Article

Learning Adaptive Coarse Spaces of BDDC Algorithms for Stochastic Elliptic Problems with Oscillatory and High Contrast Coefficients

Eric Chung [1,*], Hyea-Hyun Kim [2], Ming-Fai Lam [1] and Lina Zhao [1]

[1] Department of Mathematics, The Chinese University of Hong Kong, Shatin, NT, Hong Kong, China; mflam@math.cuhk.edu.hk (M.-F.L.); lzhao@math.cuhk.edu.hk (L.Z.)
[2] Department of Applied Mathematics and Institute of Natural Sciences, Kyung Hee University, Yongin, Korea; hhkim@khu.ac.kr
* Correspondence: tschung@math.cuhk.edu.hk

Abstract: In this paper, we consider the balancing domain decomposition by constraints (BDDC) algorithm with adaptive coarse spaces for a class of stochastic elliptic problems. The key ingredient in the construction of the coarse space is the solutions of local spectral problems, which depend on the coefficient of the PDE. This poses a significant challenge for stochastic coefficients as it is computationally expensive to solve the local spectral problems for every realization of the coefficient. To tackle this computational burden, we propose a machine learning approach. Our method is based on the use of a deep neural network (DNN) to approximate the relation between the stochastic coefficients and the coarse spaces. For the input of the DNN, we apply the Karhunen–Loève expansion and use the first few dominant terms in the expansion. The output of the DNN is the resulting coarse space, which is then applied with the standard adaptive BDDC algorithm. We will present some numerical results with oscillatory and high contrast coefficients to show the efficiency and robustness of the proposed scheme.

Keywords: BDDC; stochastic partial differential equation; artificial neural network; coarse space; high contrast

Citation: Chung, E.; Kim, H.-H.; Lam, M.-F.; Zhao, L. Learning Adaptive Coarse Spaces of BDDC Algorithms for Stochastic Elliptic Problems with Oscillatory and High Contrast Coefficients. *Math. Comput. Appl.* **2021**, *26*, 44. https://doi.org/10.3390/mca26020044

Received: 30 March 2021
Accepted: 2 June 2021
Published: 6 June 2021

1. Introduction

Coefficient functions are often one of the main difficulties in modeling a real life problem. Due to the nature of the phenomena, these coefficients can be oscillatory and with high contrast features, all such conditions hinder the development of a fast and accurate numerical solution. Under some conditions, numerical methods like discontinuous Galerkin methods could give a robust scheme, as in [1,2]. However, more importantly, coefficients are not constant and they usually involve uncertainties and randomness. Therefore, a stochastic partial differential equation (SPDE) is considered for the modeling of these stochastic coefficient functions. There are several popular approaches for solving SPDE numerically, for example, the Monte Carlo simulation, stochastic Galerkin method, and the stochastic collocation method in [3–5]. All these methods require a high computational power in realistic simulations and cannot be generalized to similar situations. In the past few decades, machine learning techniques have been applied on many aspects, such as image classification, fluid dynamics, and solving differential equations in [6–14]. In this paper, instead of directly solving the SPDE by machine learning, we choose a moderate step that involves the advantage of machine learning and, also, the accuracy from the adaptive BDDC algorithm.

The combination of machine learning and domain decomposition methods has been discussed in [15–17] for different applications. A unified review on current approaches is presented in [10], which introduces three general approaches on this combination method:

the first one uses machine learning techniques to improve the convergence properties and the computational efficiency of domain decomposition methods. For the second type of approach, DNN is used as discretization methods and solvers to solve the differential equations directly. The last one makes use of the properties and ideas of domain decomposition methods to improve the performance of neural network.

Although close to the first approach descried, our idea in this paper is not actually the same. We first apply the Karhunen–Loève expansion on the stochastic coefficient as in [18–20] to capture the characteristic with finite terms. After this expansion, the stochastic coefficient function of SPDE is reduced to a finite expansion with desired accuracy and can be solved with a balancing domain decomposition by constraints (BDDC) algorithm, which can prevent the computational problem caused by the refinement of spatial mesh resolution and give an accurate solution rather than directly using machine learning technique without suitably designed conditions. After obtaining sufficient number of random solutions from the BDDC method, we can derive the statistics of the original SPDE solution. However, the cost for forming and solving generalized eigenvalue problems in the BDDC algorithm with good accuracy is considerable especially for high dimensional problems. We note that such eigenvalue problems are considered to enrich the coarse space adaptivity in the BDDC algorithm. Therefore, we introduce neural network with a user-defined number of layers and neurons to compensate the increase in computational cost with unexpected times of BDDC solutions.

The adaptive BDDC algorithm with enriched primal unknowns is considered in this paper as its ability on oscillatory and high contrast coefficients has been shown in [21,22]. Among different domain decomposition methods, the considered adaptive BDDC does not require a strong assumption on coefficient functions and the subdomain partitions to achieve a good performance like the standard BDDC algorithm [23,24]. It is because the additional coarse space basis functions computed by the dominant eigenfunctions are related to the ill-conditioning in the standard BDDC algorithm with highly varying coefficients. The introduction of these dominant eigenfunctions, thus, could greatly improve the robustness of numerical scheme for problems with rough and high contrast coefficients. Moreover, an estimate of condition number of this adaptive BDDC algorithm can be shown to be only controlled by a given tolerance without any extra assumptions on the coefficients or subdomain partitions. For simplicity, in the remaining of the paper, we will call the resulting new numerical scheme combining the machine learning technique and adaptive BDDC algorithm as learning adaptive BDDC algorithm.

Finally, to end this section, we state the main idea of our proposed scheme and the model problem. In this paper, we integrate an artificial neural network with learning abilities into a BDDC algorithm with adaptively enriched coarse spaces for efficient solutions of the neural network approximation of stochastic elliptic problems. Let $\mathcal{D}$ be a bounded domain in $\mathbb{R}^d$, $d = 2$, and Ω be a stochastic probability space:

$$\begin{aligned} -\nabla \cdot (\rho(\boldsymbol{x}, \omega) \nabla u) &= f, \quad && \text{in } \mathcal{D}, \\ u &= 0, \quad && \text{on } \partial\mathcal{D}, \end{aligned} \tag{1}$$

where $\boldsymbol{x} \in \mathcal{D}$, $\omega \in \Omega$, and $\rho(\boldsymbol{x}, \omega)$ is uniformly positive and is highly heterogeneous with very high contrast. Here, we only assume the two-dimensional spatial domain, however, the three-dimensional case is also applicable and is shown to be robust for deterministic elliptic problems in [21]. Moreover, as the model problem (1) can be extended to fluid flow problems, we seldom call the coefficient function $\rho(\boldsymbol{x}, \omega)$ as permeability function.

The rest of the paper is organized as follows. In Section 2.1, a brief formulation of the BDDC algorithm with adaptively enriched coarse problems is presented for two-dimensional elliptic problems. Then, the Karhunen–Loève expansion is introduced and the details of the artificial neural network used are clarified in Section 2.2. Two concrete examples with an explicit analytical expression of the Karhunen–Loève expansion, some network training, and testing parameters, are presented in the first half of Section 3 and the results of various numerical experiments are presented subsequently. Finally, a concluding remark is given in Section 4.

2. Materials and Methods

2.1. Adaptive BDDC Algorithm

The main feature of the adaptive BDDC is the local generalized eigenvalue problem defined on every subdomain interface, which introduces the adaptively enriched primal unknowns. In this subsection, we repeat the formulation presented by the first and second authors in [21] and give a brief overview of the adaptive BDDC algorithm. We refer to [21,22] for further details, where the analysis of the condition number estimation and the robustness of numerical scheme with oscillatory and high contrast coefficients can be found.

2.1.1. Local Linear System

We first introduce a discrete form of the model problem (1) in a deterministic fashion. Let V_h be the space of conforming linear finite element functions with respect to a given mesh on $\mathcal{D}$ with the mesh size $h > 0$ and with the zero value on $\partial\mathcal{D}$. We will then find the approximate solution $u \in V_h$ such that

$$a(u,v) = (f,v), \qquad \forall v \in V_h, \tag{2}$$

where

$$a(u,v) = \int_{\mathcal{D}} \rho(\boldsymbol{x})\nabla u \cdot \nabla v\, dx, \quad (f,v) = \int_{\mathcal{D}} f\, v\, dx. \tag{3}$$

We assume that the spatial domain $\mathcal{D}$ is partitioned into a set of N non-overlapping subdomains $\{\mathcal{D}_i\}$, $i = 1,2,\cdots,N$, so that $\mathcal{D} = \cup_{i=1}^{N}\mathcal{D}_i$. We note that the subdomain boundaries do not cut triangles equipped for V_h. We allow the coefficient $\rho(\boldsymbol{x})$ to have high contrast jumps and oscillations across subdomains and on subdomain interfaces. Let $a_i(u,v)$ be the bilinear form of the model elliptic problem (2) restricted to each subdomain $\mathcal{D}_i$ defined as

$$a_i(u,v) = \int_{\mathcal{D}_i} \rho(\boldsymbol{x})\nabla u \cdot \nabla v\, dx, \quad \forall u,v \in X_i,$$

where X_i is the restriction of V_h to $\mathcal{D}_i$.

In the BDDC algorithm, the original problem (2) is reduced to a subdomain interface problem and solved by an iterative method combined with a pre-conditioner. The interface problem can be obtained by solving a Dirichlet problem in each subdomain. After choosing dual and primal unknowns on the subdomain interface unknowns, the interface problem is then solved by utilizing local problems and one global coarse problem corresponding to the chosen sets of dual and primal unknowns, respectively. At each iteration, the residuals are multiplied by certain scaling factors to balance the errors across the subdomain interface regarding to the energy of each subdomain problem. The coarse problem aims to correct the global part of the error in each iteration and, thus, the choice of primal unknowns is important in obtaining a good performance as the number of subdomains increases. The basis for primal unknowns is obtained by the minimum energy extension for a given constraint at the location of primal unknowns and such a basis provides a robust coarse problem with a good energy estimate. We refer to [23–26] for general introductions to BDDC algorithm.

2.1.2. Notation and Preliminary Results

To facilitate our discussion, we first introduce some notation. Let $S^{(i)}$ be the Schur complement matrix obtained from the local stiffness matrix $A^{(i)}$ after eliminating unknowns interior to $\mathcal{D}_i$, where $A^{(i)}$ is defined by $a_i(u,v) = u^T A^{(i)} v$, for all $u,v \in X_i$. In the following, we will use the same symbol to represent a finite element function and its corresponding coefficient vector in order to simplify the notations.

Recall that X_i is the restriction of the finite element space V_h to each subdomain $\mathcal{D}_i$. Let W_i be the restriction of X_i to $\partial\mathcal{D}_i$. We then introduce the product spaces

$$X = \prod_{i=1}^{N} X_i, \quad W = \prod_{i=1}^{N} W_i,$$

where we remark that the functions in X and W are totally decoupled across the subdomain interfaces. In addition, we introduce partially coupled subspaces $\widetilde{X}$, $\widetilde{W}$, and fully coupled subspaces $\widehat{X}$, $\widehat{W}$, where some primal unknowns are strongly coupled for functions in $\widetilde{X}$ or $\widetilde{W}$, while the functions in $\widehat{X}$, $\widehat{W}$ are fully coupled across the subdomain interfaces.

Next, we present the basic description of the BDDC algorithm; see [23–26]. For simplicity, the two-dimensional case will be considered. After eliminating unknowns interior to each subdomain, the Schur complement matrices $S^{(i)}$ are obtained from $A^{(i)}$, and g_i is denoted as the corresponding right-hand side unknowns on the subdomain boundary. Summing up across the subdomains, the problem (2) is reduced in the BDDC algorithm, which is to find $\widehat{w} \in \widehat{W}$, such that

$$\sum_{i=1}^{N} R_i^T S^{(i)} R_i \widehat{w} = \sum_{i=1}^{N} R_i^T g_i, \tag{4}$$

where $R_i : \widehat{W} \to W_i$ is the restriction operator into $\partial\mathcal{D}_i$, and $g_i \in W_i$ depends on the source term f. Note, the solution $\widehat{w}$ to problem (4) is actually the restriction of u in (2) to the subdomain boundaries.

The BDDC pre-conditioner is built based on the partially coupled space $\widetilde{W}$. Let $\widetilde{R}_i : \widetilde{W} \to W_i$ be the restriction into $\partial\mathcal{D}_i$, and let $\widetilde{S}$ be the partially coupled matrix defined by

$$\widetilde{S} = \sum_{i=1}^{N} \widetilde{R}_i^T S^{(i)} \widetilde{R}_i.$$

For the space $\widetilde{W}$, we can express it as the product of the two spaces

$$\widetilde{W} = W_\Delta \times \widehat{W}_\Pi,$$

where $\widehat{W}_\Pi$ consists of vectors of the primal unknowns and W_Δ consists of vectors of dual unknowns, which are strongly coupled at the primal unknowns and decoupled at the remaining interface unknowns, respectively. We define $\widehat{R} : \widehat{W} \to \widetilde{W}$ such that

$$\widehat{R} = \begin{pmatrix} R_\Delta \\ R_\Pi \end{pmatrix},$$

where R_Δ is the mapping from $\widehat{W}$ to W_Δ and R_Π is the restriction from $\widehat{W}$ to $\widehat{W}_\Pi$. We note that R_Δ is obtained as

$$R_\Delta = \begin{pmatrix} R_\Delta^{(1)} \\ R_\Delta^{(2)} \\ \vdots \\ R_\Delta^{(N)} \end{pmatrix},$$

where $R_\Delta^{(i)}$ is the restriction from $\widehat{W}$ to $W_\Delta^{(i)}$ and $W_\Delta^{(i)}$ is the space of dual unknowns of $\mathcal{D}_i$.

The BDDC pre-conditioner is then given by

$$M_{BDDC}^{-1} = \widehat{R}^T \widetilde{D} \widetilde{S}^{-1} \widetilde{D}^T \widehat{R}, \tag{5}$$

where $\widetilde{D}$ is a scaling matrix of the form

$$\widetilde{D} = \sum_{i=1}^{N} \widetilde{R}_i^T D_i \widetilde{R}_i.$$

Here, the matrices D_i are defined for unknowns in W_i and they are introduced to make the pre-conditioner robust to the heterogeneity in $\rho(\boldsymbol{x})$ across the subdomain interface. In more detail, D_i consists of blocks $D_F^{(i)}$ and $D_V^{(i)}$, where F denotes an equivalence class shared by two subdomains, i.e., $\mathcal{D}_i$ and its neighboring subdomain $\mathcal{D}_j$, and V denotes the end points of F, respectively. We call such equivalence classes F and V as edge and vertex in two dimensions, respectively. In our BDDC algorithm, unknowns at subdomain vertices are included to the set of primal unknowns and adaptively selected primal constraints are later included to the set after a change of basis formulation. For a given edge F in two dimensions, the matrices $D_F^{(l)}$ and $D_V^{(l)}$ satisfy a partition of unity property, i.e., $D_F^{(i)} + D_F^{(j)} = I$ and $\sum_{l \in n(V)} D_V^{(l)} = 1$, where $n(V)$ denotes the set of subdomain indices sharing the vertex V. The matrices $D_F^{(l)}$ and $D_V^{(l)}$ are called scaling matrices. As mentioned earlier, the scaling matrices help to balance the residual error at each iteration with respect to the energy of subdomain problems sharing the interface. For the case when $\rho(\boldsymbol{x})$ is identical across the interface F, $D_F^{(i)}$ and $D_F^{(j)}$ are chosen simply as multiplicity scalings, i.e., $1/2$, but for a general case when $\rho(\boldsymbol{x})$ has discontinuities, different choice of scalings, such as ρ-scalings or deluxe scalings, can be more effective. The scaling matrices $D_V^{(l)}$ can be chosen using similar ideas. We refer to [27] and references therein for scaling matrices.

We note that by using the definitions of $\widehat{R}$ and $\widetilde{S}$, the matrix in the left hand side of (4) can be written as

$$\sum_{i=1}^{N} R_i^T S^{(i)} R_i = \widehat{R}^T \widetilde{S} \widehat{R}.$$

In the BDDC algorithm, the system in (4) is solved by an iterative method with the preconditioner (5). Thus, its performance is analyzed by estimating the condition number of

$$M_{BDDC}^{-1} \widehat{R}^T \widetilde{S} \widehat{R} = \widehat{R}^T \widetilde{D} \widetilde{S}^{-1} \widetilde{D}^T \widehat{R} \widehat{R}^T \widetilde{S} \widehat{R}. \tag{6}$$

2.1.3. Generalized Eigenvalue Problems

After the preliminaries, we are ready to state the generalized eigenvalue problems which introduce the adaptive enrichment of coarse components. For an equivalence class F shared by two subdomains $\mathcal{D}_i$ and $\mathcal{D}_j$, the following generalized eigenvalue problem is proposed in [28]:

$$\left((D_F^{(j)})^T S_F^{(i)} D_F^{(j)} + (D_F^{(i)})^T S_F^{(j)} D_F^{(i)} \right) v = \lambda \left(\widetilde{S}_F^{(i)} : \widetilde{S}_F^{(j)} \right) v, \tag{7}$$

where $S_F^{(i)}$ and $D_F^{(i)}$ are the block matrices of $S^{(i)}$ and D_i corresponding to unknowns interior to F, respectively. The matrix $\widetilde{S}_F^{(i)}$ are the Schur complement of $S^{(i)}$ obtained after eliminating unknowns except those interior to F. In addition, for symmetric and semi-positive definite matrices A and B, their parallel sum $A : B$ is defined by, see [29],

$$A : B = B(A+B)^+ A, \tag{8}$$

where $(A+B)^+$ is a pseudo inverse of $A+B$. We note that the problem in (7) is identical to that considered in [30] when $D_F^{(i)}$ are chosen as the deluxe scalings, i.e.,

$$D_F^{(i)} = (S_F^{(i)} + S_F^{(j)})^{-1} S_F^{(i)}.$$

We note that a similar generalized eigenvalue problem is considered and extended to three-dimensional problems in the first second authors' work [21].

2.2. Learning Adaptive BDDC Algorithm

In addition to the BDDC algorithm, we perform the Karhunen–Loève (KL) expansion to decompose the stochastic permeability $\rho(\boldsymbol{x}, \omega)$ for the preparation of the learning adaptive BDDC algorithm. As shown by Theorem 2.7, Proposition 2.18 in [31] and the numerical examples in [20], a small number of terms in the KL expansion can approximate the stochastic permeability with a reasonable accuracy. Thus, corresponding computational cost can also be reduced efficiently. Once we obtain the KL series expansion, we can easily produce a certain amount of sample data as a training set for the artificial neural network. After the training process, the neural network learns the main characteristics of the dominant eigenvectors from the adaptive BDDC algorithm, and neural network approximation on coarse spaces can be obtained. In our proposed algorithm, we still need some resulting samples from the BDDC algorithm to train the neural network, which is a process of a relatively high computational cost. However, after this setup, we can apply this network to obtain an approximate solution directly without the use of other stochastic sampling methods, such as the Monte Carlo simulation, which involves even higher computational cost for the generation of large number of random samples or realizations.

In this subsection, we discuss the proposed algorithm in details. Moreover, our resulting neural network can be also applied on problems with similar stochastic properties. It thus saves the cost of training again and we can readily use the trained neural network for prediction. Some testing samples will be generated to test the network performance and to show its generalization capacity in the later section.

2.2.1. Karhunen–Loève Expansion

To start our learning adaptive algorithm, we first apply the KL expansion on the stochastic field $\rho(\boldsymbol{x}, \omega)$. To ensure the positivity of stochastic permeability field, a logarithmic transformation is considered as $K(\boldsymbol{x}, \omega) = \log(\rho(\boldsymbol{x}, \omega))$. Let $C_K(\boldsymbol{x}, \hat{\boldsymbol{x}})$ be the covariance function of $K(\boldsymbol{x}, \omega)$ at two locations $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$. Since the covariance function $C_K(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is symmetric and positive definite, it can be decomposed into:

$$C_K(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=1}^{\infty} \lambda_i f_i(\boldsymbol{x}) f_i(\hat{\boldsymbol{x}}), \tag{9}$$

where λ_i and f_i are eigenvalues and eigenfunctions computed from the following Fredholm integral equation:

$$\int_{\mathcal{D}} C_K(\boldsymbol{x}, \hat{\boldsymbol{x}}) f(\boldsymbol{x}) \, d\boldsymbol{x} = \lambda f(\hat{\boldsymbol{x}}), \tag{10}$$

where $\{f_i(\boldsymbol{x})\}$ are orthogonal and deterministic functions. It is noted that for some specially chosen covariance functions, we can find the (λ, f) eigenpair analytically. We will list some examples in the section of numerical results. After solving the Fredholm integral equation, we are ready to express the KL expansion of the log permeability:

$$K(\boldsymbol{x}, \omega) = E[K](\boldsymbol{x}) + \sum_{i=1}^{\infty} \sqrt{\lambda_i} \xi_i f_i(\boldsymbol{x}). \tag{11}$$

Here $E[K](\boldsymbol{x})$ is the expected value of $K(\boldsymbol{x}, \omega)$ and ξ_i are independent and identically distributed Gaussian random variables with mean 0 and variance 1. As mentioned above, a few terms in KL expansion can give a reasonably accurate approximation, which can be explained now as we can just pick the dominant eigenfunctions in (11) and the other eigenvalues will decay rapidly.

Therefore, KL expansion is an efficient method to represent the stochastic coefficient and favors the subsequent process. Although analytical eigenfunctions and eigenvalues cannot always be found, there are various numerical methods to compute the KL coefficients in [31,32]. After the generation of some sets of $\{\xi_i\}$, which are the only stochastic variables in the permeability after KL expansion, as the input of neural network, we plug back KL expanded permeability coefficients into the adaptive BDDC algorithm, and the resulting dominant eigenfunctions in the coarse spaces will be the target output of the our designated neural network.

2.2.2. Neural Network

There are many types of neural networks for different usages. Since in our learning adaptive BDDC algorithm the desired neural network is to capture numeric feature from the data as a supervised learning, a fully connected feed-forward neural network is chosen. An illustration of the neural network structure is shown in Figure 1. In a general feed-forward neural network, there are L hidden layers between the input and output layer, and the arrows from layers to layers indicate that all the neurons in the previous layer are used to produce every neuron in the next layer. Computational cost increases with L, but significant improvements in predictions of our tests are not always resulted, $L = 1$ is thus chosen for most of the numerical experiments.

We denote the number of neurons in the input layer to be R, which is the number of terms in the truncated KL expansion. The inputs are actually the Gaussian random variables $\{\xi_i\}$ in one realization. The output of the network is a column vector that consists of all dominant eigenvectors in the coarse space. The number of neurons in the output layer is denoted as O, which is the sum of length of all resulting eigenvectors obtained from the BDDC algorithm. This value O depends on many factors, such as the geometry of the spatial domain, the structure of the grid, and parameters of the BDDC algorithm.

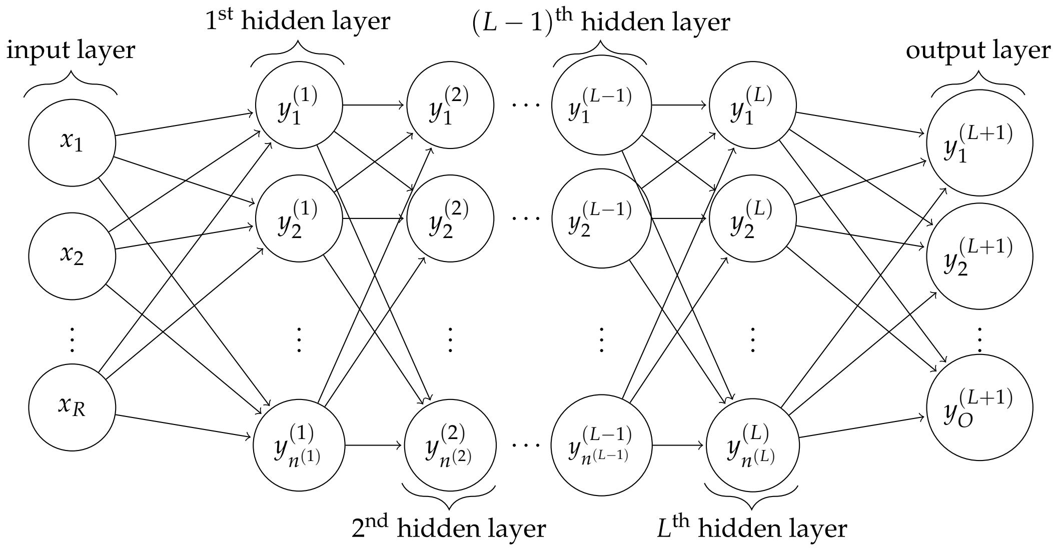


Figure 1. An illustration of neural network structure of a $(L+1)$-layer perceptron with R input neurons and O output neurons. The lth hidden layer contains $n^{(l)}$ hidden neurons.

In the training, we use the scaled conjugate gradient algorithm in [33] to do the task of minimization. Therefore, unlike the gradient descent method, we do not need to determine the value of learning rate at each step. However, the initial choice for the scaled conjugate gradient algorithm may still affect the performance of training. Before training, we have to generate training samples and decide some conditions on stopping criteria. When the minimization process is completed for all the training samples, we are said to finish one epoch. There are two stopping criteria that are usually considered. They are the tolerance of the cost function gradient and the maximum number of epochs trained. When either of these conditions are satisfied, that is, the cost function gradient is below the specified tolerance or the number of epochs reaches the maximum number, the training is stopped and the network will be tested for performance on a testing set.

In both training and testing results, in order to obtain an error measure less affected by scales of data set, a normalized root mean squared error (NRMSE) is used as an error of reference to see how well are the network estimation and prediction:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M} \frac{1}{O}||y(\boldsymbol{\xi}^{(i)}) - f_{NN}(\boldsymbol{\xi}^{(i)})||_2^2}{M}},$$
$$NRMSE = \frac{RMSE}{\max_{i=1,2,\ldots,M} ||y(\boldsymbol{\xi}^{(i)}) - f_{NN}(\boldsymbol{\xi}^{(i)})||_\infty},$$

where $\boldsymbol{\xi}^{(i)} \sim N(0,1) \in \mathbb{R}^R$ is a column vector of random variables that follow the standard normal distribution in the i-th realization, $y(\cdot)$ is the dominant eigenvectors obtained from the adaptive BDDC algorithm when the KL expanded stochastic permeability function is used, and $f_{NN}(\cdot)$ is the regression function describing the neural network (NN). Here, O is the dimension of network output, and M is the number of samples. For simplicity, the considered NRMSE can be understood as the root mean squared error divided by the maximum element of the absolute difference of all the eigenvectors.

Overall, the proposed scheme can be concluded as follows:

- Step 1: Perform KL expansion on the logarithmic stochastic permeability function $K(\boldsymbol{x},\omega)$;
- Step 2: Generate M realizations of $\{\boldsymbol{\xi}^{(i)}\}$ and obtain the corresponding BDDC dominant eigenvectors $\{y(\boldsymbol{\xi}^{(i)})\}$, which are the training data for the neural network;
- Step 3: Define training conditions and train the neural network;
- Step 4: Examine the network performance whether the NRMSE is satisfied, otherwise, go back to Step 3 and change training conditions.

3. Results and Discussion

In this section, supporting numerical results are presented to show the performance of our proposed learning adaptive BDDC algorithm. We will consider various choices of permeability coefficients $\rho(\boldsymbol{x},\omega)$ with two major stochastic behaviors. As the concern of our numerical tests are on the learning algorithm, we fix all the parameters that are not related. In most of the experiments, $\mathcal{D} \in \mathbb{R}^2$ is chosen to be a unit square spatial domain, and it is partitioned into 16 uniform square subdomains with $H = 4^{-1}$ as the coarse grid size. Each subdomain is then further divided into uniform grids with a fine grid size $h = 32^{-1}$. We will state clearly if other grid sizes are used. In the following, we will first describe the considered stochastic coefficients and training conditions.

3.1. Choices of Stochastic Coefficients

From the KL expansion expression (11), we know the eigenvalues and eigenfunctions computed from the Fredholm integral equation are related to the covariance function of $K(\boldsymbol{x},\omega)$, which can be treated as the starting point of KL expansion. Throughout the experiments, two specially chosen covariance functions whose eigenvalues and eigenfunctions in the Fredholm integral equation can be computed analytically are considered with different expected values. For $\boldsymbol{x} = (x_1,x_2)^T$ and $\hat{\boldsymbol{x}} = (\hat{x}_1,\hat{x}_2)^T$, these two covariance functions are

1. Brownian sheet covariance function:

$$C_K(\boldsymbol{x},\hat{\boldsymbol{x}}) = \min(x_1,\hat{x}_1)\min(x_2,\hat{x}_2),$$

where the corresponding eigenvalues and eigenfunctions are

$$\lambda_k = \frac{16}{((2i-1)^2\pi^2)((2j-1)^2\pi^2)},$$
$$f_k(\boldsymbol{x}) = 2\sin\Big((i-\frac{1}{2})\pi x_1\Big)\sin\Big((j-\frac{1}{2})\pi x_2\Big).$$

2. Exponential covariance function:

$$C_K(\boldsymbol{x},\hat{\boldsymbol{x}}) = \sigma_K^2 \exp\Big(-\frac{|x_1-\hat{x}_1|}{\eta_1} - \frac{|x_2-\hat{x}_2|}{\eta_2}\Big),$$

where σ_K^2 and η_i are the variance and correlation length in the x_i direction of the process, respectively. The corresponding eigenvalues and eigenfunctions are

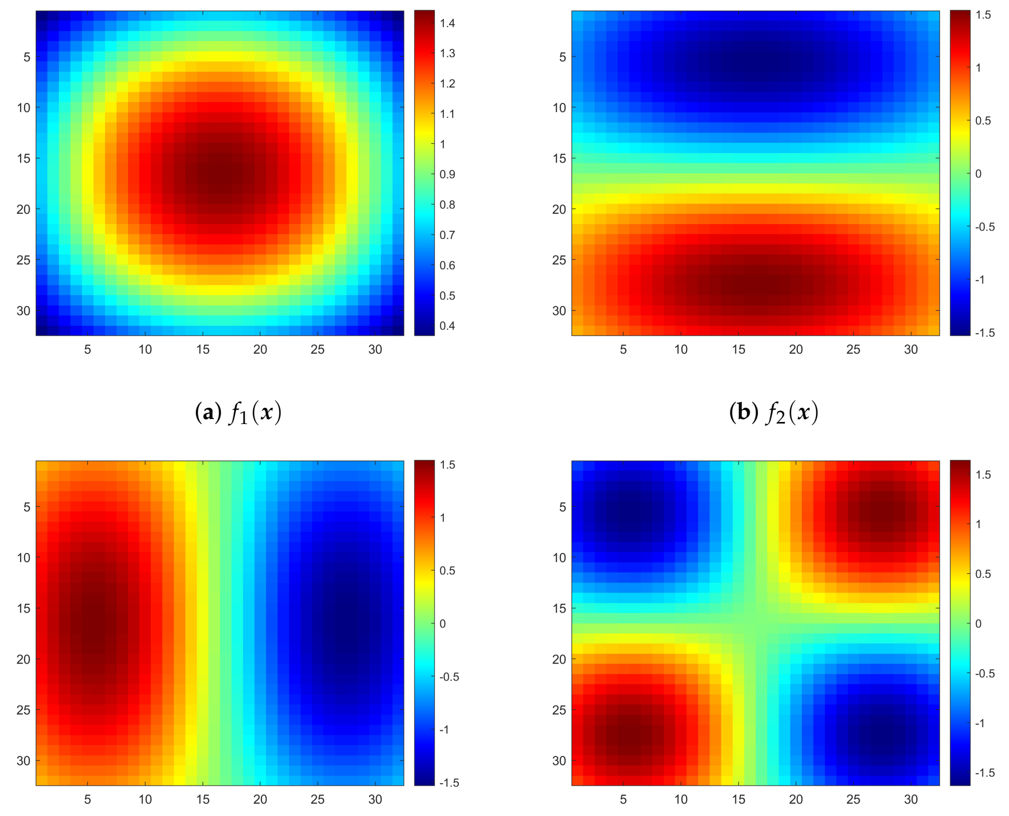
$$\lambda_k = \frac{4\eta_1\eta_2\sigma_K^2}{(r_{1,i}^2\eta_1^2+1)(r_{2,j}^2\eta_2^2+1)},$$
$$f_k(\boldsymbol{x}) = \frac{r_{1,i}\eta_1\cos(r_{1,i}x_1)+\sin(r_{1,i}x_1)}{\sqrt{(r_{1,i}^2\eta_1^2+1)/2+\eta_1}}\,\frac{r_{2,j}\eta_2\cos(r_{2,j}x_2)+\sin(r_{2,j}x_2)}{\sqrt{(r_{2,j}^2\eta_2^2+1)/2+\eta_2}},$$

where $r_{1,i}$ is the i-th positive root of the characteristic equation

$$(r_{1,i}^2\eta_1^2-1)\sin(r_{1,i}) = 2\eta_1 r_{1,i}\cos(r_{1,i})$$

in the x_1 direction. The same for $r_{2,j}$, which is in the j-th positive root of the characteristic equation in the x_2 direction. After arranging the roots $r_{1,i}$ and $r_{2,j}$ in ascending order, we can immediately obtain a monotonically decreasing series of λ_k, which favors our selection of dominant eigenfunctions in the truncated KL expansion with R terms.

For clarification, the positive indices i and j are mapped to index k so that λ_k is monotonically decreasing. It is noted these eigenvalues and eigenfunctions are just the product of solutions of the one-dimensional process in the Fredholm integral equation, since the covariance functions are separable. First four terms of the eigenfunctions for the second covariance function with $(\sigma_K^2,\eta_1,\eta_2) = (1, 0.25, 0.25)$ are illustrated in Figure 2.



(**a**) $f_1(\boldsymbol{x})$ (**b**) $f_2(\boldsymbol{x})$

(**c**) $f_3(\boldsymbol{x})$ (**d**) $f_4(\boldsymbol{x})$

Figure 2. Illustration of first four eigenfunctions $f_k(\boldsymbol{x})$ for the second covariance function with $(\sigma_K^2, \eta_1, \eta_2) = (1, 0.25, 0.25)$.

3.2. Adaptive BDDC Parameters and Training Conditions

In the setting of BDDC parameters, we fix the number of dominant eigenvectors on every coarse edge to be 2 and deluxe scaling is always considered. We remark that we have fixed the number of coarse basis functions on each edge. In general, one should vary the number of coarse basis functions according to the local heterogeneities. If the number of coarse basis function is too large, then the coarse problem is more expensive to solve. On the other hand, if the number of coarse basis function is too small, then the number of iteration will increase. It is worth note that, the output layer of neural network is composed by the total dominant eigenvectors on all the coarse edges, so the output layer size O is determined by the coarse grid size H, fine grid size h, and also the number of dominant eigenvectors on every coarse edge. When solve the system by the conjugate gradient method, the iteration is stopped when the relative residual is below 10^{-10}.

In the training, there are different sets of hyperparameter that could affect the training performance of the neural network. Unless specified, the following training conditions are considered:

- Number of truncated terms in KL expansion: $R = 4$;
- Number of hidden layers: $L = 1$;
- Number of neurons in the hidden layer: 10;
- Number of neurons in the output layer: $O = 336$;
- Activation function in hidden layer: hyperbolic tangent function;
- Activation function in output layer: linear function;
- Stopping criteria:
  - Minimum value of the cost function gradient: 10^{-6}; or
  - Maximum number of training epochs: 1,000,000;

- Sample size of training set: M =10,000;
- Sample size of testing set: $M = 500$.

After we obtain an accurate neural network from the proposed algorithm, besides the NRMSE of testing set samples, we also present some characteristics of the pre-conditioner based on the approximate eigenvectors from the learning adaptive BDDC algorithm, which include the number of iterations required, minimum and maximum eigenvalues of the pre-conditioner. To show the performance of pre-conditioning on our predicted dominant eigenvectors, we will use the infinity norm to measure the largest difference and also the following symmetric mean absolute percentage error (sMAPE), proposed in [34], to show a relative error type measure with an intuitive range between 0% and 100%:

$$sMAPE = \frac{1}{M}\sum_{i=1}^{M}\frac{|q_i - \widehat{q}_i|}{|q_i| + |\widehat{q}_i|},$$

where q_i are some target quantities from pre-conditioning and $\widehat{q}_i$ is the same type of quantity as q_i from pre-conditioning using our predicted eigenvectors. In the following subsections, all the computation and results were obtained using MATLAB R2019a with Intel Xeon Gold 6130 CPU and GeForce GTX 1080 Ti GPU in parallel.

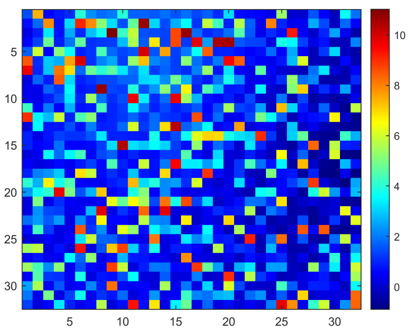
3.3. Brownian Sheet Covariance Function

The first set of numerical tests bases on the KL expansion of Brownian sheet covariance function with the following expected functions $E[K](\boldsymbol{x})$:

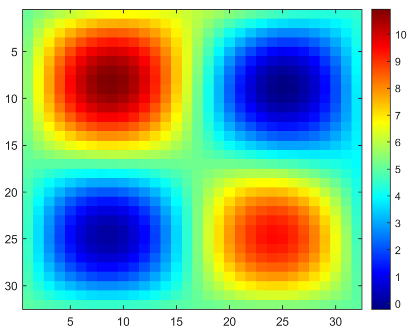
($\mathcal{A}$) 10^s

($\mathcal{B}$) $5\sin(2\pi x_1)\sin(2\pi x_2) + 5$.

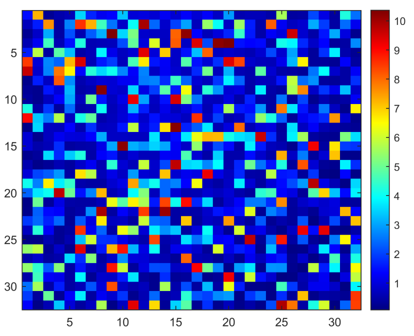
Expected function $\mathcal{A}$ is a random coefficient as $s \in [-1, 1]$ is randomly chosen for each fine grid element, and expected function $\mathcal{B}$ is a smooth trigonometric function with values between $[0, 10]$. The common property between these two choices is that the resulting permeability function $\rho(\boldsymbol{x}, \omega)$ is highly oscillatory with magnitude order about 10^{-2} to 10^5, which are good candidates to test our method on oscillatory and high contrast coefficients. Here, we show the appearance of mean functions $\mathcal{A}$ and $\mathcal{B}$ and the corresponding permeability function $\mathcal{K}(\boldsymbol{x}, \omega)$ in Figure 3. We can observe that from the first row of Figure 3 to the second row, that is, from the appearances of expected function $E[K](\boldsymbol{x})$ to the logarithmic permeability coefficient $K(\boldsymbol{x}, \omega)$, there are no significant differences except only a minor change on the values on each fine grid element. Nevertheless, these small stochastic changes cause a high contrast $\rho(\boldsymbol{x}, \omega)$ after taking exponential function as in the third row of Figure 3.



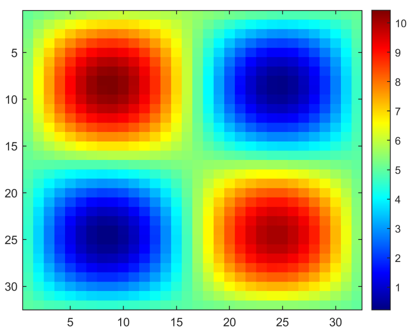
(**a**) Expected function $\mathcal{A}$



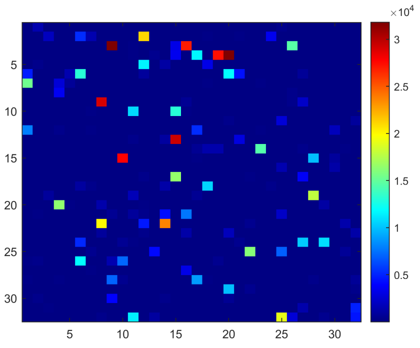
(**b**) Expected function $\mathcal{B}$



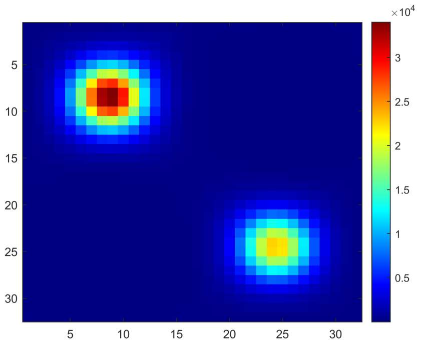
(**c**) A realization of $K(\boldsymbol{x},\omega)$ with expected function $\mathcal{A}$



(**d**) A realization of $K(\boldsymbol{x},\omega)$ with expected function $\mathcal{B}$



(**e**) A realization of $\rho(\boldsymbol{x},\omega)$ with expected function $\mathcal{A}$



(**f**) A realization of $\rho(\boldsymbol{x},\omega)$ with expected function $\mathcal{B}$

Figure 3. Realizations of permeability coefficient when expected functions $\mathcal{A}$ and $\mathcal{B}$ are used.

In Table 1, the training results of these two expected functions $\mathcal{A}$ and $\mathcal{B}$ under the training conditions mentioned are presented. Although the random initial choice of conjugate gradient algorithm for minimization may have influences, in general, the neural network of expected function $\mathcal{A}$ usually needs more epochs until stopping criterion is reached and, thus, the training time is also longer. However, this increase in training epochs does not bring a better training NRMSE when compared with expected function $\mathcal{B}$. We can see the same phenomenon when the testing set is considered. The main reason is due to the smoothness of expected function $\mathcal{B}$, the neural network can capture its characteristics easier than a random coefficient.

Table 1. Training records when expected functions $\mathcal{A}$ and $\mathcal{B}$ are considered.

Case	Epochs Trained	Training NRMSE	Training Time	Preparation Time for 1 Sample
$\mathcal{A}$	2.29×10^5	1.97×10^{-2}	1.98×10^3 s	1.53 s
$\mathcal{B}$	5.39×10^4	3.77×10^{-3}	6.33×10^2 s	1.54 s

In the testing results, we consider another set of data containing $M = 500$ samples as a testing set. After obtaining the predicted eigenvectors from neural network, we plug it into the BDDC pre-conditioned solver and obtain an estimated pre-conditioner. For the estimated pre-conditioner, we report several properties such as the number of iteration needed for the iterative solver, the maximum and minimum eigenvalues of the pre-conditioned system. Therefore, to better show the performance of network, besides the testing error NRMSE, we also use the sMAPE and l_∞ norm of difference of the iteration numbers, the maximum and minimum eigenvalues of the estimated pre-conditioner and the target pre-conditioner. The comparison results are listed in Table 2 below.

Table 2. Testing records when expected functions $\mathcal{A}$ and $\mathcal{B}$ are considered.

Case	Testing NRMSE	sMAPE (l_∞ Error) in		
		Iteration Number	λ_{min}	λ_{max}
$\mathcal{A}$	6.58×10^{-2}	6.94×10^{-2} (2)	2.58×10^{-7} (4.02×10^{-6})	4.45×10^{-3} (3.58×10^{-2})
$\mathcal{B}$	6.73×10^{-3}	6.93×10^{-2} (1)	3.19×10^{-5} (1.50×10^{-4})	2.01×10^{-3} (4.94×10^{-2})

It is clear that the testing NRMSE of expected function $\mathcal{B}$ is also much smaller than expected function $\mathcal{A}$. One possible reason is because the function used is smooth, and the neural network can better capture the property of the resulting stochastic permeability function. We can see that values of the testing NRMSE of both expected function $\mathcal{A}$ and expected function $\mathcal{B}$ are just a bit larger than the training NRMSE. This suggests that the neural network considered is capable to give a good prediction on dominant eigenvectors in the coarse space even when the magnitude of coefficient function value changes dramatically across each fine grid element. Nevertheless, even though a more accurate set of eigenvectors can be obtained for the expected function $\mathcal{B}$, but its performance after the BDDC pre-conditioning is not always better than that of expected function $\mathcal{A}$. Obviously, l_∞ errors in the aspects of λ_{max} and λ_{min} are larger for the expected function $\mathcal{B}$.

Detailed comparisons can be found in Figure 4, where the area of overlapping represents how close the specified quantity of the estimated pre-conditioner is compared to the target pre-conditioner. In the first row of figures, we can see the number of iteration required for the predicted eigenvectors is usually more than the target iteration number, however, the differences in the iteration number is not highly related to the difference in minimum and maximum eigenvalues. We can see more examples to verify it in this and later results. The bin width of the histogram of minimum eigenvalues for expected function $\mathcal{A}$ is about 10^{-7}, which is in coherence with the l_∞ error, but we eliminate this small magnitude effect and the sMAPE is considered. The high ratio of overlapping area for expected function $\mathcal{A}$ confirms the usage of sMAPE as a good measure, as the resulting sMAPE 2.58×10^{-7} of expected function $\mathcal{A}$ is much smaller than 3.19×10^{-5} of expected function $\mathcal{B}$.

For the last column of errors in Table 2, the l_∞ norm of maximum eigenvalues is larger for expected function $\mathcal{B}$. Nevertheless, it does not mean that the neural network of expected function $\mathcal{B}$ performs worse. From its histogram, we can see an extremely concentrated and overlapping area at the bin around 1.04, moreover, there are some outliers for the prediction results, which is one of the source for a larger l_∞ norm. Therefore, the sMAPE can well represent the performance of preconditioning on the estimated results, and we can conclude that both neural networks show a good results on oscillatory and high contrast coefficients, which can be represented by NRMSE and sMAPE. In Section 3.4, besides an artificial coefficient, we will consider one that is closer to a real life case.

Although the l_∞ norm seems not describe the characteristics of performance in a full picture, it is still a good and intuitive measure, for example, in the difference of iteration number, we can immediately realize how large will be the worst case.

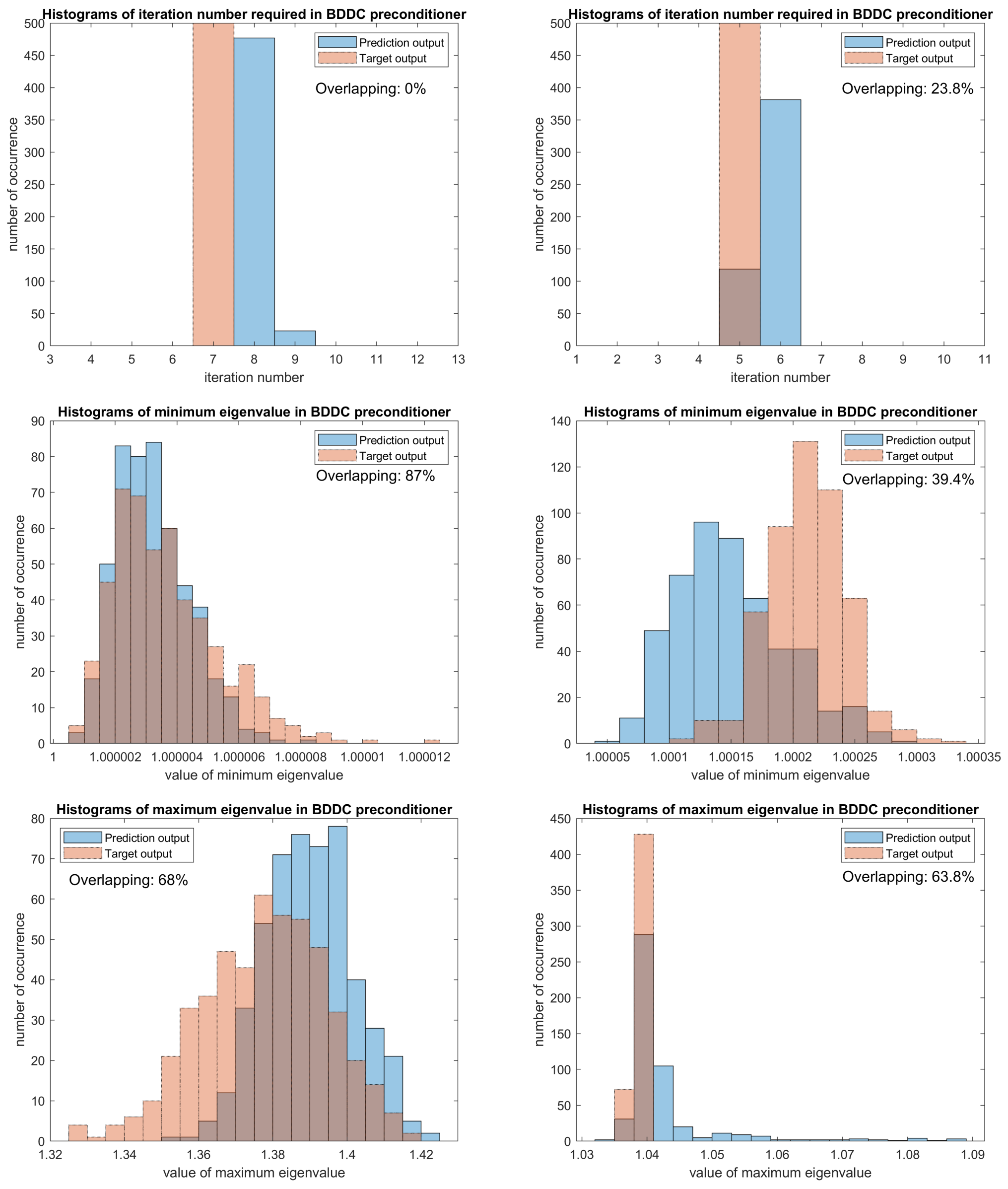


Figure 4. Comparisons in the performance of the BDDC preconditioner when expected functions $\mathcal{A}$ (**left column**) and $\mathcal{B}$ (**right column**) are used.

Further Experiments on Different Training Conditions

To emphasize the changes of using different training conditions, we display the results of increasing the truncated terms R in KL expansion and using less training samples. The expected functions and stochastic property considered are the same as above to show the effect of changing conditions only. For clarification, there are two experiments each with only one condition is modified and other conditions are the same as stated in Section 3.2. We state the new conditions as the follows:

- Modified experiment 1: $R = 9$;
- Modified experiment 2: Training sample is 5000.

When R is increased, we can see in Table 3, the training time increases as the total number of parameters increases with the input layer. In Table 4, the testing results of expected function $\mathcal{B}$ clearly have better performance than in Table 2, which suggests that a larger R could bring a smaller NRMSE.

Table 3. Training records when modified experiment 1 is considered.

Case	Epochs Trained	Training NRMSE	Training Time	Preparation Time for 1 Sample
$\mathcal{A}$	2.17×10^5	2.74×10^{-2}	2.47×10^3 s	1.77 s
$\mathcal{B}$	1.15×10^5	4.02×10^{-3}	1.19×10^3 s	1.75 s

Table 4. Testing records when modified experiment 1 is considered.

Case	Testing NRMSE	sMAPE (l_∞ Error) in		
		Iteration Number	λ_{min}	λ_{max}
$\mathcal{A}$	6.97×10^{-2}	5.85×10^{-2} (1)	2.15×10^{-6} (1.35×10^{-5})	2.29×10^{-2} (9.05×10^{-2})
$\mathcal{B}$	5.40×10^{-3}	6.73×10^{-2} (1)	2.99×10^{-5} (1.76×10^{-4})	1.96×10^{-3} (5.64×10^{-2})

The training and testing results of modified experiment 2 are shown in Tables 5 and 6, respectively. Obviously, the testing results in Table 6 are worse than when 10,000 training samples are used. Moreover, as the training time does not have a distinct reduction, and with the aid of parallel computation, more training samples can be easily obtained. Therefore, using less samples is not always a good choice, we should use appropriate number of samples.

Table 5. Training records when modified experiment 2 is considered.

Case	Epochs Trained	Training NRMSE	Training Time	Preparation Time for 1 Sample
$\mathcal{A}$	9.12×10^4	3.34×10^{-2}	8.94×10^2 s	1.53 s
$\mathcal{B}$	1.01×10^5	4.91×10^{-3}	1.12×10^3 s	1.54 s

Table 6. Testing records when modified experiment 2 is considered.

Case	Testing NRMSE	sMAPE (l_∞ Error) in		
		Iteration Number	λ_{min}	λ_{max}
$\mathcal{A}$	7.82×10^{-2}	6.68×10^{-2} (2)	7.15×10^{-7} (6.59×10^{-6})	3.82×10^{-3} (1.57×10^{-1})
$\mathcal{B}$	8.43×10^{-3}	6.91×10^{-2} (1)	2.73×10^{-5} (1.64×10^{-4})	2.39×10^{-3} (6.61×10^{-2})

3.4. Exponential Covariance Function

To test our method on realistic highly varying coefficients, we consider expected functions that come from the second model of the 10th SPE Comparative Solution Project (SPE10), with the KL expanded exponential covariance function as the stochastic source. For clarification, in this set of experiment, we first use the modified permeability of Layer 35 in the x-direction of SPE10 data as the expected function $E[K](x)$ with $(\sigma_K^2, \eta_1, \eta_2) = (1, 0.25, 0.25)$ in the exponential covariance function $C_K(x, \hat{x})$. We then train the neural network, and the resulting neural network is used to test its generalization capacity on the following different situations:

1. Different stochastic behavior:
    - All remain unchanged except $(\eta_1, \eta_2) = (0.2, 0.125)$ or $(\eta_1, \eta_2) = (1, 1)$;
2. Different mean permeability function:

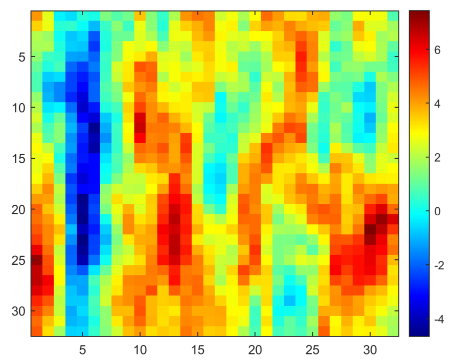
- The expected function is changed to the Layers 1, 4, 34 in the x-direction of SPE10 data, but the stochastic parameters are unchanged, i.e., $(\sigma_K^2, \eta_1, \eta_2) = (1, 0.25, 0.25)$.

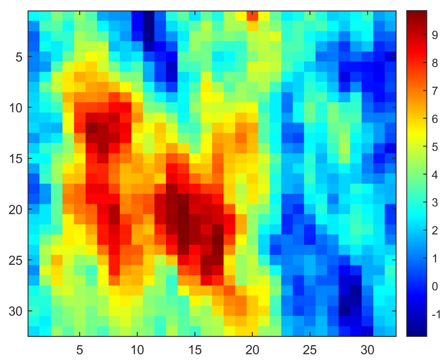
It is of note that, all the permeability fields used are modified into our computational domain and shown in Figure 5 for a clear comparison. In the Layer 35 permeability field, four sharp properties can be observed. There are one blue strip on the very left side, and a reversed but narrower red strip is next to it. In the top right corner, a low permeability area is shown and just below it, a small high permeability area is located in the bottom right corner. So a similar but with slightly different properties of Layer 34, and two Layers 1, 4 with more distinct differences are chosen to test the generalization capacity of our neural network.

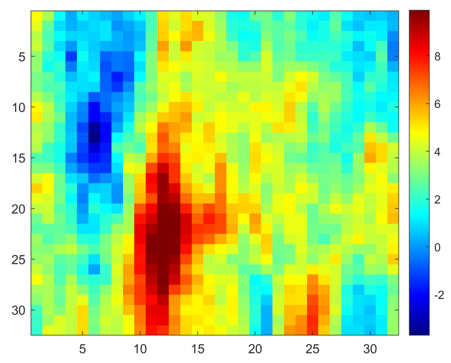
Before showing the training and testing records of neural network, we further present the logarithmic permeability $K(\boldsymbol{x}, \omega)$, and permeability $\rho(\boldsymbol{x}, \omega)$ when $(\sigma_K^2, \eta_1, \eta_2) = (1, 0.25, 0.25)$, $(1, 0.2, 0.125)$ and $(1, 1, 1)$ in Figure 6. The sets of $\{\xi_i\}$ used in these three rows of realizations are the same. We can thus see the change in parameters does cause the stochastic behavior to change, too. We present the training and testing records of neural network as below in Tables 7 and 8, where we specify the expected functions considered in the first column.

Table 7. Training record when Layer 35 permeability is considered.

Case	Epochs Trained	Training NRMSE	Training Time	Preparation Time for 1 Sample
Layer 35	2.62×10^5	1.52×10^{-2}	1.07×10^3 s	12.34 s


(**a**) Mean permeability field of Layer 1


(**b**) Mean permeability field of Layer 4


(**c**) Mean permeability field of Layer 34

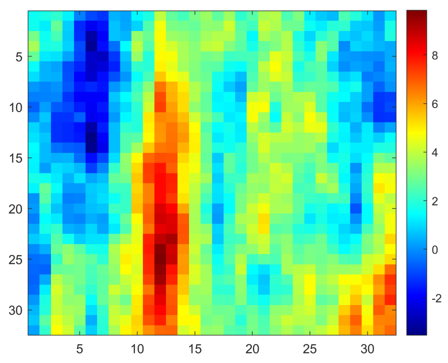
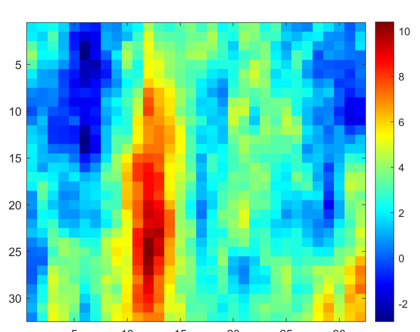
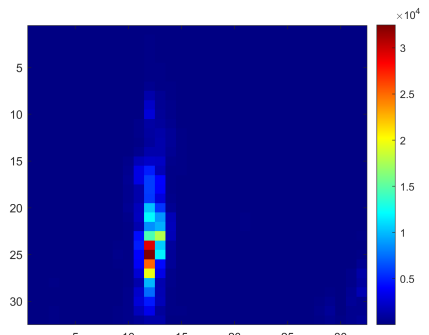

(**d**) Mean permeability field of Layer 35

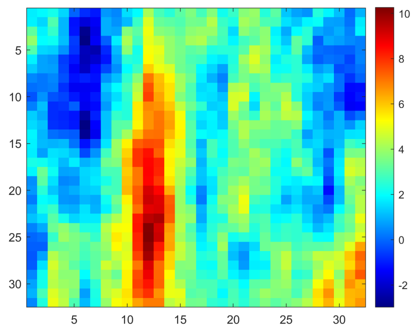
Figure 5. Mean permeability fields of different layers in the x-direction.



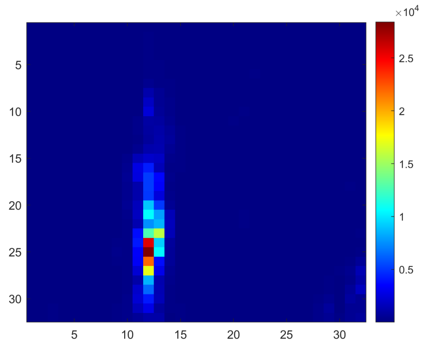
(**a**) Log permeability when $(\sigma_K^2, \eta_1, \eta_2) = (1, 0.25, 0.25)$



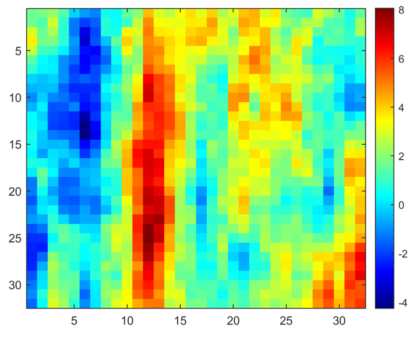
(**b**) Permeability $\rho(\boldsymbol{x}, \omega)$ when $(\sigma_K^2, \eta_1, \eta_2) = (1, 0.25, 0.25)$



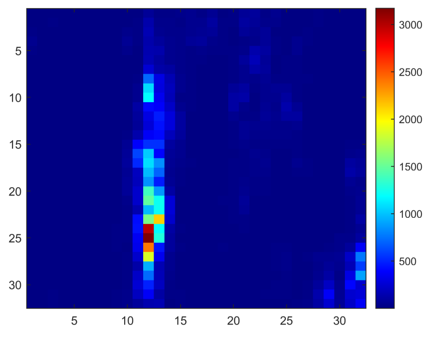
(**c**) Log permeability when $(\sigma_K^2, \eta_1, \eta_2) = (1, 0.2, 0.125)$



(**d**) Permeability $\rho(\boldsymbol{x}, \omega)$ when $(\sigma_K^2, \eta_1, \eta_2) = (1, 0.2, 0.125)$



(**e**) Log permeability when $(\sigma_K^2, \eta_1, \eta_2) = (1, 1, 1)$



(**f**) Permeability $\rho(\boldsymbol{x}, \omega)$ when $(\sigma_K^2, \eta_1, \eta_2) = (1, 1, 1)$

Figure 6. Realizations of Layer 35 permeability coefficient when different η_1 and η_2 are used.

Table 8. Testing records when Layer 1, 4, 34, 35, 35 *, and 35 ** are considered.

Case	**Testing NRMSE**	**sMAPE (l_∞ Error) in**		
		Iteration Number	λ_{min}	λ_{max}
Layer 1	8.02×10^{-2}	1.04×10^{-1} (2)	3.74×10^{-6} (3.71×10^{-5})	4.00×10^{-2} (1.36×10^{-1})
Layer 4	6.95×10^{-2}	7.18×10^{-2} (1)	1.11×10^{-5} (1.36×10^{-4})	1.61×10^{-2} (6.85×10^{-2})
Layer 34	5.08×10^{-2}	2.52×10^{-3} (1)	3.25×10^{-5} (1.74×10^{-4})	3.06×10^{-2} (8.47×10^{-2})
Layer 35	2.48×10^{-2}	3.65×10^{-2} (1)	7.52×10^{-6} (7.50×10^{-5})	1.36×10^{-3} (6.66×10^{-2})
Layer 35 *	2.27×10^{-2}	3.89×10^{-2} (1)	7.57×10^{-6} (7.17×10^{-5})	1.27×10^{-3} (6.59×10^{-2})
Layer 35 **	2.75×10^{-2}	4.29×10^{-2} (1)	8.52×10^{-6} (7.13×10^{-5})	1.81×10^{-3} (8.10×10^{-2})

In Table 8, we clarify that the results of Layer 35 * also use Layer 35 permeability as the expected function but with $(\eta_1, \eta_2) = (0.2, 0.125)$. Similarly, Layer 35 ** is with another set of parameters $(\eta_1, \eta_2) = (1, 1)$. Each row corresponds to the results of different testing sets, however, all the results are obtained using the same neural network of Layer 35 in Table 7. Therefore, the testing results of Layer 35 are used as a reference to decide whether the results of other testing sets are good or not.

We first focus on the testing results of Layer 35. From the corresponding histograms in the left column of Figure 7, besides the iteration number required for predicted eigenvectors are still usually larger by 1, we can observe a large area of overlapping, in particular, there is only 8.4% of data that are not overlapped in the histogram of maximum eigenvalues. Moreover, the small NRMSE and sMAPE again confirm with the graphical results. For generalization tests, we then consider other testing sets with different stochastic behaviors and expected functions.

When the stochastic behavior of Layer 35 is changed to $(\sigma_K^2, \eta_1, \eta_2) = (1, 0.2, 0.125)$ and $(\sigma_K^2, \eta_1, \eta_2) = (1, 1, 1)$, we list the corresponding sMAPE and l_∞ norms of differences in iteration numbers, maximum eigenvalues and minimum eigenvalues of pre-conditioners in the last two rows of Table 8. We can see the values of errors are very similar to the row of Layer 35 and the NRMSE are very close. Therefore, we expect that the Layer 35 neural network can generalize the coefficient functions with similar stochastic properties, and give a good approximation on dominant eigenvectors for the BDDC pre-conditioner. It is then verified by the histograms in the middle column of Figure 7. Moreover, a large population of samples are concentrated around 0 in the histograms of difference between the estimated pre-conditioner and target pre-conditioner in Figure 8. All these show our method performs well with stochastic oscillatory and high contrast coefficients, and also coefficient functions with similar stochastic properties.

Finally, we compare the performances when the expected function is changed to Layer 1, 4, and 34. Note, the NRMSE, sMAPE and l_∞ errors of the three layers are almost all much higher than those of Layer 35, which are crucial clues for a worse performance. It is clear that when the permeability fields do not have many similarities such as Layer 1 and 4, the testing NRMSE increases distinctly. Even though the permeability fields of Layer 34 and Layer 35 share some common properties and have a near magnitude, unlike previous results for Layer 35 and Layer 35 *, we can see in the right column of Figure 7, the prediction output and target output on maximum and minimum eigenvalues of pre-conditioners are even two populations with different centres and only a little overlapping area. The main reason is a minor difference in $K(\boldsymbol{x}, \omega)$ can already cause a huge change after taking exponential function. Therefore, when $E[K](\boldsymbol{x})$ is changed to another entirely different mean permeability with just some common properties, the feed-forward neural network may not be able to capture these new characteristics, which will be part of our future research.

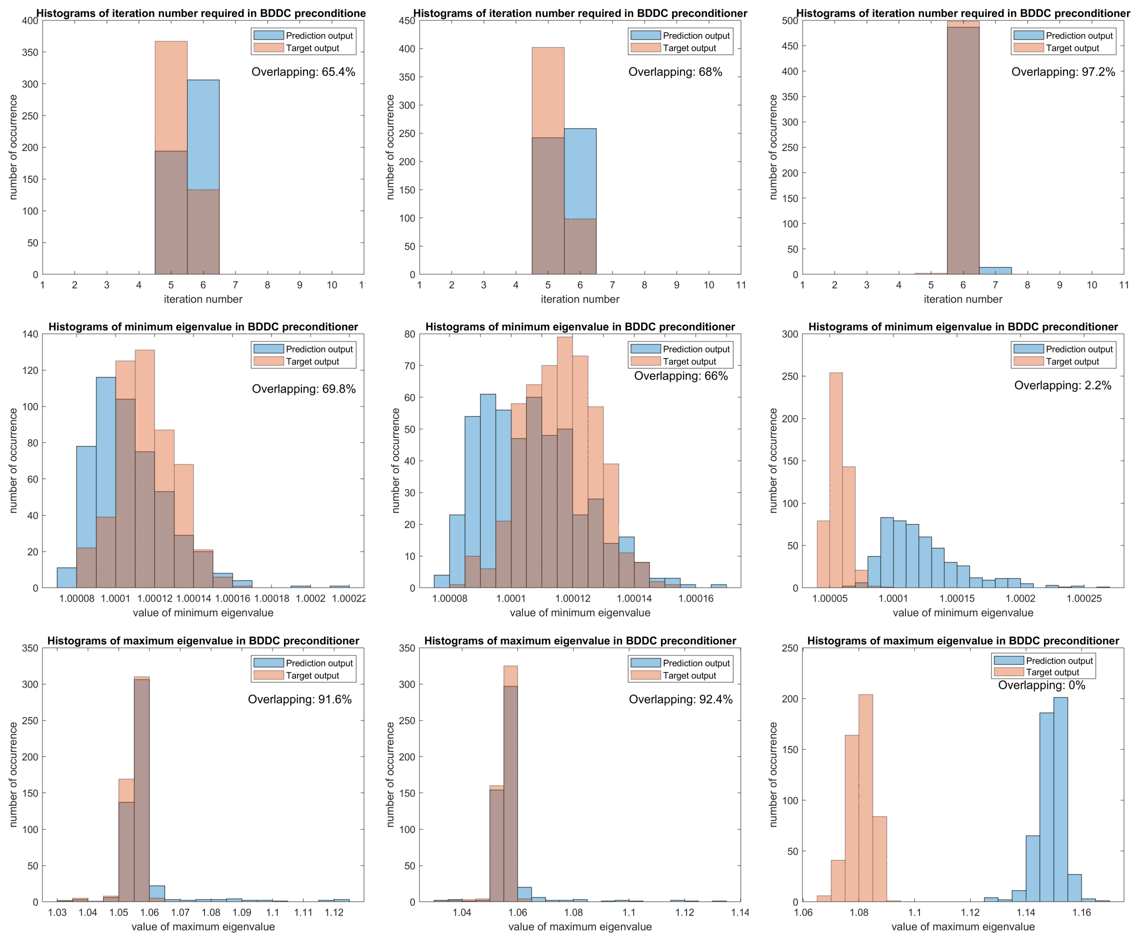


Figure 7. Comparisons in the performance of the BDDC preconditioner when expected functions of SPE10 Layer 35 (**left column**), SPE10 Layer 35 * (**middle column**) and SPE10 Layer 34 (**right column**) are used.

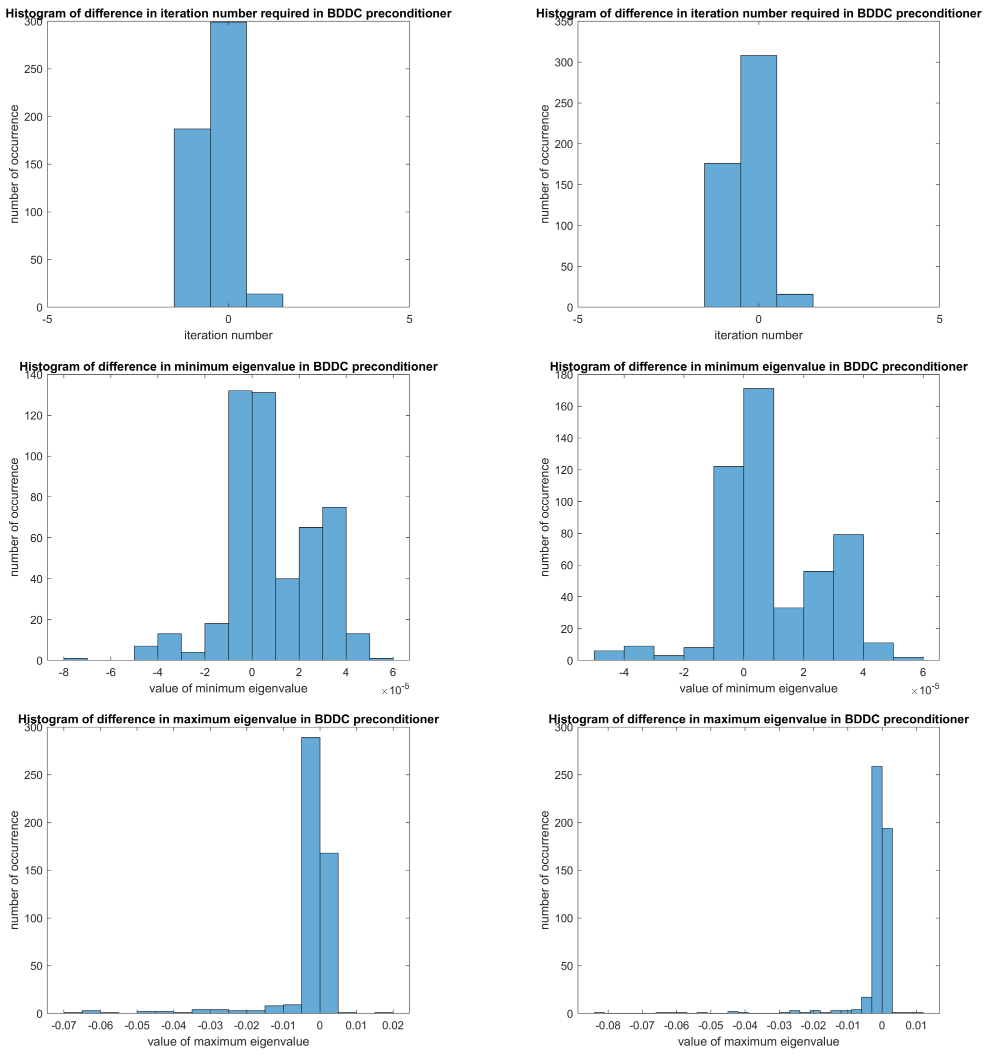
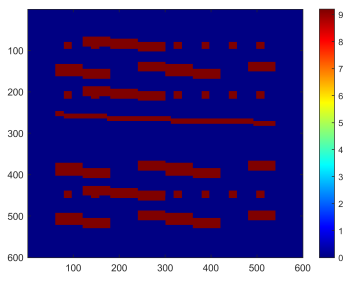


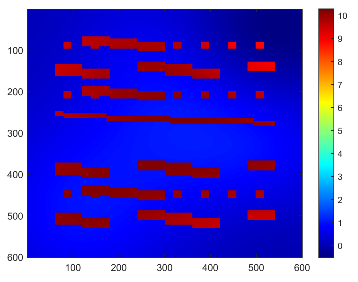
Figure 8. Differences in the performance of the BDDC preconditioner when expected functions of SPE10 Layer 35 (**left column**) and SPE10 Layer 35 * (**right column**) are used.

3.5. Exponential Covariance Function with Multiple Hidden Layers

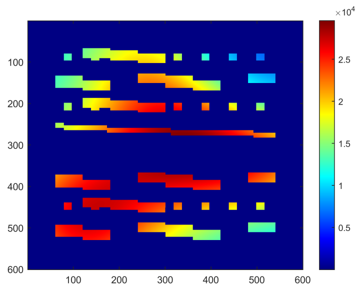
It is noted that after taking exponential function, the permeability fields in previous experiments only show some global peaks, the single layer structure of neural network may then suffice to give an accurate prediction. Therefore, in this subsection, we illustrate an example with a high contrast coefficient cutting more interfaces and a realization of this new coefficient function $\widehat{\rho}(\boldsymbol{x},\omega)$ is displayed in Figure 9. It is of note that, the stochastic properties considered are still the exponential covariance function with $(\sigma_K^2,\eta_1,\eta_2) = (1, 0.25, 0.25)$.



(**a**) Expected function $E[\widehat{K}](\boldsymbol{x})$



(**b**) A realization of $\widehat{K}(\boldsymbol{x},\omega)$



(**c**) A realization of permeability $\widehat{\rho}(\boldsymbol{x},\omega)$

Figure 9. Realization of new coefficient function $\widehat{\rho}(\boldsymbol{x},\omega)$.

To extract more features and details from the new coefficient, we consider a much finer grid sizes, which are $H = 1/10$ and $h = 1/600$, and the spatial domain is still the unit square $[0,1]^2$. With respect to this change, we change the following training conditions accordingly:

- Number of hidden layers: $L = 1$ or $L = 2$;
- Number of neurons in the hidden layers: $n^{(1)} = 10$ and $n^{(2)} = 0, 5$ or 7;
- Number of neurons in the output layer: $O =$ 21,240;
- Sample size of training set: $M = 2000$;
- Sample size of testing set: $M = 100$.

Other conditions are kept to be unchanged as in Section 3.2. We show the new training and testing records in Tables 9 and 10.

Table 9. Training records when new coefficient function $\widehat{\rho}(\boldsymbol{x},\omega)$ is considered.

$(n^{(1)}, n^{(2)})$	Epochs Trained	Training NRMSE	Training Time	Preparation Time for 1 Sample
(10,0)	1.00×10^6	3.81×10^{-2}	4.89×10^4 s	130.77 s
(10,5)	6.46×10^5	4.44×10^{-2}	2.65×10^4 s	
(10,7)	1.00×10^6	4.11×10^{-2}	4.62×10^4 s	

Table 10. Testing records when new coefficient function $\widehat{\rho}(\boldsymbol{x},\omega)$ is considered.

$(n^{(1)}, n^{(2)})$	Testing NRMSE	sMAPE (l_∞ Error) in		
		Iteration Number	λ_{min}	λ_{max}
(10,0)	5.47×10^{-2}	2.12×10^{-2} (4)	1.86×10^{-6} (2.98×10^{-5})	1.42×10^{-1} (4.34×10^2)
(10,5)	3.57×10^{-2}	7.81×10^{-3} (5)	1.21×10^{-6} (3.53×10^{-5})	4.55×10^{-2} (7.89×10^1)
(10,7)	2.30×10^{-2}	1.12×10^{-2} (5)	1.13×10^{-6} (2.50×10^{-5})	9.18×10^{-2} (7.01×10^1)

On one hand, when the number of neurons in the second hidden layer $n^{(2)}$ increases, the training results do not show a distinct difference, the three training NRMSE are all around 0.04 with a much longer training time than previous experiments. However, on the other hand, the testing results do show some improvements when $n^{(2)}$ increases. Obviously, the testing NRMSE and pre-conditioner errors improve when a second hidden layer is introduced. Moreover, the testing NRMSE also decreases when $n^{(2)}$ increases. We can see the testing NRMSE of multiple hidden layers are slightly lower than the corresponding training NRMSE.

This indicates that for more complicated coefficient functions, introduction of multiple hidden layers could help to bring better performance. However, as it also increases the complexity of neural network and the training time, thus, a single layer of neural network should be considered first if the problem is not complicated. In Figure 10, comparisons of the iteration number, maximum and minimum eigenvalues are illustrated in histograms for easy observation. Unlike the histograms in Figure 7, we can see more iterations are needed for solving the problem and more outliers exist in the maximum eigenvalue comparison as the problem itself becomes more complicated than previous experiments.

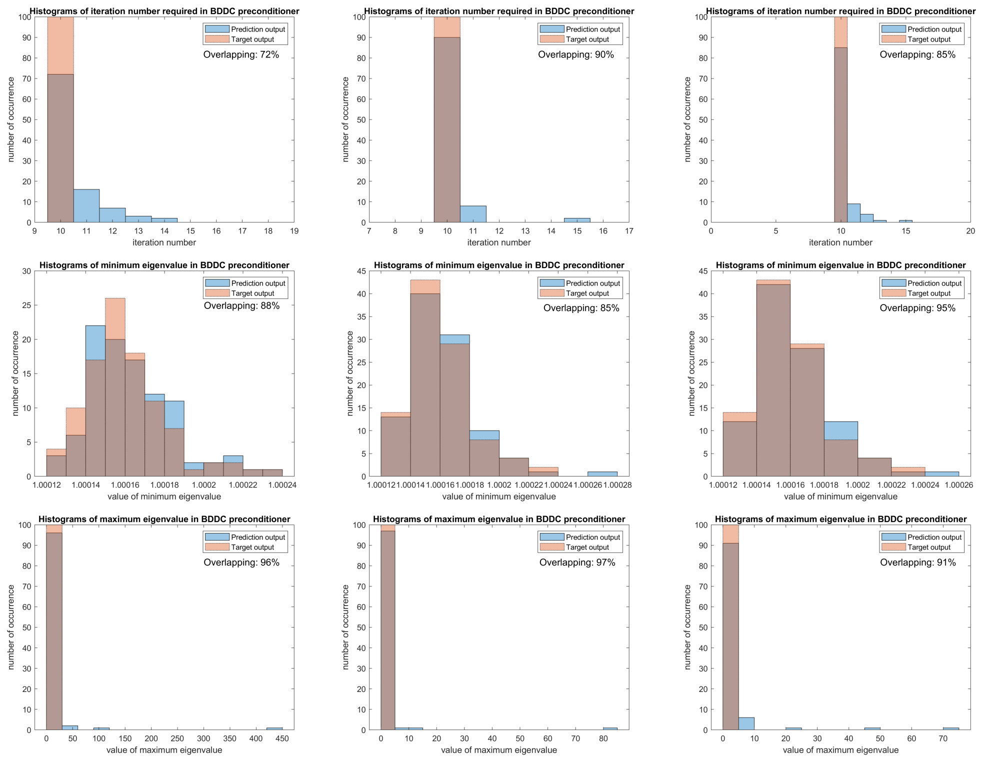


Figure 10. Comparisons in the performance of the BDDC pre-conditioner when single hidden layer (**left column**), two hidden layers with $n^{(2)} = 5$ (**middle column**) and two hidden layers with $n^{(2)} = 7$ (**right column**) are used.

4. Conclusions

A new learning adaptive BDDC algorithm is introduced and it consists of three main parts, which are the Gaussian random variables in Karhunen–Loève expansion, artificial neural network and the dominant eigenvectors obtained from the adaptive BDDC algorithm in the coarse spaces. The considered neural network acts as a fast computational tool, which turns the input Gaussian random variables into predicted dominant eigenvectors as output. In addition, the neural network in the proposed algorithm is suitable for other permeability coefficients with similar stochastic properties for generalization purpose. Numerical results confirm the efficiency of the proposed algorithm and the generalization abilities. On the other hand, we currently just use the simplest feed-forward neural network structure without retraining when new characteristics are entered. In our future plan, we will improve the neural network structure that matches with the computational domain and some adaptive BDDC features. Deep learning technique will also be included for higher accuracy in prediction, pre-conditioning, and also better generalization ability on wider aspects of applications.

Author Contributions: Conceptualization, E.C. and H.-H.K.; methodology, E.C. and H.-H.K.; software, H.-H.K. and M.-F.L.; investigation, E.C., M.-F.L., and L.Z.; writing—original draft preparation, M.-F.L. and L.Z.; writing—review and editing, E.C. and H.-H.K.; visualization, M.-F.L. and L.Z.; supervision, E.C.; project administration, E.C. All authors have read and agreed to the published version of the manuscript.

Funding: The research of Eric Chung is supported by Hong Kong RGC General Research Fund (Project numbers 14304719 and 14302018). The research of Hyea-Hyun Kim is supported by the National Research Foundation of Korea (NRF) grants funded by NRF-2015R1A5A1009350 and by NRF-2019R1A2C1010090.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found here: https://www.spe.org/web/csp/datasets/set02.htm (accessed on 3 June 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chung, E.T.; Efendiev, Y.; Leung, W.T. An adaptive generalized multiscale discontinuous Galerkin method for high-contrast flow problems. *Multiscale Model. Simul.* **2018**, *16*, 1227–1257. [CrossRef]
2. Chung, E.T.; Efendiev, Y.; Li, G. An adaptive GMsFEM for high-contrast flow problems. *J. Comput. Phys.* **2014**, *273*, 54–76. [CrossRef]
3. Babuška, I.; Nobile, F.; Tempone, R. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM J. Numer. Anal.* **2007**, *45*, 1005–1034. [CrossRef]
4. Babuska, I.; Tempone, R.; Zouraris, G.E. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM J. Numer. Anal.* **2004**, *42*, 800–825. [CrossRef]
5. Ghanem, R.G.; Spanos, P.D. *Stochastic Finite Elements: A Spectral Approach*; Courier Corporation: North Chelmsford, MA, USA, 2003.
6. Brunton, S.L.; Noack, B.R.; Koumoutsakos, P. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **2020**, *52*, 477–508. [CrossRef]
7. Kutz, J.N. Deep learning in fluid dynamics. *J. Fluid Mech.* **2017**, *814*, 1–4. [CrossRef]
8. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A deep learning library for solving differential equations. *SIAM Rev.* **2021**, *63*, 208–228. [CrossRef]
9. Zhao, W.; Du, S. Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 4544–4554. [CrossRef]
10. Heinlein, A.; Klawonn, A.; Lanser, M.; Weber, J. Combining machine learning and domain decomposition methods for the solution of partial differential equations—A review. *GAMM Mitteilungen* **2021**, *44*, e202100001. [CrossRef]
11. Vasilyeva, M.; Leung, W.T.; Chung, E.T.; Efendiev, Y.; Wheeler, M. Learning macroscopic parameters in nonlinear multiscale simulations using nonlocal multicontinua upscaling techniques. *J. Comput. Phys.* **2020**, *412*, 109323. [CrossRef]
12. Wang, Y.; Cheung, S.W.; Chung, E.T.; Efendiev, Y.; Wang, M. Deep multiscale model learning. *J. Comput. Phys.* **2020**, *406*, 109071. [CrossRef]
13. Chung, E.; Leung, W.T.; Pun, S.M.; Zhang, Z. A multi-stage deep learning based algorithm for multiscale model reduction. *J. Comput. Appl. Math.* **2021**, *394*, 113506. [CrossRef]
14. Yeung, T.S.A.; Chung, E.T.; See, S. A deep learning based nonlinear upscaling method for transport equations. *arXiv* **2020**, arXiv:2007.03432.
15. Burrows, S.; Frochte, J.; Völske, M.; Torres, A.B.M.; Stein, B. Learning overlap optimization for domain decomposition methods. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Gold Coast, Australia, 14–17 April 2013; pp. 438–449.
16. Heinlein, A.; Klawonn, A.; Lanser, M.; Weber, J. Machine Learning in Adaptive Domain Decomposition Methods—Predicting the Geometric Location of Constraints. *SIAM J. Sci. Comput.* **2019**, *41*, A3887–A3912. [CrossRef]
17. Li, K.; Tang, K.; Wu, T.; Liao, Q. D3M: A deep domain decomposition method for partial differential equations. *IEEE Access* **2019**, *8*, 5283–5294. [CrossRef]
18. Dostert, P.; Efendiev, Y.; Hou, T.Y.; Luo, W. Coarse-gradient Langevin algorithms for dynamic data integration and uncertainty quantification. *J. Comput. Phys.* **2006**, *217*, 123–142. [CrossRef]
19. Wheeler, M.F.; Wildey, T.; Yotov, I. A multiscale preconditioner for stochastic mortar mixed finite elements. *Comput. Methods Appl. Mech. Eng.* **2011**, *200*, 1251–1262. [CrossRef]
20. Zhang, D.; Lu, Z. An efficient, high-order perturbation approach for flow in random porous media via Karhunen–Loève and polynomial expansions. *J. Comput. Phys.* **2004**, *194*, 773–794. [CrossRef]
21. Kim, H.H.; Chung, E.; Wang, J. BDDC and FETI-DP preconditioners with adaptive coarse spaces for three-dimensional elliptic problems with oscillatory and high contrast coefficients. *J. Comput. Phys.* **2017**, *349*, 191–214. [CrossRef]
22. Kim, H.H.; Chung, E.T. A BDDC algorithm with enriched coarse spaces for two-dimensional elliptic problems with oscillatory and high contrast coefficients. *Multiscale Model. Simul.* **2015**, *13*, 571–593. [CrossRef]
23. Dohrmann, C.R. A preconditioner for substructuring based on constrained energy minimization. *SIAM J. Sci. Comput.* **2003**, *25*, 246–258. [CrossRef]
24. Mandel, J.; Dohrmann, C.R.; Tezaur, R. An algebraic theory for primal and dual substructuring methods by constraints. *Appl. Numer. Math.* **2005**, *54*, 167–193. [CrossRef]
25. Li, J.; Widlund, O.B. FETI-DP, BDDC, and block Cholesky methods. *Int. J. Numer. Methods Eng.* **2006**, *66*, 250–271. [CrossRef]
26. Toselli, A.; Widlund, O. *Domain Decomposition Methods—Algorithms and Theory*; Springer: Berlin, Germany, 2005; Volume 34.

27. Klawonn, A.; Radtke, P.; Rheinbach, O. A comparison of adaptive coarse spaces for iterative substructuring in two dimensions. *Electron. Trans. Numer. Anal.* **2016**, *45*, 75–106.
28. Klawonn, A.; Radtke, P.; Rheinbach, O. FETI-DP with different scalings for adaptive coarse spaces. *Proc. Appl. Math. Mech.* **2014**. [CrossRef]
29. Anderson, W.N., Jr.; Duffin, R.J. Series and parallel addition of matrices. *J. Math. Anal. Appl.* **1969**, *26*, 576–594. [CrossRef]
30. Dohrmann, C.R.; Pechstein, C. Modern Domain Decomposition Solvers: BDDC, Deluxe Scaling, and an Algebraic Approach. 2013. Available online: http://people.ricam.oeaw.ac.at/c.pechstein/pechstein-bddc2013.pdf (accessed on 3 June 2021).
31. Schwab, C.; Todor, R.A. Karhunen–Loève approximation of random fields by generalized fast multipole methods. *J. Comput. Phys.* **2006**, *217*, 100–122. [CrossRef]
32. Wang, L. Karhunen–Loève Expansions and Their Applications. Ph.D. Thesis, London School of Economics and Political Science, London, UK, 2008.
33. Møller, M.F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.* **1993**, *6*, 525–533. [CrossRef]
34. Makridakis, S. Accuracy measures: Theoretical and practical concerns. *Int. J. Forecast.* **1993**, *9*, 527–529. [CrossRef]