutions, without the additional quadrature errors, to generate training data for the training of a selected frequency range. This allows us to deal with a large data set and higher dimensional problems. However, the coupled PhaseDNN cannot be parallelized, and we must choose the frequencies ω_m before training and then build DNN $T(x)$ using these ω_m frequencies. If coupled PhaseDNN does not contain enough frequencies, we can modify the coupled phaseDNN with additional frequencies to improve the result.

3.2. Solving differential equations through least squares residual minimization. The coupled PhaseDNN (3.1) will be taken as an ansatz for finding the solution of differential equations by minimizing the least squares of the differential equation's residual, similar to the least squares finite element method [13], [1] and the physics-informed neural network [17]. DNNs have also been used to solve differential equations by a variational method [9].

The coupled PhaseDNN will approximate the solution of the following high frequency Helmholtz equation:

$$\mathcal{L}[u] \triangleq u'' + (\lambda^2 + c\omega(x))u = f(x), \tag{3.6}$$

where $\lambda > 0$, $c\omega(x)$ can be viewed as a perturbation modeling the inhomogeneity of the otherwise homogeneous media.

The PhaseDNN solution, in the form of (3.1) or (3.4), for (3.6) with different boundary conditions can be sought by minimizing the following loss function:

$$L_N(\theta) = L_{ode}(\theta) + \rho L_{bc}(\theta), \tag{3.7}$$

where

$$L_{ode}(\theta) = \sum_{i=1}^{N} \left| \mathcal{L}[T](\cdot, \theta)(x_i) - f(x_i) \right|^2, \tag{3.8}$$

$\{x_i\}_{i=1}^N \in [-1, 1]$ are preselected locations to evaluate the residual of the DE by the DNN, and L_{bc} is the boundary condition regularization term with ρ as the regularization parameter.

We consider two typical kinds of boundary value problems. One is the Dirichlet boundary condition for an *interior Helmholtz problem*,

$$\begin{cases} u'' + (\lambda^2 + c\omega(x))u = f(x), \\ u(a) = u_1, u(b) = u_2, \end{cases} \tag{3.9}$$

where the media perturbation $c\omega(x)$ will be specified in the numerical tests. For this case, the L_{bc} term is chosen naturally as

$$L_{bc} = (T(a, \theta) - u_1)^2 + (T(b, \theta) - u_2)^2. \tag{3.10}$$

Remark 2. For 1D Dirichlet boundary value problems, we can also deal with the boundary condition using a slight modification of the coupled PhaseDNN by replacing the coupled PhaseDNN $T(x, \theta)$ in (3.8) with

$$\tilde{T}(x, \theta) = T(x, \theta) + P(\theta)x + Q(\theta), \tag{3.11}$$

where $P(\theta) = (u_2 - u_1 + T(a, \theta) - T(b, \theta))/(b - a)$, $Q(\theta) = (u_1(b - T(a, \theta)) - u_2(a - T(b, \theta)))/(b - a)$. With this modification, $\tilde{T}(x, \theta)$ will be a coupled PhaseDNN which satisfies the Dirichlet boundary condition naturally, and we can then use $L_N(\theta) = L_{ode}(\theta)$ as the loss function.

In the following 1D numerical tests, we use both (3.7) and (3.11) to deal with the Dirichlet boundary conditions, and they perform similarly, though the latter approach will be difficult to apply to problems in complex 3D domains.

The second type is an outgoing radiation condition for an *exterior Helmholtz problem* for the wave scattering of a finite inhomogeneity described by a compact supported function $\omega(x)$,

$$\begin{cases} u'' + (\lambda^2 + c\omega(x))u = f(x), \\ u' \pm \lambda u \to 0, (x \to \mp\infty). \end{cases} \tag{3.12}$$

For the exterior problem, we assume both the perturbation $\omega(x)$ and resource function $f(x)$ are compactly supported in $[-1,1]$, and we are only interested in the solution in $[-1,1]$. To solve the differential equation on the unbounded domain, we need to truncate the domain to a finite one with an absorbing boundary condition, which in this case is the same as the radiation condition. So, we will consider the following Robin problem of the Helmholtz equation:

$$\begin{cases} u'' + (\lambda^2 + c\omega(x))u = f(x), \\ u'(-a) + \lambda u(-a) = 0, \quad u'(a) - \lambda u(a) = 0, \end{cases} \tag{3.13}$$

where a constant $a \geq 2$ is chosen. It can be shown that with $\omega(x)$ and $f(x)$ supported in $[-1,1]$, boundary value problems (3.12) and (3.13) have the same solution in $[-1,1]$. The L_{bc} is chosen as

$$L_{bc} = |T'(-a,\theta) + i\lambda T(-a,\theta)|^2 + |T'(-a,\theta) - i\lambda T(-a,\theta)|^2.$$

Note that the solution is complex-valued, $T(x,\theta)$ here should use the form (3.1), and each T_m in (3.1) should also be complex-valued.

3.3. Solving integral equations for exterior Helmholtz problems. For exterior scattering problem, a more convenient approach is converting (3.12) into an integral equation via a Green's function.

When $c = 0$, the Green's function of problem (3.12) is simply

$$G(x,x') = \frac{1}{2i\lambda} e^{i\lambda|x-x'|}. \tag{3.14}$$

We can write the solution to (3.12) with $c > 0$ in terms of $G(x,x')$ by an integral equation:

$$\begin{aligned} u(x) &= \int_{-\infty}^{\infty} f(x')G(x,x')\,\mathrm{d}x' - \int_{-\infty}^{\infty} c\omega(x')u(x')G(x,x')\,\mathrm{d}x' \\ &= \int_{-1}^{1} f(x')G(x,x')\,\mathrm{d}x' - \int_{-1}^{1} c\omega(x')u(x')G(x,x')\,\mathrm{d}x' \\ &\triangleq f_G(x) - \mathcal{K}[u]. \end{aligned} \tag{3.15}$$

The second equality holds because $f(x)$ and $\omega(x)$ are supported in $[-1,1]$. The term $f_G(x)$ can be calculated by a Gaussian quadrature before training.

In order to apply PhaseDNN to approximate the solution of the integral equation (3.15), we will first discretize the integral operator in a finite dimensional space by considering a finite element mesh $\{\xi_j\}_{j=1}^M$ for the interval $[-1,1]$ and a finite element nodal basis $\{\phi_j(x)\}_{j=1}^M$ with the Kronecker property, i,e.,

$$\phi_j(\xi_k) = \delta_{jk}. \tag{3.16}$$

For a function $u(x)$ expressed in terms of the basis function $\phi_j(x)$,

$$u(x) = \sum_{j=1}^{M} u_j \phi_j(x), \qquad u_j = u(\xi_j), \tag{3.17}$$

the application of integral operator $\mathcal{K}[u]$ gives

$$\mathcal{K}[u](x) = \sum_{j=1}^{M} u_j \int_{-1}^{1} G(x,\xi)\omega(\xi)\phi_j(\xi)d\xi \triangleq \sum_{j=1}^{M} u_j \psi_j(x), \tag{3.18}$$

where

$$\psi_j(x) = \int_{-1}^{1} G(x,\xi)\omega(\xi)\phi_j(\xi)d\xi. \tag{3.19}$$

Substituting (3.17) and (3.19) into (3.15), we have

$$\sum_{j=1}^{M} u_j\phi_j(x) = f_G(x) - c\sum_{j=1}^{M} u_j\psi_j(x). \tag{3.20}$$

We will find a DNN $T(x,\theta)$ approximation for solution $u(x)$ by minimizing the loss function of the residual of (3.20) at N-locations $\{x_i\}_{j=1}^{N}$ with u_j replaced by $T(\xi_j,\theta)$,

$$L_N(\theta) = \|\boldsymbol{A}\boldsymbol{T}(\theta) + c\boldsymbol{B}\boldsymbol{T}(\theta) - \boldsymbol{f_G}\|^2, \tag{3.21}$$

where $\boldsymbol{T}(\theta) = [T(\xi_1,\theta), T(\xi_2,\theta), \ldots, T(\xi_M,\theta)] \in R^M$, $\boldsymbol{f_G} = [(f_G)(x_1), \ldots (f_G)(x_N)] \in R^N$, and $\boldsymbol{A}_{ij} = \phi_j(x_i)$, $\boldsymbol{B}_{ij} = \psi_j(x_i)$, $1 \le i \le N, 1 \le j \le M$. The matrix $\boldsymbol{B}$ can also be calculated by a Gaussian quadrature before training.

The integral equation method also applies to other types of homogeneous boundary conditions. Provided that one can write down the Green's function for (3.6) with the corresponding boundary condition, the corresponding matrix $\boldsymbol{B}$ can be computed by Gaussian quadrature similarly.

Remark 3. The residual of the integral equation formulation can also be viewed as a preconditioned version of that of the differential equation. If we write $\mathcal{L} = \mathcal{L}_1 + c\mathcal{L}_2$, where $\mathcal{L}_1[u] = u'' + \lambda^2 u$, $\mathcal{L}_2[u] = \omega(x)u(x)$, the operator $G * (\cdot)$ can be regarded as the inverse operator of $\mathcal{L}_1$. Thus (3.15) is just $(I + c\mathcal{L}_1^{-1}\mathcal{L}_2)u = \mathcal{L}_1^{-1}f$. When c is small, this preconditioned residual is expected to give a better performance than the PhaseDNN with least squares residual of the differential equation, and our numerical results later will confirm this.

In the next section, we will apply both the parallel PhaseDNN and the coupled PhaseDNN methods to approximation problems and solve differential equations with the coupled PhaseDNN only. All problems in this paper are done on a NVIDIA Tesla V100 GPU with TensorFlow and PyTorch. We use TensorFlow 1.13 to solve differential equations and PyTorch to solve the integral equations. All the training processes are carried out by the Adam algorithm [14] with the default parameters of Adam except the learning rate, which will be clarified for specific examples.

4. Numerical results.

4.1. Approximation of functions with PhaseDNN.

4.1.1. Parallel PhaseDNN. In this section, we will present numerical results to demonstrate the capability of PhaseDNN to learn high frequency content of target functions. In practice, we could sweep over all frequency ranges with a prescribed frequent increment $\Delta k = 5$. For the test function for which we have some rough idea about the range of frequencies in the data, only a few frequency intervals are selected for the phase shift.

We choose a target function $f(x)$ in $[-\pi,\pi]$:

$$f(x) = \begin{cases} 10(\sin x + \sin 3x) & \text{if } x \in [-\pi, 0], \\ 10(\sin 23x + \sin 137x + \sin 203x) & \text{if } x \in [0,\pi]. \end{cases} \tag{4.1}$$

Because the frequencies of this function are well separated, we need not sweep all the frequencies in $[-\infty, +\infty]$. Instead, we choose $\Delta k = 5$ and use the functions

$$\begin{aligned} \phi_1(k) &= \chi_{[-205,-200]}(k), & \phi_2(k) &= \chi_{[-140,-135]}(k), \\ \phi_3(k) &= \chi_{[-25,-20]}(k), & \phi_4(k) &= \chi_{[-5,0]}(k), \\ \phi_5(k) &= \chi_{[0,5]}(k), & \phi_6(k) &= \chi_{[20,25]}(k), \\ \phi_7(k) &= \chi_{[135,140]}(k), & \phi_8(k) &= \chi_{[200,205]}(k) \end{aligned}$$

to collect the frequency information in the corresponding frequency intervals and shift the center of the interval to the origin by a phase factor. For each $f_j(x) = \mathcal{F}^{-1}[\hat{f}\phi_j](x)$, we construct two DNNs to learn its real part and imaginary part separately. Every DNN has 4 hidden layers, and each layer has 40 neurons. Namely, the DNN has a structure 1-40-40-40-40-1. The training data is obtained by 10,000 samples from the uniform distribution on $[-\pi, \pi]$, and the testing data is 10000 evenly spaced points in $[-\pi, \pi]$. We train these DNNs with 1000 epochs by the Adam optimizer with training rate 0.002. The batchsize is 2000 for each DNN. The result is shown in Figure 4.1 while the details of the training result are shown in Figure 4.2. These figures clearly show that PhaseDNN can capture the various high frequencies, from low frequency ± 1, ± 3 to high frequency ± 203, quite well. The training times of PhaseDNN are collected in Table 4.1

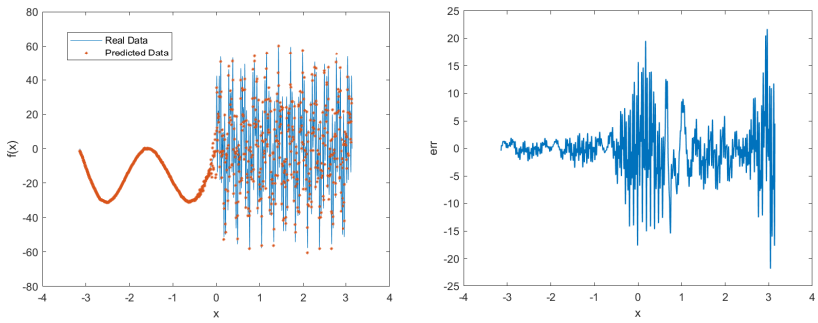

FIG. 4.1. *The fitting result for $f(x)$ using the parallel PhaseDNN. Left panel: The blue solid line is $f(x)$, and the data marked by red dots are the predicted value by PhaseDNN at testing data set. Right panel: The error plot. (Color available online.)*

TABLE 4.1

The training time statistics. For each j, the training time is the sum of training time of real and imaginary parts. Each DNN is trained by $1,000$ *epochs with batchsize* $2,000$.

Frequency interval	Convolution time(s)	Training time(s)	Total time (s)
$[-205, -200]$	11.32	15.05	26.38
$[-140, -135]$	11.32	15.18	26.51
$[-25, -20]$	11.46	14.99	26.45
$[-5, 0]$	10.70	14.97	25.67
$[0, 5]$	11.22	14.98	26.21
$[20, 25]$	11.26	15.00	26.27
$[135, 140]$	11.32	15.13	26.45
$[200, 205]$	11.32	15.03	26.36
Total	89.94	120.37	210.32

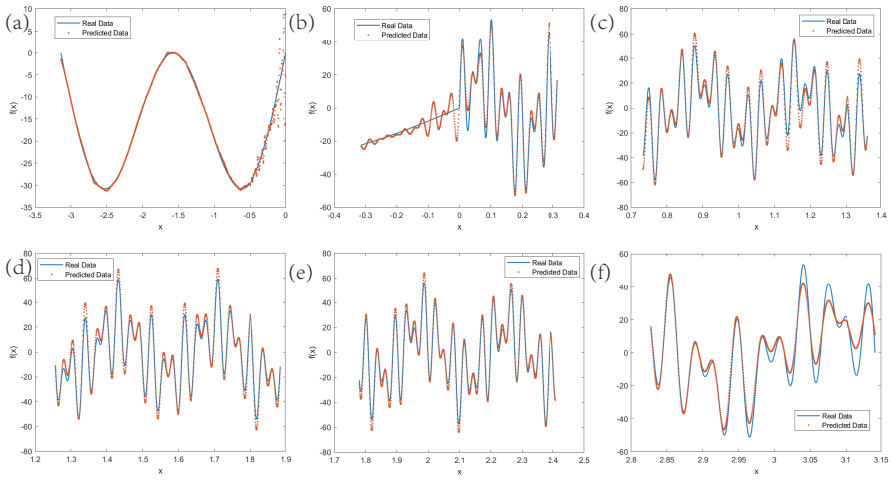

FIG. 4.2. *The detailed results of training in different intervals. The subfigures* (a)–(f) *show the results in intervals* $[-\pi, 0]$, $[-\pi/10, \pi/10]$, $[\pi/3 - \pi/10, \pi/3 + \pi/10]$, $[\pi/2 - \pi/10, \pi/2 + \pi/10]$, $[2\pi/3 - \pi/10, 2\pi/3 + \pi/10]$, *and* $[\pi - \pi/10, \pi]$, *correspondingly. The blue solid line is* $f(x)$, *and the data marked by red dots are the value of PhaseDNN at testing data set. (Color available online.)*

It is shown that the convolution calculus for preparing data for $f_j^{\text{shift}}(x)$ costs about 40% of the total computing time. It is quite a large portion and inefficient. Because, in different intervals, $f_j(x)$ can be trained in parallel, PhaseDNN is ideal to take advantage of parallel computing architectures. Although the total computing time is 210 seconds, in practice, the computation can be done in 27 seconds with parallelization. In comparison, a normal single fully connected 24 layer DNN with 640 neurons per hidden layer shows nonconvergence in Figures 4.3(c)–(d) after over 5 hours of training.

4.1.2. Coupled PhaseDNN. 1D problem. We apply the coupled PhaseDNN method to the same test problem (4.1). The frequencies $\{\omega_m\}$ are selected to be $0, 5, 25, 135, 200$. For each A_m and B_m, we also set it as a 1-40-40-40-40-1 DNN.

The training parameters are set to be the same as before. Testing data is 10000 evenly spaced points in $[-\pi, \pi]$. The testing result is shown in Figure 4.3(a). The average L^2 relative training error and testing error are both 1.4×10^{-3}. The pointwise testing error is shown in Figure 4.3(b). It is clear that the error is concentrated in the neighborhood of 0, where the derivative of $f(x)$ is discontinuous. Outside of this neighborhood, the relative maximum error is 8×10^{-3}.

To show the accuracy and efficiency of the coupled PhaseDNN, we try to learn $f(x)$ by a single fully connected DNN. The DNN is set to have 24 hidden layers and 640 neurons in each layer. The training is also carried out with 10000 random training samples, batchsize 2000, and learning rate 0.001. After 50000 epochs during 5 hours of training, the result with a total loss of 100 is shown in Figures 4.3(c)–(d) (blue line). (Color available online.)

One can see that a single fully connected DNN cannot learn this highly oscillated function even with such a large network and a very long training time. The convergence behaviors of a single DNN and a coupled PhaseDNN are shown in Figure 4.3(d). It is shown that the training loss of coupled phaseDNN reduces quickly to $O(10^{-1})$ after 1000 epochs, while the loss of a single DNN stays $O(10^2)$ even after 10000 epochs.

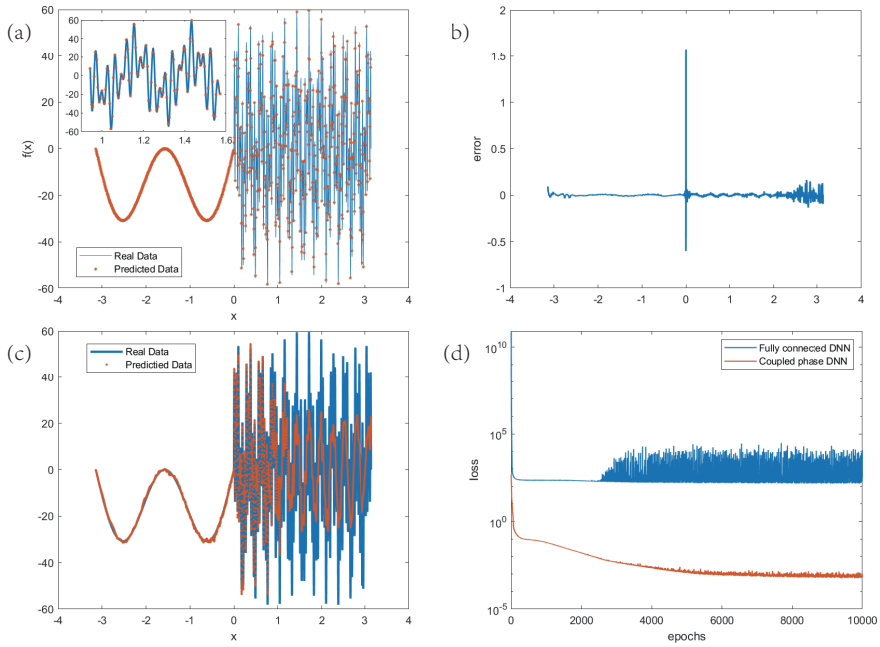

FIG. 4.3. *Fitting result for $f(x)$ using coupled PhaseDNN and a single fully connected DNN.* (a) *Fitting result of couple phase DNN after* $10,000$ *epochs of training. We select frequency* $\{\omega_m\} = \{0, 5, 25, 135, 200\}$. *The blue solid line is real data, while the red dots are the predicted value of couple PhaseDNN. The subplot at the upper left is the local detail plot for interval* $[0.9, 1.6]$. (b) *The pointwise error of coupled PhaseDNN.* (c) *Fitting result of a fully connected DNN after* $50,000$ *epochs of training. The DNN has* 24 *layers and* 640 *neurons in each layer.* (d) *The convergence properties of coupled phase DNN and single fully connected DNN in log scale. The blue line is the training error of fully connected DNN. The red line is the training error of coupled phase DNN. (Color available online.)*

Coupled PhaseDNN is proved efficient in learning high frequency functions.

In fact, 10000 samples are too much for this example. It turns out that 1000 samples can lead to a good approximation of (4.1). Even with 500 samples, which is a much too small a data set for the frequency 203, we can still get a "reasonable" result. The results with 10000 evenly spaced points in $[-\pi, \pi]$ are shown in Figure 4.4.

Shift frequency adaptivity. In general, there is no prior knowledge of the distribution of frequency content in the target function; it will be difficult to prefix the shift frequencies in the PhaseDNN. However, we can adopt an adaptive approach where the shift frequencies ω_m are dynamically added in the buildup of the PhaseDNN.

Let us assume we start with a coupled PhaseDNN $T(x) = \sum_{m=1}^{M} A_m \cos(\omega_m x)$ $+B_m \sin(\omega_m x)$ with some preselected frequencies based on the best knowledge of the target function and assume it has a loss L_M. We continue to train the DNN for another n_0 epochs. If the new loss L'_M does not decrease sufficiently enough, say, $L'_M > \eta L_M$ for some constant η, then we conclude that the coupled PhaseDNN as it is does not contain enough frequencies to learn the target function well. Then, to improve the coupled PhaseDNN, we can add a new frequency ω_{M+1} to $T(x)$ so $T(x) \leftarrow T(x) + A_{M+1} \cos(\omega_{M+1} x) + B_{M+1} \sin(\omega_{M+1} x)$ and train the new coupled PhaseDNN for another n_0 epochs. If the loss decreases significantly, we can continue the training; otherwise we can add another new (higher) frequency again.

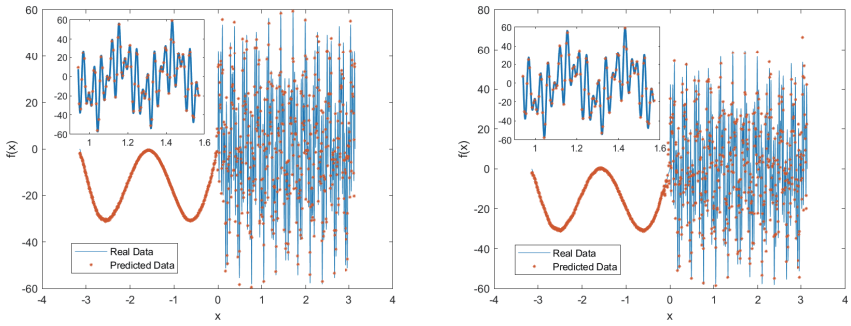

FIG. 4.4. *Fitting result for $f(x)$ using less data. We select frequency $w_n \in \{0, 5, 25, 135, 200\}$. Left panel: Training with $1,000$ random samples. Right panel: training with 500 random samples. The subplots at the upper left in each panel are local detail plots for interval $[0.9, 1.6]$.*

The additional ω_{M+1} should be bigger than all existing ω_m, $m = 1, 2, \ldots M$, where ω_m can be simply set as $\omega_m = (m-1)\Delta K$. The decay parameter η is chosen in $[0.8, 0.9]$. The adaptive phase shift frequency strategy is summarized as follows:

1. Set $T(x) = T_0(x)$ with an initial loss $L = L_0$, $m = 0$.
2. Train the coupled PhaseDNN for n_0 epochs; denote the current loss as L'.
3. If $L' > \eta L$, $0 < \eta < 1$, then choose a new $\omega_{m+1} > \max_{1 \le i \le m} \omega_i$; set $T(x) \leftarrow A_{m+1} \cos(\omega_{m+1} x) + B_{m+1} \sin(\omega_{m+1} x)$.
4. $m \leftarrow m + 1$; back to step 2.
5. Repeat the process until the loss is small enough or ω_m is sufficiently large.

This strategy is tested on the problem in section 4.2.2 with the same training parameters, and $\omega_m = 20m, m = 1, 2, \ldots, 12$. The decay parameter $\eta = 0.9$ while $n_0 = 500$. The loss curve of training is shown in Figure 4.5

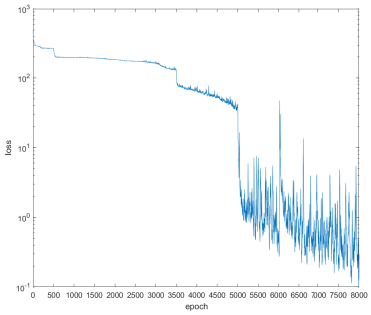

FIG. 4.5. *The loss curve in log scale for fitting $f(x)$ using the adaptive phase range strategy. We use frequency $w_m = 20m$, $m = 1, 2, \ldots, 12$, sequentially and $\eta = 0.9$. The steps on the curve correspond to the adding of frequency 20, 140, and 200.*

One can see that there are three abrupt drops on the loss curve, indicating the significant decrease of loss due to the adaptive procedure; i.e., these three drops at epoch 500, 3500, and 5000 correspond to adding frequency 20, 140, and 200, respectively. We should note that adding more frequencies will lead to larger DNNs and thus more computing cost. Therefore, how to identify the most relevant frequencies so that the coupled PhaseDNN can be most efficient is an important issue demanding

further investigations.

Discontinuous functions and frequency sweep. Next we consider discontinuous functions, and we replace the sin in (4.1) by a square wave function with the same frequency and learn it by (3.4) with $\omega_m \in \{0, 5, 25, 135, 200\}$ and $\omega_m = -1600 : 10 : 1600$. These two DNNs are trained with 10000 samples and 1000 epochs. The results are shown in Figures 4.6(a)–(b). It can be seen that the coupled PhaseDNN has a larger error for a discontinuous function, compared with the smooth sin case. The sweeping strategy is preferred for discontinuous functions. From the plot of errors DFT in Figures 4.6(c)–(d), one may find that the sweeping strategy does learn the information in frequency domain $[-1600, 1600]$. For a frequency larger than 1600, neither strategy can learn it.

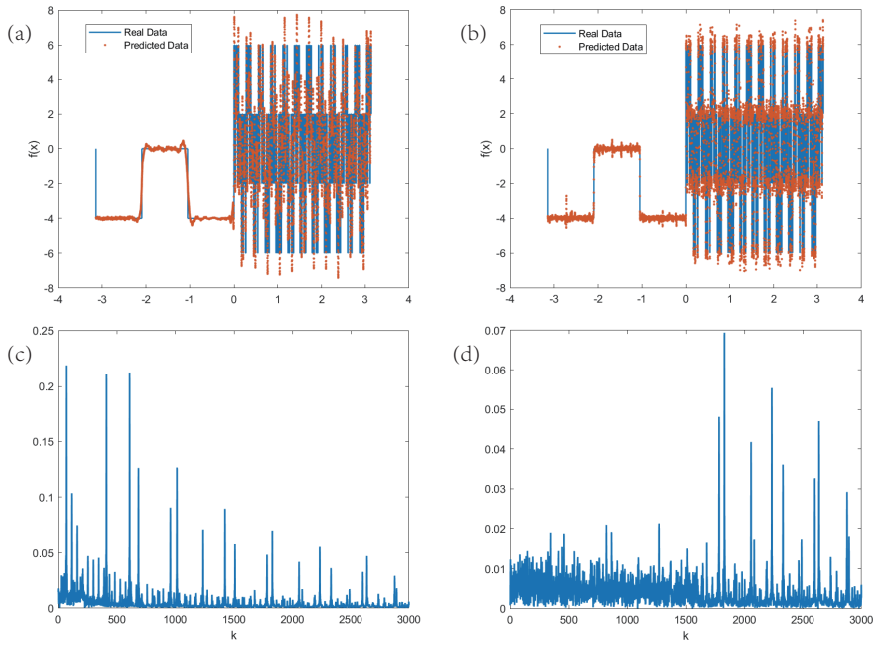

FIG. 4.6. *Fitting result for square wave function using selecting and sweeping methods.* (a) *Fitting result using selecting method with* $\omega_m \in \{0, 5, 25, 135, 200\}$. (b) *Fitting result using sweeping method. Frequency domain is* $[-1600, 1600]$. (c) *DFT of error of selecting method.* (d) *DFT of error of sweeping method.*

2D problem. We use (3.4) to learn 2D and 3D problems. For the 2D test, the function $G(x, y) = g(x)g(y)$ is used, where $g(x)$ is defined by

$$g(x) = \begin{cases} \sin x + \sin 3x & \text{if } x \in [-\pi, 0], \\ \sin 23x + \sin 137x & \text{if } x \in [0, \pi]. \end{cases} \tag{4.2}$$

The function $g(x)$ is the $f(x)$ in (4.1) without the $\sin 203x$ component. In this test, we choose $\{w_m\} \in \{0, 5, 25, 135\} \times \{0, 5, 25, 135\}$. Training settings are $640 \times 640 = 409600$ samples and 80 epochs with batchsize 100. Testing data is 100×100. The result is shown in Figure 4.7(left).

The result is good even for the highest frequency region. In this example, the highest frequency is 137. With more data, we can learn a function of even higher

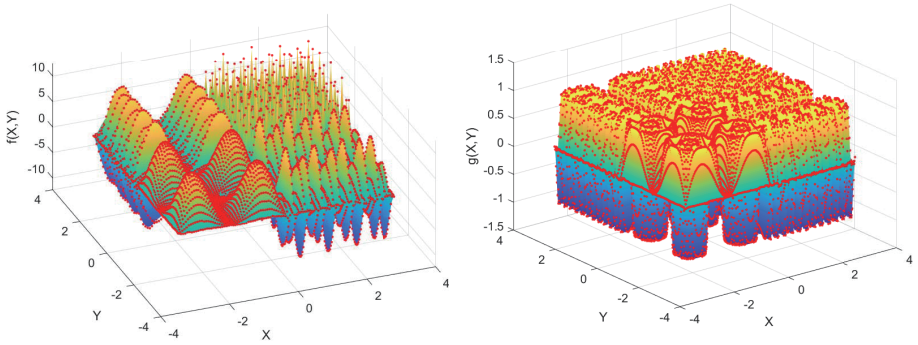

FIG. 4.7. *Fitting results for 2D problems using coupled PhaseDNN. Left panel: Fitting result for $G(x,y)$. Right panel: Fitting result for $\tilde{G}(x,y)$.*

frequency.

Furthermore, we test another problem with $\tilde{G}(x,y) = \sin(\tilde{g}(x)\tilde{g}(y))$ with

$$\tilde{g}(x) = \begin{cases} \sin x + \sin 3x & \text{if } x \in [-\pi, 0], \\ \sin 23x & \text{if } x \in [0, \pi]. \end{cases} \tag{4.3}$$

The highest frequency of $\tilde{G}(x,y)$ is about 200. We use training setups similar to those in the previous test and choose $\omega_m \in \{10:10:210\} \times \{10:10:210\}$. The fitting result is shown in Figure 4.7(right). The L^2 fitting error is 5.2×10^{-3}.

3D problem. The test problem for 3D is $H(x,y,z) = h(x)h(y)h(z)$, where

$$h(x) = \begin{cases} \sin x + \sin 3x & \text{if } x \in [-\pi, 0], \\ \sin 23x + \sin 32x & \text{if } x \in [0, \pi]. \end{cases} \tag{4.4}$$

The selected frequency is $\omega_m \in \{0,5,25,30\} \times \{0,5,25,30\} \times \{0,5,25,30\}$. Training uses $250 \times 250 \times 250 = 1.5625 \times 10^7$ random samples and 150 epochs with batchsize 15625. For plotting, we choose hypersurface $z = 1$ and $z = \frac{1}{2}(x+y)$ as test data. Each A_m, B_m is chosen to be 1-20-20-20-20-1. The relative L^2 error is 3×10^{-2}. Results are shown in Figure 4.8.

The number of data. Basically, the data set must be big enough that it can reveal all the frequencies. That means we still need $O(N^d)$ data. For each direction, N samples must reveal the highest frequency of this direction. This means that, even though DNN has the advantage that the number of unknowns p increases linearly with respect to the number of dimensions, we still need an exponentially large data set. In our 3D test, we use 250 samples to reveal frequency 32 on average; the total number 1.56×10^7 is really a very big data set.

It is clear that the PhaseDNN approach cannot overcome curse of dimensionality. If we have no prior knowledge on the frequency distribution, coupled phaseDNN can be considered as building a mesh in Fourier space. Thus, in general, the number of ω_m will increase exponentially. In our 3D test problem, there are 172 different ω_m, which corresponds to 343 sub DNNs. The whole coupled phaseDNN $T(x)$ has a width over 6000. It is a shallow but very wide DNN. The number of parameters is large. With a large number of data, the whole training takes about 8 hours.

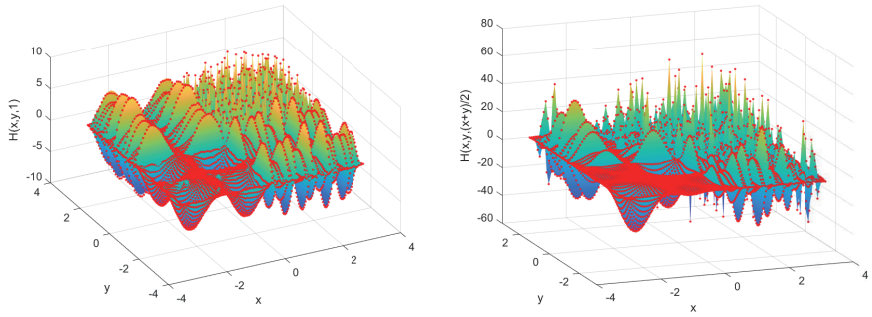

FIG. 4.8. *Fitting result for $H(x, y, z)$. Left panel: Fitting result on hypersurface $z = 1$. Right panel: Fitting result on hypersurface $z = \frac{1}{2}(x + y)$.*

4.2. Coupled PhaseDNN for solving PDEs with high frequency solutions.

4.2.1. Helmholtz equation with constant wave numbers. We will solve the constant coefficient case for (3.9), namely, $c = 0$ with zero boundary condition $u(-1) = u(1) = 0$ and the following high oscillatory forcing term

$$f(x) = (\lambda^2 - \mu^2)\sin(\mu x). \tag{4.5}$$

We set $\omega_m \in \{0, \lambda, \mu\}$ and each A_m, B_m to be 1-40-40-40-40-1 DNN. The whole $T(x, \theta)$ is trained with 10000 evenly spaced samples, 100 epochs, and batchsize 100. We choose four special cases: $\lambda = 3, \mu = 2$; $\lambda = 200, \mu = 2$; $\lambda = 2, \mu = 200$; and $\lambda = 300, \mu = 200$. The result is shown in Figure 4.9.

The training takes about 5 minutes with a maximum error of $O(10^{-4})$. For comparison, a single fully connected DNN with a scale similar to that of $T(x)$ cannot solve the equation at all when the frequency is high. The training results after 1500 epochs for $\lambda = 3, \mu = 2$ and $\lambda = 200, \mu = 2$ are shown in Figure 4.10, showing the nonconvergence for the high frequency solution using a common fully connected DNN (Figure 4.10(right)).

Next, we will consider the case of a more complicated solution beyond plane waves with an exact solution $u(x) = J_0(\mu x) + 0.2\cos(\lambda x)$, where $J_0(x)$ is the 0-order Bessel function. For this case, the forcing term in (3.9) is $f(x) = \mu^2 J_0''(\mu x) + \lambda^2 J_0(\mu x)$ with corresponding nonzero Dirichlet boundary conditions. We choose $\omega_j \in \{0, \lambda, \mu\}$ in a coupled PhaseDNN, which gives accurate numerical results for $\lambda = 200, \mu = 100$ as shown in Figure 4.11

4.2.2. Helmholtz equation with variable wave numbers. Next we solve problem (3.9) with $u(-1) = u(1) = 0$ and $c > 0$, and a variable wave number

$$\omega(x) = \sin(mx^2), \tag{4.6}$$

where $m > 0$ is a constant. As there is no explicit exact solution to this equation, the numerical solution obtained by a finite difference method with a fine mesh will be used as the reference solution.

Differential equation formulation. We will first find the solution by solving the Helmholtz differential equation with a coupled PhasedDNN.

We consider the parameters $\lambda = 2$, $\mu = 200$, $c = 0.9\lambda^2 = 3.6$, and $m = 1$ in (3.9), which corresponds to a high frequency external wave source and a low wave

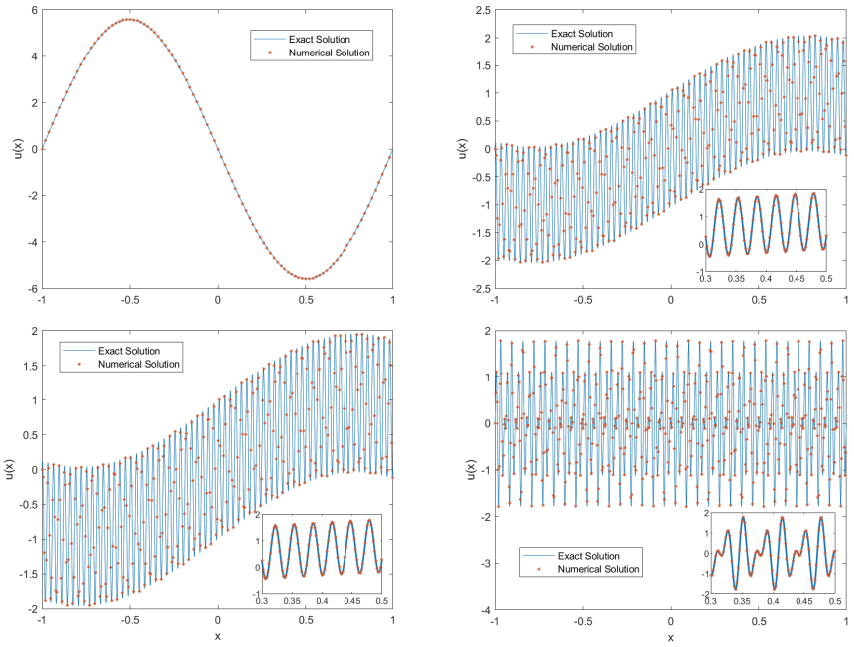

FIG. 4.9. *Numerical and exact solution of problem* (3.9) *with* $c = 0$ *and different* λ *and* μ. (a) $\lambda = 3, \mu = 2$. (b) $\lambda = 200, \mu = 2$. (c) $\lambda = 2, \mu = 200$. (d) $\lambda = 300, \mu = 200$. *The subplots in figures* (b), (c), *and* (d) *are local detail plots for interval* $[0.3, 0.5]$.

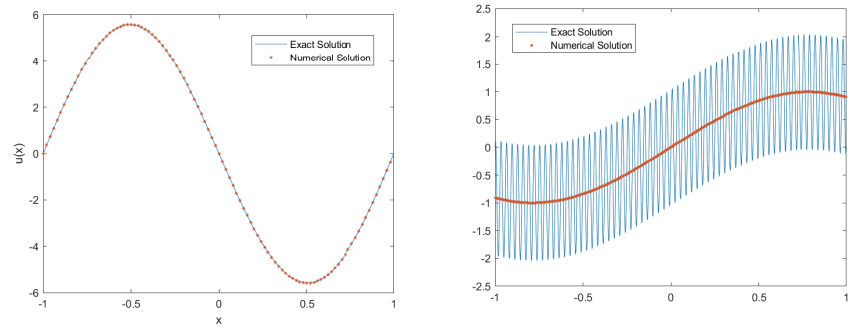

FIG. 4.10. *Nonconvergence of usual fully connected DNN for high frequency case: Numerical and exact solutions of problem* (3.9) *with different* λ *and* μ. *Left panel:* $\lambda = 3, \mu = 2$. *Right panel:* $\lambda = 200, \mu = 2$.

number with small background media inhomogeneity. In the coupled PhaseDNN, we choose $w_m \in \{1, 2, 3, 4, 200\}$. Other training parameters are set to be similar to those in the constant coefficient case. The numerical results of coupled PhaseDNN and the reference solution are shown in Figure 4.12, and the absolute error is on the order of $O(10^{-3})$.

Next, we choose $\lambda = 100$, $\mu = 200$, $c = 0.1\lambda^2 = 1000$, and $m = 100$, which corresponds to a high frequency external wave source and a high wave number with larger background media inhomogeneity. ω_m is chosen to be $\{0, 90, 100, 110, 190, 200, 210\}$.

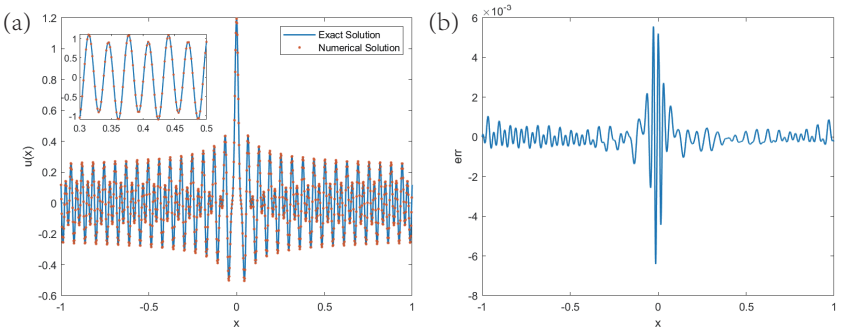

FIG. 4.11. *Numerical and exact solutions of problem* (3.9) *with exact solution* $J_0(\mu x) + 0.2\cos(\lambda x)$. *We choose* $\lambda = 200$, $\mu = 100$. (a) *Numerical and exact solutions of problem* (3.9). *Blue line: Exact solution. Red dots: Numerical solution obtained by coupled PhaseDNN. The subplot at the upper left is the local detail plot for interval* $[0.3, 0.5]$. (b) *Error of the numerical solution. (Color available online.)*

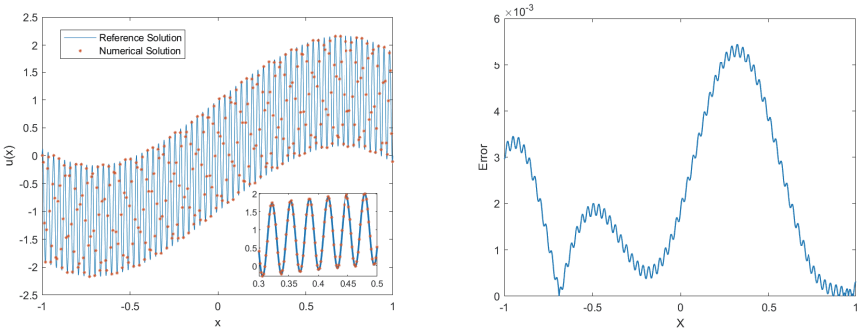

FIG. 4.12. *Variable coefficient Helmholtz equations* (3.9)*: Numerical and exact solutions using coupled phaseDNN.* $\lambda = 2$, $\mu = 200$, $c = 3.6$, *and* $m = 1$. *Left panel: The numerical solution and reference solution obtained by the finite difference method. The subplot at the lower right is the local detail plot for interval* $[0.3, 0.5]$. *Right panel: The absolute value of the difference between the numerical solution and the reference solution.*

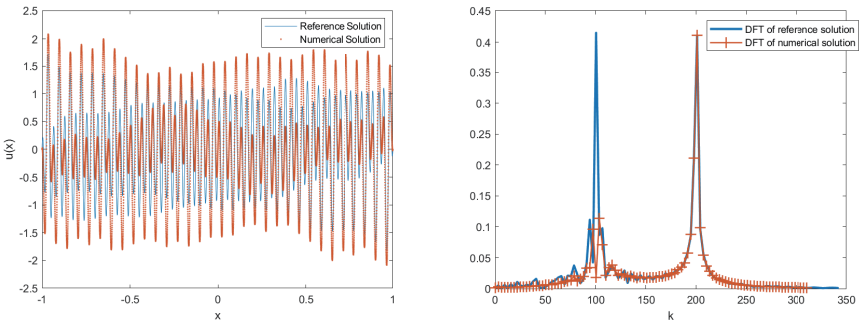

FIG. 4.13. *Numerical and reference solutions to problem* (3.9) *using coupled PhaseDNN.* $\lambda = 100$, $\mu = 200$, $c = 0.1\lambda^2$, *and* $m = 100$. *Left panel: The numerical solution obtained by the least squares based coupled PhaseDNN method and the reference solution. Right panel: The discrete Fourier transform of the reference solution and the numerical solution. Blue line: DFT of reference solution. Red line: DFT of PhaseDNN numerical solution. (Color available online.)*

The learning result is shown in Figure 4.13. One can see in the Fourier space for the solution (the right panel of Figure 4.13) that coupled PhaseDNN with least squares residual of the differential equation shows the unresolved error at the $\pm\lambda$ frequency while the μ frequency converges well. This error is due to the behavior of the Fourier symbol of the differential operator at the λ frequency and turns out to be difficult to avoid. A theoretical analysis and a remedy for this phenomenon will be carried out in a future paper.

Discontinuous coefficient—waves in layered media. We apply the PhaseDNN to the wave propagation in a two layer medium. Let $c = 1$ in (3.9) and

$$\omega(x) = \begin{cases} -0.75\lambda^2 & \text{if } x < 0, \\ 0 & \text{if } x \geqslant 0. \end{cases}$$

This equation models a wave in a two layer medium with an interface at $x = 0$ where transmission conditions are imposed. On the left of the boundary, the wave number is $\lambda/2$, and on the right it is λ. With Dirichlet boundary conditions $u(-1) = u(1) = 0$, the numerical result is shown in Figure 4.14. The parameters here are $\lambda = 50, \mu = 200$. We choose $\omega_j \in \{0, \lambda/2, \lambda, \mu\}$ for the coupled PhaseDNN.

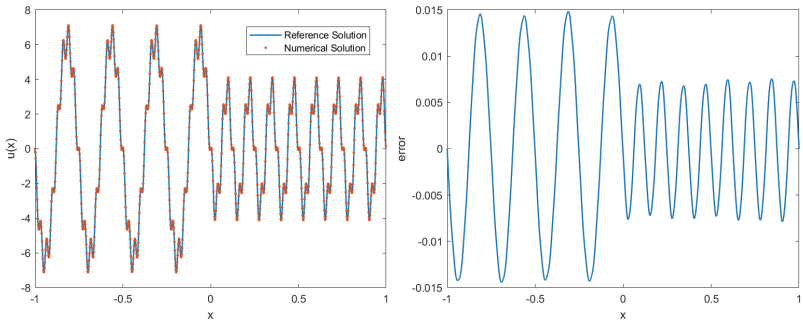

FIG. 4.14. *Discontinuous coefficient Helmholtz equations* (3.9)*: Numerical and reference solutions to two layer media problem using coupled PhaseDNN. $\lambda = 50$, $\mu = 200$. Left panel: The numerical solution obtained by the least squares based coupled PhaseDNN method and the reference solution. Right panel: The difference between the numerical solution and the reference solution.*

Integral equation formulation. Next, we will apply the integral equation approach (3.15), (3.21) to this problem. The Green's function to $u'' + \lambda^2 u = \delta(x - x')$ with Dirichlet boundary condition $u(-1) = u(1) = 0$ is given by

$$G(x', x) = \begin{cases} \dfrac{(-\tan\lambda\cos\lambda x' + \sin\lambda x')(\tan\lambda\cos\lambda x + \sin\lambda x)}{2\lambda\tan\lambda}, & s \leq x, \\ \dfrac{(\tan\lambda\cos\lambda x' + \sin\lambda x')(-\tan\lambda\cos\lambda x + \sin\lambda x)}{2\lambda\tan\lambda}, & s > x. \end{cases} \tag{4.7}$$

With the same parameter setting, the numerical solution obtained by integral equation method is shown in Figure 4.15. The absolute error is on the order of $O(10^{-3})$. It is clear that coupled PhaseDNN with least squares residual of the integral equation (3.15), (3.21) gives a much more accurate solution than that with the differential equation method and does not suffer from high wave number errors as in Figure 4.13.

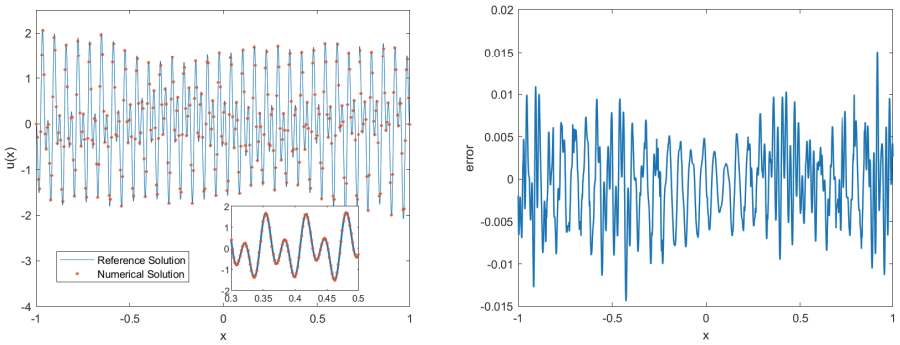

FIG. 4.15. *Numerical and reference solutions of problem* (3.9) *using coupled PhaseDNN and the integral equation method.* $\lambda = 100$, $\mu = 200$, $c = 1000$, *and* $m = 100$. *Left panel: The numerical solution with the reference solution. The subplot is the local detail plot for interval* $[0.3, 0.5]$. *Right panel: The difference between the numerical solution and the reference solution.*

4.2.3. Solving the elliptic equation. We can also solve the elliptic differential equation with high frequency external sources using the coupled PhaseDNN. We consider a test problem

$$\begin{cases} u'' - \lambda^2 u = -(\lambda^2 + \mu^2)\sin(\mu x), \\ u(-1) = u(1) = 0, \end{cases} \tag{4.8}$$

which has an exact solution as

$$u(x) = -\frac{\sin\mu}{\sinh\lambda}\sinh(\lambda x) + \sin(\mu x). \tag{4.9}$$

We choose $\lambda = 3$, $\mu = 250$ in (4.8). To solve this equation, we set a coupled PhaseDNN with $\omega_m \in \{0, \mu\}$. Each subnetwork is a fully connected DNN with 4 layers and 20 neurons in each layer. An accurate training result is shown in Figure 4.16.

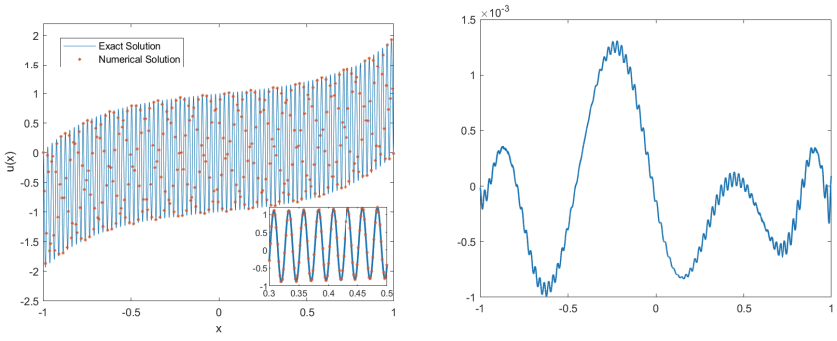

FIG. 4.16. *The numerical solution of* (4.8) *using coupled PhaseDNN.* $\lambda = 3$, $\mu = 250$. *Left panel: Numerical solution (in red) and exact solution (in blue) of* (4.8). *The subplot is the local detail plot for interval* $[0.3, 0.5]$. *Right panel: The error of numerical solution. (Color available online.)*

4.2.4. Coupled PhaseDNN for solving the exterior wave scattering problem. We consider problem (3.12) with $\lambda = 100$, $\mu = 200$, $c = 0.1\lambda^2$. The variable wave coefficient is

$$\omega(x) = \chi_{[-1,1]}(x)\sin(1-x^2), \tag{4.10}$$

and the forcing term is

$$f(x) = \chi_{[-1,1]}(x)(\lambda^2-\mu^2)(1-x^2)\sin(\mu x). \tag{4.11}$$

We first solve the problem with radiation boundary condition (3.13), which is an exact absorbing boundary condition in this case, by the coupled PhaseDNN for the differential equation. The real part of the solution is shown in Figure 4.17. We set each subnetwork in (3.4) to be a 1-20-20-20-20-1 DNN. The training data set is 3000 evenly spaced points in $[-2, 2]$. The training runs 3000 epochs with batchsize 600.

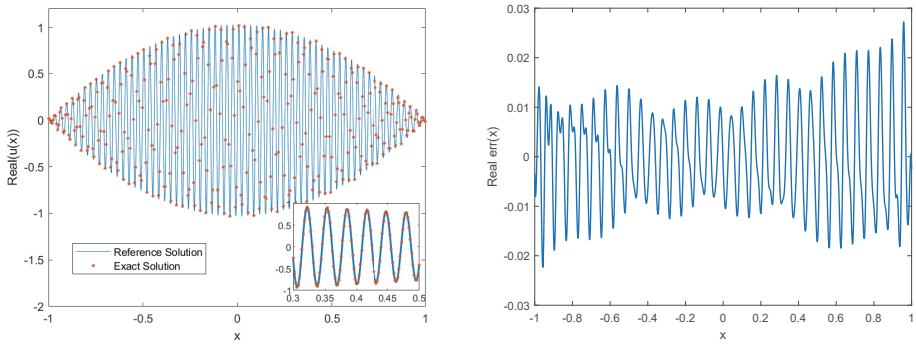

FIG. 4.17. *The result of the exterior problem using coupled PhaseDNN for the differential equation* (3.12) *after* 3,000 *epochs training. Left panel: The real part of the numerical and reference solutions to the exterior problem. Blue line: Reference solution. Red dots: Numerical solution. The subplot is the local detail plot for interval* $[0.3, 0.5]$. *Right panel: The error of the real part of the numerical solution. (Color available online.)*

Again, this problem will be solved by the coupled PhaseDNN with the integral equation (3.15), (3.21), and the result is given in Figure 4.18. The training parameters are set similarly to those for the differential equation method. Training runs 300 epochs. Better performance of the coupled PhaseDNN for the integral equation approach (3.15), (3.21), requiring much fewer training epochs, is shown, compared with the differential equation coupled PhaseDNN. Again, we can see that the PhaseDNN with an integral equation formulation gives much better results than that with a differential equation formulation.

4.3. PhaseDNN as a meshless solver for 2D Helmholtz equation in a complex domain. Since the coupled PhaseDNNs based on least squares loss do not require a mesh, this method is ideal for handling complex domains without expensive meshing costs as in traditional finite element methods. We consider the following 2D Helmholtz equation in a domain Ω of an equilateral triangle with a circle inside removed. The bottom of the triangle is $\{(x, -\frac{\sqrt{3}}{3})| -1 \leqslant x \leqslant 1\}$, the center of the circle is located at the center of the triangle, and the radius is $1/\sqrt{12}$.

$$\begin{cases} \Delta u + \lambda^2 u = f(x,y) & \text{if } (x,y) \in \Omega, \\ u(x,y) = g(x,y) & \text{if } (x,y) \in \partial\Omega. \end{cases} \tag{4.12}$$

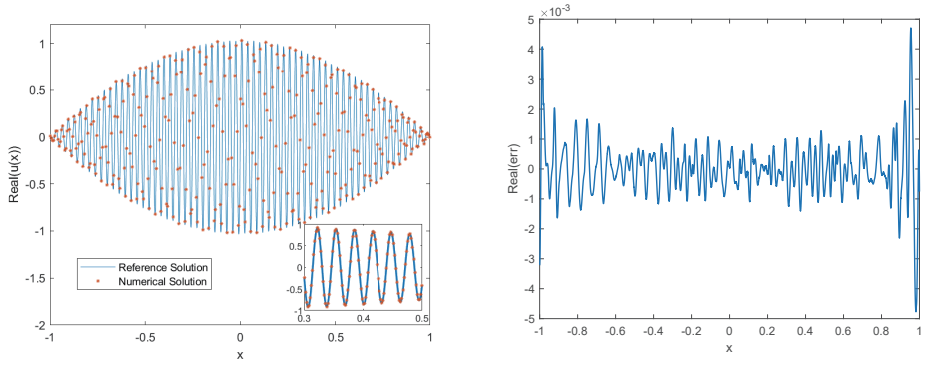

FIG. 4.18. *The result of the exterior problem using the integral equation method after* 300 *epochs of training. Left panel: The real part of the numerical and reference solutions to the exterior problem. Blue line: Reference solution. Red dots: Numerical solution. The subplot is the local detail plot for interval* $[0.3, 0.5]$. *Right panel: The error of the real part of the numerical solution. (Color available online.)*

The exact solution is chosen as $u(x,y) = \exp(\sin(\mu_1 x)\sin(\mu_2 y))$. $f(x,y), g(x,y)$ in (4.12) is chosen according to the differential equation. In the numerical test, the parameters are $\lambda = 100$, $\mu_1 = 30$, $\mu_2 = 50$. A coupled PhaseDNN with $\omega_j \in \{0, \lambda, \mu_1, \mu_2, 2\mu_1, 2\mu_2\} \times \{0, \lambda, \mu_1, \mu_2, 2\mu_1, 2\mu_2\}$ gives an accurate numerical solution as shown in Figure 4.19

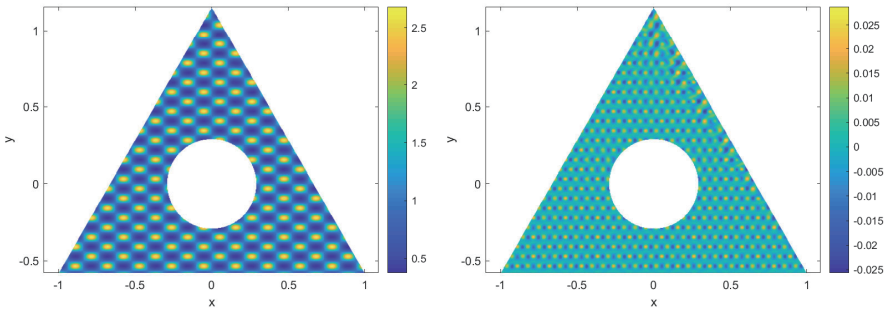

FIG. 4.19. *The numerical solution and error of* 2*D Helmholtz equation* (4.12) *in a complicated domain.* $\lambda = 100$, $\mu_1 = 30$, $\mu_2 = 50$. *Left panel: The color map of the numerical solution. Right panel: The absolute error of the numerical solution.*

5. Conclusion and future work. In this paper, we have proposed a phase shift DNN to learn high frequency information by using frequency shifts to convert the high frequency learning to a low frequency one. As shown by various numerical tests, this approach increases dramatically the capability of the DNN as a viable tool for approximating high frequency functions and solutions of high frequency wave differential and integral equations in inhomogeneous media as a meshless numerical method for PDEs and integral equations.

The optimization problem with the training of DNNs is complex and not well

understood during the search of parameter spaces of the DNNs for a local or global minima. The specific structure of the proposed PhaseDNN seems to provide a favorable parameter structure (inspired by the mathematical or physical properties of the solutions), within which the optimization can be carried much more efficiently than the structures of the common fully connected DNNs can allow. Moreover, our numerical results also show that the PhaseDNN with integral equation formulation of the high frequency wave problems gives better accuracy than that with differential equations. All these issues will be the subject of a future theoretical analysis of the PhaseDNN.

For our future studies, we will explore the generalization capability of the DNNs to predict high frequency wave solutions for general material property parameters once the DNNs are trained for some selected parameters and, furthermore, to use this generalization feature to study high dimensional random material parameter spaces related to high frequency wave propagations and scattering in terms of the random media's correlations and other stochastic properties.

Appendix A. In this appendix, we will show that under a condition that the weight of the input layer for each sub DNN $T_m(x)$ is small enough, coupled PhaseDNN is approximately equivalent to parallel PhaseDNN in approximating a function during training.

For simplicity, we only consider the 1D case $d = 1$ and assume $|\omega_i - \omega_j| = |i-j|\Delta k$ for all $1 \leqslant i, j \leqslant M$ and some $\Delta k > 0$.

Generally, the Fourier transform of a target function $f(x)$ and DNN function $T(x)$ may not exist. However, we are only interested in the target function in a compact domain $\Omega \subset \mathbb{R}^d$. To avoid this technical problem in analysis, we choose a smooth mollifier function $\kappa(x)$ that satisfies $\kappa(x) = 1$ for $x \in \Omega$ and $\kappa(x) = 0$ for $x \in \mathbb{R}^d \backslash \Omega'$, where $\Omega \subset \Omega' \subset \mathbb{R}^d$. We assume $f(x)\kappa(x) \in H^r$ and $T(x)\kappa(x) \in H^r$ for some $r \geq 1$. This assumption ensures the existence of a Fourier transform for $f(x)\kappa(x)$ and $T(x)\kappa(x)$. For simplicity purposes, in the following text, we still use $f(x)$ and $T(x)$ to indicate $f(x)\kappa(x)$ and $T(x)\kappa(x)$.

Although in practice we usually use the real form of coupled phase DNN (3.4), in analysis we prefer the complex form (3.1). Recall that $T_m(x)$ in (3.1) is complex-valued and each DNN T_m in (3.1) can be written as $T_m(x) = u_m(\eta w_m x + b_m)$, where $\eta > 0$ is a small parameter, and $u_m(t)$ are (complex-valued) DNN functions except the input layer. Again, in general, $u_m(x)$ is not an $\mathbb{L}^2$ function; however, as we are not interested in its behavior at infinity, we can multiply it by a smooth mollifier function $\kappa(x)$ s.t. $\kappa(x) = 1$ in a compact domain $\Omega \subset \mathbb{R}$ and $\kappa(x) = 0$ in $\mathbb{R}\backslash\Omega'$. Here, the compact domain Ω' satisfies $\Omega \subset \Omega' \subset \mathbb{R}$. We assume the new function $u(x)\kappa(x) \in H^r(\mathbb{R})$, $r \geq 1$. For simplicity, in the following text, $u(x)$ stands for the H^r function $u(x)\kappa(x)$. Without losing generality, we can assume $w_m \neq 0$ for all $1 \leqslant m \leqslant M$.

We know that $u_m(x, \theta) \in L^2(\mathbb{R})$, where $\theta = \theta(t)$; t stands for the training process. It is reasonable to assume the following.

ASSUMPTION 1. *During the training $t \in [0, T]$, there is a constant $H > 0$ s.t.*

$$H^{-1} < \inf_{t\in[0,T]} \|u_m\|_2 \leqslant \sup_{t\in[0,T]} \|u_m\|_2 < H$$

holds for any $1 \leqslant m \leqslant M$.

This assumption is reasonable because if $\|u\|_2$ has no upper bound, this means the training blows up; while if it has no positive lower bound, this means this term

vanishes. The constant is uniform because M is finite. Furthermore, we can assume the following.

ASSUMPTION 2. *For any $\epsilon > 0$, there is constant $A > 0$ s.t.*

$$\int_A^{+\infty} |\hat{u}_m|^2 \,\mathrm{d}k < \epsilon, \text{ and } \int_{-\infty}^{-A} |\hat{u}_m|^2 \,\mathrm{d}k < \epsilon$$

holds for any $1 \leqslant m \leqslant M$ and $t \in [0, T]$.

It is straightforward to show that the Fourier transform of (3.1) yields

$$\hat{T}(k) = \sum_{m=1}^{M} \hat{T}_m(k - \omega_m) = \sum_{m=1}^{M} \frac{1}{\eta|w_m|} \exp\left(- \frac{ib_m(k-\omega_m)}{\eta w_m}\right) \hat{u}_m\left(\frac{k-\omega_m}{\eta w_m}\right) \tag{A.1}$$

and

$$L(\theta) = \int_{-\infty}^{\infty} |\hat{f} - \hat{T}|^2 \,\mathrm{d}k = \int_{-\infty}^{\infty} |\hat{f}|^2 \,\mathrm{d}k - 2Re \int_{-\infty}^{\infty} \hat{T}\bar{\hat{f}} \,\mathrm{d}k + \int_{-\infty}^{\infty} |\hat{T}|^2 \,\mathrm{d}k. \tag{A.2}$$

In (A.2), $\int_{-\infty}^{\infty} |\hat{T}|^2 \,\mathrm{d}k = \int_{-\infty}^{\infty} \sum_{m,n=1}^{M} \hat{T}_m(k-\omega_m)\bar{\hat{T}}_n(k-\omega_n) \,\mathrm{d}k$, we will show that under Assumptions 1 and 2, when $\eta \to 0$, for all $m = 1, 2, \ldots, M$ and $n \neq m$,

$$\frac{\int_{-\infty}^{\infty} |T_m(k-\omega_m)||T_n(k-\omega_n)| \,\mathrm{d}k}{\int_{-\infty}^{\infty} |T_m(k-\omega_m)|^2 \,\mathrm{d}k} \to 0. \tag{A.3}$$

To this end, we first notice that

$$\begin{aligned} \int_{-\infty}^{\infty} |T_m(k-\omega_m)|^2 \,\mathrm{d}k &= \frac{1}{\eta^2 w_m^2} \int_{-\infty}^{\infty} \left| \hat{u}_m\left(\frac{k-\omega_m}{\eta w_m}\right)\right|^2 \mathrm{d}k \\ &= \frac{1}{\eta\,|w_m|} \int_{-\infty}^{\infty} |\hat{u}_m(t)|^2 \,\mathrm{d}t = \frac{\|u_m\|_2^2}{\eta\,|w_m|} > \frac{1}{\eta H\,|w_m|}. \end{aligned} \tag{A.4}$$

At the same time,

$$\begin{aligned} &\int_{-\infty}^{\infty} |T_m(k-\omega_m)||T_n(k-\omega_n)| \,\mathrm{d}k = \frac{1}{\eta^2\,|w_m|\,|w_n|} \int_{-\infty}^{\infty} \left|\hat{u}_m\left(\frac{k-\omega_m}{\eta w_m}\right)\right| \left|\hat{u}_n\left(\frac{k-\omega_n}{\eta w_n}\right)\right| \mathrm{d}k \\ &\quad = \frac{1}{\eta\,|w_n|} \int_{-\infty}^{\infty} |\hat{u}_m(t)| \left|\hat{u}_n\left(\frac{w_m}{w_n}t + \frac{\omega_m - \omega_n}{\eta w_n}\right)\right| \mathrm{d}t = \frac{1}{\eta\,|w_n|} \left(\int_{-\infty}^{-A} + \int_{-A}^{A} + \int_{A}^{\infty} \right). \end{aligned} \tag{A.5}$$

Applying the Cauchy–Schwarz inequality, we can estimate the first integral by

$$\begin{aligned} &\int_{-\infty}^{-A} |\hat{u}_m(t)| \left|\hat{u}_n\left(\frac{w_m}{w_n}t + \frac{\omega_m - \omega_n}{\eta w_n}\right)\right| \mathrm{d}t \\ &\qquad \leqslant \left(\int_{-\infty}^{-A} |\hat{u}_m(t)|^2 \,\mathrm{d}t \right)^{1/2} \left(\int_{-\infty}^{-A} \left|\hat{u}_n\left(\frac{w_m}{w_n}t + \frac{\omega_m - \omega_n}{\eta w_n}\right)\right|^2 \mathrm{d}t \right)^{1/2} \\ &\qquad \leqslant \sqrt{\epsilon \left|\frac{w_n}{w_m}\right|}\, \|\hat{u}_n\|_2 < \epsilon \sqrt{\left|\frac{w_n}{w_m}\right|} H. \end{aligned} \tag{A.6}$$

Similarly, we can also estimate

$$\int_A^\infty |\hat{u}_m(t)|\left|\hat{u}_n\left(\frac{w_m}{w_n}t+\frac{\omega_m-\omega_n}{\eta w_n}\right)\right|\,\mathrm{d}t \leqslant \epsilon\sqrt{\left|\frac{w_n}{w_m}\right|}H. \tag{A.7}$$

For the second integral, we have

$$\begin{aligned}
&\int_{-A}^{A} |\hat{u}_m(t)|\left|\hat{u}_n\left(\frac{w_m}{w_n}t+\frac{\omega_m-\omega_n}{\eta w_n}\right)\right|\,\mathrm{d}t \\
&\qquad \leqslant \left(\int_{-A}^{A}|\hat{u}_m(t)|^2\,\mathrm{d}t\right)^{1/2}\left(\int_{-A}^{A}\left|\hat{u}_n\left(\frac{w_m}{w_n}t+\frac{\omega_m-\omega_n}{\eta w_n}\right)\right|^2\,\mathrm{d}t\right)^{1/2} \\
&\qquad \leqslant \sqrt{\left|\frac{w_n}{w_m}\right|}\,\|u_m\|_2\left(\int_{\frac{|\omega_m-\omega_n|}{\eta|w_n|}-\left|\frac{w_m}{w_n}\right|A}^{\frac{|\omega_m-\omega_n|}{\eta|w_n|}+\left|\frac{w_m}{w_n}\right|A}|\hat{u}_n(t)|^2\,\mathrm{d}t\right)^{1/2}.
\end{aligned} \tag{A.8}$$

Since $n\neq m$, $\omega_m-\omega_n\neq 0$. When $\eta<\frac{|\omega_m-\omega_n|}{|w_m|+|w_n|}$,

$$\int_{\frac{|\omega_m-\omega_n|}{\eta|w_n|}-\left|\frac{w_m}{w_n}\right|A}^{\frac{|\omega_m-\omega_n|}{\eta|w_n|}+\left|\frac{w_m}{w_n}\right|A}|\hat{u}_n(t)|^2\,\mathrm{d}t<\int_A^\infty|\hat{u}_n(t)|^2\,\mathrm{d}t<\epsilon.$$

To choose a uniform η, it is sufficient to set

$$\eta<\min_{m\neq n}\frac{|\omega_m-\omega_n|}{|w_m|+|w_n|}=\frac{\Delta k}{2\max_m|w_m|}>0.$$

Combining (A.6), (A.7), (A.8), and (A.4), we can conclude that

$$\frac{\int_{-\infty}^{\infty}|T_m(k-\omega_m)||T_n(k-\omega_n)|\,\mathrm{d}k}{\int_{-\infty}^{\infty}|T_m(k-\omega_m)|^2\,\mathrm{d}k}\to 0\quad(\eta\to 0)$$

holds for all $1\leqslant m\neq n\leqslant M$. Thus, when η is small enough, we can estimate

$$\begin{aligned}
\int_{-\infty}^{\infty}|\hat{T}|^2\,\mathrm{d}k&=\int_{-\infty}^{\infty}\sum_{m,n=1}^{M}\hat{T}_m(k-\omega_m)\bar{\hat{T}}_n(k-\omega_n)\,\mathrm{d}k\\
&\approx\sum_{m=1}^{M}\int_{-\infty}^{\infty}\left|\hat{T}_m(k-\omega_m)\right|^2\,\mathrm{d}k.
\end{aligned} \tag{A.9}$$

We can do decomposition of $\hat{f}(k)=\sum_{m=1}^{M}\hat{f}(k)\chi_m(k)=\sum_{m=1}^{M}\hat{f}_m$, where $\chi_m(k)$ is the indicator function of interval $[\omega_m-\Delta k/2,\omega_m+\Delta k/2]$. It is easy to see $|\hat{f}|^2=\sum_{m=1}^{M}|\hat{f}_m|^2$ and

$$\int_{-\infty}^{\infty}\hat{T}\bar{\hat{f}}\,\mathrm{d}k=\sum_{m,n=1}^{M}\int_{-\infty}^{\infty}\hat{T}_m(k-\omega_m)\hat{f}_n(k)\,\mathrm{d}k.$$

With the same argument we used in proving (A.3), we can also deduce that if $\hat{f}_n\neq 0$,

$$\frac{\int_{-\infty}^{\infty}\left|\hat{T}_m(k-\omega_m)\right||\hat{f}_n|\,\mathrm{d}k}{\int_{-\infty}^{\infty}\left|\hat{T}_n(k-\omega_n)\right||\hat{f}_n|\,\mathrm{d}k}\to 0 \tag{A.10}$$

for all $m = 1, 2, \ldots, M$ and $n \neq m$. Thus

$$\int_{-\infty}^{\infty} \hat{T}\bar{\hat{f}}\,\mathrm{d}k \approx \sum_{m=1}^{M} \int_{-\infty}^{\infty} \hat{T}_m(k-\omega_m)\bar{\hat{f}}_m. \tag{A.11}$$

Substituting approximations (A.9) and (A.11) into loss function (A.2), a good approximation of loss function, when η is small, is

(A.12)

$$\begin{aligned} L(\theta) &\approx \int_{-\infty}^{\infty} \sum_{m=1}^{M} \left|\hat{f}_m(k)\right|^2 - 2Re\hat{T}_m(k-\omega_m)\bar{\hat{f}}_m + \left|\hat{T}_m(k-\omega_m)\right|^2 \mathrm{d}k \\ &= \sum_{m=1}^{M} \int_{-\infty}^{\infty} \left|\hat{f}_m(k) - \hat{T}_m(k-\omega_m)\right|^2 \mathrm{d}k = \sum_{m=1}^{M} \int_{-\infty}^{\infty} \left|\mathcal{F}^{-1}[\hat{f}_m(k+\omega_m)] - T_m(x)\right|^2 \mathrm{d}x \\ &= \sum_{m=1}^{M} \int_{-\infty}^{\infty} \left|e^{-i\omega_m x}\mathcal{F}^{-1}[\hat{f}_m](x) - T_m(x)\right|^2 \mathrm{d}x. \end{aligned}$$

Here, $e^{-i\omega_m x}\mathcal{F}^{-1}[\hat{f}_m](x)$ is exactly the $f_m^{\text{shift}}(x)$ in (2.12), whose support in frequency space is $[-\Delta k/2, \Delta k/2]$. Hence, the total loss function $L(\theta)$ is approximately the sum of M individual DNNs T_m that learn $f_m^{\text{shift}}(x)$ separately.

Acknowledgment. The authors would like to thank a reviewer for suggesting the work in [15] and [11].

REFERENCES

[1] P. B. Bochev and M. D. Gunzburger, *Finite element methods of least-squares type*, SIAM Rev., 40 (1998), pp. 789–837, https://doi.org/10.1137/S0036144597321156.
[2] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.
[3] J. Bruna and S. Mallat, *Invariant scattering convolution networks*, IEEE Trans. Pattern Anal. Machine Intel., 35 (2013), pp. 1872–1886.
[4] W. Cai, X. Li, and L. Liu, *PhaseDNN—A Parallel Phase Shift Deep Neural Network for Adaptive Wideband Learning*, preprint, https://arxiv.org/abs/1905.01389, 2019.
[5] W. Cai and Z.-Q. Xu, *Multi-Scale Deep Neural Networks for Solving High Dimensional PDEs*, arXiv preprint, https://arxiv.org/abs/1910.11710, 2019.
[6] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. F. Greengard, J. F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao, *A wideband fast multipole method for the Helmholtz equation in three dimensions*, J. Comput. Phys., 216 (2006), pp. 300–325.
[7] E. Darve and P. Havé, *A fast multipole method for Maxwell equations stable at all frequencies*, Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., 362 (2004), pp. 603–628.
[8] I. Daubechies, *Ten Lectures on Wavelets*, CBMS-NSF Reg. Conf. Ser. Appl. Math. 61, SIAM, Philadelphia, 1992.
[9] W. E and B Yu, *The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, Commun. Math. Stat., 6 (2018), pp. 1–12.
[10] Y. Fan, C. O. Bohorquez, and L. Ying, *BCR-Net: A neural network based on the nonstandard wavelet form*, J. Comput. Phys., 384 (2019), pp. 1–15.
[11] J. Fang, J. Qian, L. Zepeda-Núñez, and H. Zhao, *Learning dominant wave directions for plane wave methods for high-frequency Helmholtz equations*, Res. Math. Sci., 4 (2017), 9.
[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, 2016.
[13] B. N. Jiang and L. A. Povinelli, *Least-squares finite element method for fluid dynamics*, Comput. Methods Appl. Mech. Engrg., 81 (1990), pp. 13–37.
[14] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, preprint, https://arxiv.org/abs/1412.6980, 2014.

[15] I. Livshits and A. Brandt, *Accuracy properties of the wave-ray multigrid algorithm for Helmholtz equations*, SIAM J. Sci. Comput., 28 (2006), pp. 1228–1251, https://doi.org/10.1137/040620461.
[16] T. Luo, Z. Ma, Z.-Q. J. Xu, and Y. Zhang, *Theory of the Frequency Principle for General Deep Neural Networks*, preprint, https://arxiv.org/abs/1906.09235, 2019.
[17] R. Maziar, 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. Comput. Phys., 378 (2019), pp. 686–707.
[18] M Rahaman, A. Baratin, D. Arpit, F. Draxler, M Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, *On the Spectral Bias of Neural Networks*, preprint, https://arxiv.org/abs/1806.08734, 2018.
[19] T. Xia, L. L. Meng, Q. S. Liu, H. H. Gan, and W. C. Chew, *A low-frequency stable broadband multilevel fast multipole algorithm using plane wave multipole hybridization*, IEEE Trans. Antennas Propagation, 66 (2018), pp. 6137–6145.
[20] Z.-Q. J. Xu, *Understanding Training and Generalization in Deep Learning by Fourier Analysis*, preprint, https://arxiv.org/abs/1808.04295, 2018.
[21] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma, *Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks*, preprint, https://arxiv.org/abs/1901.06523, 2019.