MULTISCALE MODEL REDUCTION AND LEARNING

A Dissertation

by

MIN WANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Yalchin Efendiev
Co-Chair of Committee,	Tsz Shun Eric Chung
Committee Members,	I. Yucel Akkutlu
	Raytcho Lazarov
	Jianxin Zhou
	Yuhe Wang
Head of Department,	Emil J. Straube

August 2019

Major Subject: Mathematics

ABSTRACT

Many engineering problems have multiscale features. These problems usually require some model reduction since the computational cost of a fine-scale solution is extremely expensive. Existing model reduction methods such as Generalized Multiscale Finite Element Method (GMsFEM) and Non-local multi-continuum approach (NLMC) have shown extensive success in solving multiscale problems especially on various flow simulation problems.

However, there are still challenges in developing effective multiscale models for flow in more complicated heterogeneous media. The geometries of domain, coexistence of multiple continuum, and lack of observation data can all give rise to the difficulty of developing the reduced-order model. In this thesis, I will concentrate on the development of novel multiscale methods following the idea of the existing model reduction methods to address such problems. Moreover, deep learning techniques are combined to overcome certain difficulties met along model construction. These proposed models are targeted to tackle specific problems, where the performance is verified both numerically and analytically.

For instance, flow simulation within a heterogeneous thin domain is one of such challenging problems. Though homogenization methods are proven to be successful when the media have clear scale separation, that's not always the case for flow simulation within a capillary system. Using only one basis function in each coarse region can lead to large errors. We thus design a customized GMsFEM instead, which is able to automatically enrich the approximation space and significantly reduce the error.

When simulating flow in a fractured vuggy reservoir, on the other hand, I develop a coarse solver under the framework of GMsFEM by combining it with multi-continuum model and Discrete Fracture Model (DFM). Instead of treating the media as a single continuum, I treat the multiscale formation hierarchically and consider it as a coupled system of matrix, fractures and vugs. This allows us to explicitly represent the mass transfers between continuum as well as model the local effects of the discrete fractures.

We further investigate how deep learning can facilitate multiscale model construction for nonlinear flow dynamics. Utilizing a multi-layer neural network to approximate the reduced order model, the observed data can be easily incorporated to adjust the model. Deep learning techniques are also used to conduct model reduction. With a soft thresholding operator as an activation function, a novel neural network is proposed which can identify important multiscale features that are crucial in modeling the underlying flow. The forward input-output maps are thus learned in a reduced way.

Extensive applications to engineering problems and numerical analysis are presented in supplement of the proposed approaches. It is shown that our proposed methods can significantly advance the computational efficiency and accuracy for multiscale flow simulation in various heterogeneous media.

ACKNOWLEDGMENTS

Firstly, I would like to express my special appreciation to my advisor Prof. Yalchin Efendiev for encouraging my research and for allowing me to grow as a research scientist. His advice on both research and career have been invaluable. On a personal level, Prof. Efendiev inspired me by his hardworking and passionate attitude.

Secondly, I am very grateful to my co-advisor Dr. Eric Chung for his brilliant suggestions on research and my career developments. He always shows great patience in reviewing my drafts and answering my questions.

I would also like to thank my committee members, Dr. Jianxin Zhou, Dr. Raytcho Lazarov, Dr. I. Yucel Akkutlu, and Dr. Yuhe Wang. I would especially like to thank Dr. Yuhe Wang for his generosity and hospitality he showed during my visit to Qatar. The great opportunities Dr. Wang has provided to attend international conferences and to communicate with scientists in different fields is significantly beneficial to me.

I appreciate all the help from the Department of Mathematics, Texas A&M University. It would be impossible for me to pursue a research career without the support of the department. The Department of Mathematics not only provides me with opportunities to attend academic seminars and conferences but also allows me to collaborate with my talented colleagues.

Last but not least, I would like to express my deepest gratitude to my family and friends. This dissertation would not have been possible without their warm love, continued patience, and endless support. I would like to send my special thanks to Yanbo Li, Siu Wun Cheung, Jingyan Zhang, Yating Wang, Win Tat Leung, Xin Ma, Qing Zhang, Peng Wei and Bohan Zhou for always being both my friends and family whenever I needed.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professor Yalchin Efendiev and Professor Eric Chung of the Department of Mathematics.

Funding Sources

Graduate study was partially supported by the Department of Mathematics from Texas A& M University. All other work conducted for the thesis was completed by the student independently.

NOMENCLATURE

GMsFEM	Generalized Multiscale Finite Element Method
Ω	Computational domain
$\mathcal{T}^h$	A partition of Ω into fine elements
$\mathcal{T}^H$	A partition of Ω into coarse elements
h	The fine mesh size
H	The coarse mesh size
x_i	A coarse grid node
E_i	A coarse grid edge
ω_i	A coarse neighborhood of coarse node x_i or coarse edge E_i
ω_i^+	An oversampled subdomain ω_i
K_i	A coarse grid element
K_i^+	A oversampled subdomain for K_i
χ_i	The partition of unity function
u_h	The fine-grid solution
u_H	The coarse-grid solution
V_h	The fine-scale space
N_v	The number of coarse-grid nodes
N_e	The number of coarse-grid edge

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

Modeling of multiscale process has been of great interest in diverse applied fields. These include flow in porous media, mechanics, biological applications, and so on. Among many tools that have been developed to address the characteristics of such problems, an easy and convenient way to explore such process is to use a fine scale simulation. A common fine scale simulation can be conducted under the frameworks such as the Finite Element Method (FEM) [1], Finite Volume Method (FVM) or Finite Difference Method (FDM). However, resolving the fine-scale features of such processes could be computationally expensive due to scale disparity.

For example, when considering mass transfer process in porous media, the media properties can vary over many scales. Due to the multiscale nature of the medium and the coexistence of different continua, the fine-grid resolution gives rise to a large number of degrees of freedom. Thus, simulating flow in a multiscale media could be considerably demanding.

Therefore, some types of reduced-order models are derived to cut down the computational cost. Such models are commonly constructed following a local model reduction scheme. In particular, researchers find ways to bring the fine-grid information to the coarse grid. This generally reduces to constructing appropriate low-order approximation spaces when solving a governing PDEs numerically. A fine partition is first required to divide the domain into local pieces and then a global description of the flow is obtained by putting the local solutions together.

Homogenization is one commonly used local model reduction method. The domain of interest is first partitioned into many coarse blocks, and the effective properties are then calculated for each coarse block. The idea is to homogenize local heterogeneous media using information in finer scales within the block. These pre-computed properties can thus catch and average fine-scale characteristics and further calibrate the coarse solution accordingly [2]. This up-scaling scheme has been proved to be quite effective for simulations on media with scale separation or periodicity

[3], but it fails to model other cases especially when multiple continuum coexist [4, 5, 6, 7, 8, 9, 10].

In order to overcome the limitations of the homogenization scheme and enrich the heterogeneous information at the local fine-scale region, many other multiscale methods and solvers [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 9, 24, 25, 26, 27, 28] are also designed. These methods usually construct coarse spaces and sustain unresolved scales to a desired accuracy via additional computing. Examples include the Multiscale Finite Element Method (MsFEM) [29, 30, 31, 32], the Generalized Multiscale Finite Element Method (GMsFEM), the Nonlocal Multicontinuum Method (NLMC), etc.

For these methods, like homogenization, the computational domain is first partitioned with a coarse grid $\mathcal{T}^H$. A local model reduction methods is then able to identify the local multiscale basis functions supported in each coarse region on the fine grid $\mathcal{T}^h$, which is essentially a refinement of $\mathcal{T}^H$ that resolves all multiscale features. Therefore, the macroscopic equations can then be formulated as a coarse-scale system using the local multiscale basis functions. Moreover, the space that formed by these basis functions has a much smaller dimension compared to that of a fine-scale space.

As in many model reduction techniques, the computations of multiscale basis functions, can be performed in an offline manner. For a fixed multiscale parameter, these multiscale basis functions are reusable for any force terms and boundary conditions. Therefore, these methods provide a substantial computational savings in the online stage, in which a coarse-scale system is constructed and solved on the reduced-order space.

The GMsFEM was first developed in [33]. It has been proven that GMsFEM can strengthen the ability of MsFEM on solving multiscale problems. By conducting a spectral decomposition over the local snapshot space, GMsFEM can identify basis functions corresponding to dominant modes of local heterogeneous regions and eliminate unnecessary degrees of freedom on a coarse-grid level. This makes automatic enrichment of the multiscale space possible [34, 35]. In the paper [36], a one-on-one correspondence between GMsFEM basis functions and high-conductivity networks was presented. This property of basis makes GMsFEM a necessity when dealing with

practical examples like carbonate reservoir simulations, as many fracture channels may coexist in a single local region. Additionally, GMsFEM can also be easily adopted to couple with other models such as DFM and Multi-continuum Models, which provides a way to describe a fractured vuggy reservoir in a hierarchical fashion. It allows us to consistently develop an approximation space that contains prominent sub-grid scale information based on the multi-continuum and DFM.

NLMC [37], on the other hand, identifies the coarse-grid parameters in each cell and their connectivity to neighboring variables. This approach derives its foundation from the Constraint Energy Minimizing Generalized Multiscale Finite Element Method (CEM-GMsFEM) [38], which has a convergence rate H/Λ, where Λ represents the local heterogeneities. Using the concept of CEM-GMsFEM, NLMC defines new basis functions such that the degrees of freedom have physical meanings (in this case, they represent the solution averages). In this dissertation, we will limit our study of model reduction methods to GMsFEM and NLMC.

These methods have been successfully applied to construct reduced-order model for various multiscale process. Though the accuracy and affordability of them are widely celebrated, greater demands have been placed on such models. We expect the models to reflect not only the governing PDE but also the observation data from the realistic process. However, such forward models are difficult to construct.

The deep learning concepts, on the other hand, provides a straightforward approach to construct data-based models. Taking the observed data as training data, a deep neural network can be optimized to "learn" the underlying true process. Mathematically speaking, the neural network will approximate the operator that maps inputs to output from the given examples.

We thus would like to apply the deep learning in multiscale model constructions. Hence, we remove the limitations that current multiscale models have in honoring observation while maintaining their advantages as successful coarse solvers. In fact, for nonlinear problems that are in the presence of observed data, deep learning can be used to construct multiscale models that are conditioned to these data [39, 40, 41]. Alternatively, by supplementing observation training data with the data simulated using a coarse model, a deep neural network can build a desired model that

interpolates between observation and simulation.

A common neural network is usually composed of a relatively large number of layers of non-linear processing units, called neurons, for feature extraction. The neurons are connected to other neurons in the successive layers. The information propagates from the input, through the intermediate hidden layers, and to the output layer. In the propagation process, the output in each layer is used as the input in the consecutive layer. In between layers, a nonlinear activation function is used as the nonlinear transformation on the input, which increases the descriptive power of neural networks. Extensive applications show that neural networks can accurately represent and approximate a large class of non-linear functions with complicated forms or even without explicit expression. Convincing cases include speech recognition, image recognition, as well as natural language processing[42, 43, 44].

Scientists have also explored other ways to apply deep learning to model reductions and partial differential equations. In [8], the authors studied deep convolution networks for surrogate model construction on dynamic flow problems in heterogeneous media. In [45], the authors studied the relationship between the residual networks (ResNet) and characteristic equations of linear transport, and proposed an interpretation of deep neural networks by continuous flow models. In [46], the authors combined the idea of the Ritz method and deep learning techniques to solve elliptic problems and eigenvalue problems. In [47], a neural network has been designed to learn the physical quantities of interest as a function of random input coefficients. The concept of using deep learning to generate a reduced-order model for a dynamic flow has also been applied to proper orthogonal decomposition (POD) [48].

In this dissertation, we concentrate on constructing reduced-order model for multiscale process under the framework of GMsFEM and NLMC. Moreover, the multiscale models are improved when the deep learning techniques are combined.

1.2 Outline of the Dissertation

In Chapter 2, we present preliminary background materials. We first state the physical model that we are interested in. Later, the constructions of multiscale models following GMsFEM and

NLMC are illustrated. Additionally, a general introduction of deep learning is provided.

In Chapter 3, we propose a numerical scheme based on the GMsFEM and a triple-continuum model to simulate flow in a highly heterogeneous reservoir. We aim to obtain a more efficient numerical approach that can explicitly represent the interactions among different continua. To enhance the applicability of our proposed model, we also combine the Discrete Fracture Model (DFM). In the proposed model, the GMsFEM, as an advanced model reduction technique, enables capturing the multiscale flow dynamics. This is accomplished by systematically generating an approximation space through solving a series of local snapshot and spectral problems. The resulting eigenfunctions can pass the local features to the global level when acting as basis functions in the global coarse problems. Several numerical experiments are conducted to confirm the success of our proposed method, and a rigorous convergence proof is also given.

In Chapter 4, we further study GMsFEM for simulation of flow in a narrow domain. In particular, we consider the mixed form of the elliptic equation. We aim to analyze how the geometry affects the convergence rate of our multiscale approximation. We limit our study to the case when the length/width ratio of the domain is large. Numerical results also validate the necessity of using GMsFEM as adopting more than one basis function in each coarse neighborhood can significantly improve the accuracy.

In Chapter 5, we introduce the idea of deep learning to the construction of multiscale model. We would like to model the fluid dynamics within a heterogeneous domain utilizing field data. The neural network is used to model the forward map underlying the fluid dynamics. The network is defined in a way that the connections between layers are decided by the NLMC. The observation data are then used as training data together with computational data to condition the model. Numerical examples show that the design can naturally lighten the network training. Moreover, the learned model is able to honor the realistic physical process under observation.

In Chapter 6, we further develop the idea of multiscale learning. We reformulate the feed-forward neural network as the solution to an optimization problem with l_1 regularization. The weights and biases learned can then be shown to have a solid relationship with the operator been

approximated. Based on this understanding, we then present a neural network architecture that can conduct further global model reduction to the reduced-order solutions while sustaining accurate flow simulation.

Finally, Chapter 7 concludes this dissertation.

2. PRELIMINARY

In this chapter, we present an overview of the problems in heterogeneous domains, as well as the construction of some multiscale models. We also introduce the general concepts of the deep learning.

2.1 Problems of Interest

In this dissertation, we consider the flow problem in highly heterogeneous media

$$-\text{div}\big(\kappa\,\nabla u\big) = g, \quad \text{in } \Omega. \tag{2.1}$$

We also consider a time-dependent flow problem in heterogeneous domain , that is

$$\frac{\partial u}{\partial t} - \text{div}(\kappa \nabla u) = g(t), \quad \text{in } \Omega, \tag{2.2}$$

where Ω is the computational domain, κ is the permeability coefficient in $L^{\infty}(\Omega)$, and g is a source function in $L^2(\Omega)$ while u represents the solution to be sought. We assume the coefficient κ is highly heterogeneous with high contrast in space (see Figure 2.1 for an illustration). The classical finite element method for solving (2.1) is given by: find $u_h \in V_h$ such that

$$a(u_h, v) = (f, v), \quad v \in V_h. \tag{2.3}$$

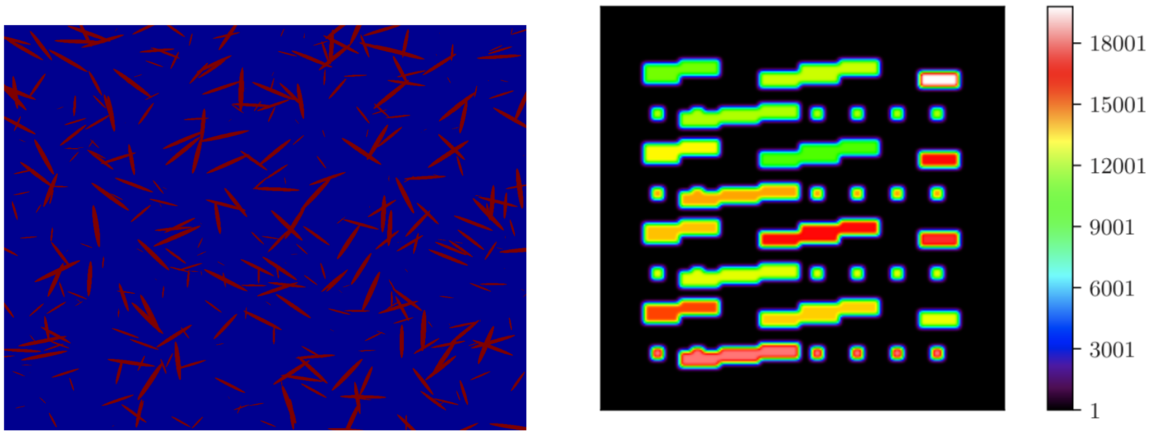
For (2.2), that is:

$$(\frac{\partial u_h}{\partial t}, v) + a(u_h, v) = (g, v), \quad \text{for all } v \in V_h. \tag{2.4}$$

Here, V_h is a standard conforming finite element space over a partition $\mathcal{T}^h$ of Ω with mesh size h, and the bilinear forms $a(\cdot,\cdot)$ and $(\cdot,\cdot)$ are defined as

$$\begin{aligned} a(u,v) &:= \int_\Omega \kappa \nabla u \cdot \nabla v \, dx, \\ (u,v) &:= \int_\Omega uv \, dx. \end{aligned} \tag{2.5}$$

However, with the highly heterogeneous property of coefficient κ, the domain has to be partitioned into extremely small elements to capture the underlying fine-scale features of κ. This ends up with a large computational cost.



(a) Random channel (b) Channelized high-contrast domain

Figure 2.1: Examples of heterogeneous domain.

2.2 Multiscale Basis Construction

To reduce the dimension of the resulting system, we would like to construct an approximation space V_{ms} that has less degrees of freedom while sustaining the approximation accuracy. As discussed in 1.1, such approximation space can be taken as the expansion of multiscale basis functions, which are constructed following GMsFEM or NLMC.

2.2.1 Construction of Basis Functions for GMsFEM

For GMsFEM, basis functions are computed locally to capture the underlying dominant modes[33]. More specifically, a local snapshot space is first constructed followed by a spectral decomposition aiming at keeping only the dominant degrees of freedom.

- Step 1: Partition of domain

  We first partition the computational domain Ω with a coarse mesh $\mathcal{T}^H$. The coarse mesh is then refined to a fine mesh $\mathcal{T}^h$ with mesh size $h \ll H$, which is fine enough to restore the multiscale properties of the problem. Let $\mathcal{V}^H := \{x_i \,|1 \leq i \leq N_v\}$ be the set of nodes of the coarse mesh $\mathcal{T}^H$. For each coarse grid node $x_i \in \mathcal{V}^H$, the coarse neighborhood ω_i is defined by

  $$\omega_i := \bigcup_j \{K_j \in \mathcal{T}^H \mid x_i \in \overline{K}_j\}, \tag{2.6}$$

  that is, the union of the coarse elements $K_j \in \mathcal{T}^H$ that contains the coarse grid node x_i. An example of the coarse and fine mesh, coarse blocks and a coarse neighborhood is shown in Figure 2.2.

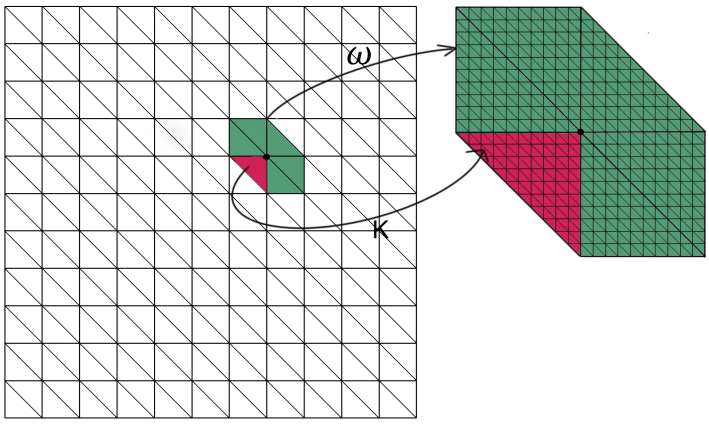


Figure 2.2: An illustration of coarse mesh (left), a coarse neighborhood and coarse blocks (right).

For each coarse neighbourhood ω_i, we construct multiscale basis functions $\{\psi_k^{\text{ms},\omega_i}\}_{k=1}^{L_{\omega_i}}$. Further, the global multiscale finite element space V_{ms} is constructed with all such $\psi_k^{\text{ms},\omega_i}$ in each coarse neighborhood ω_i. Moreover, we have $\dim(V_{\text{ms}}) \ll \dim(V_h)$.

- Step 2: Snapshot problem

  To construct the GMsFEM basis functions, we first construct a snapshot space $V_{\text{snap}}^{\omega_i}$ spanned by local snapshot basis functions $\phi_k^{\text{snap},\omega_i}$ for each coarse neighborhood ω_i. The snapshot basis function $\phi_k^{\text{snap},\omega_i}$ is the solution to the local problem

$$\begin{aligned} -\text{div}(\kappa\nabla\phi_k^{\text{snap},\omega_i}) &= 0, \quad \text{in } \omega_i, \\ \phi_k^{\text{snap},\omega_i} &= \delta_k^i, \quad \text{on } \partial\omega_i. \end{aligned} \tag{2.7}$$

  The fine grid functions δ_k^i is a function defined for $\{x_s \mid x_s \in \partial\omega_i\}$ which denotes the fine degrees of freedom on the boundary of ω_i. In specific,

$$\delta_k^i(x_s) = \begin{cases} 1, & \text{if } s = k, \\ 0, & \text{if } s \neq k. \end{cases}$$

  The linear span of these harmonic extensions forms the local snapshot space

$$V_{\text{snap}}^{\omega_i} := \operatorname*{span}_k\{\phi_k^{\text{snap},\omega_i}\}. \tag{2.8}$$

  One can also use randomized boundary conditions to reduce the computational cost associated with snapshot calculations [49].

- Step 3: Spectral problem

  Next, an analysis-based spectral problem is used to further reduce the dimension of the local multiscale space. More precisely, we seek for eigenvalues $\lambda_k^{\omega_i}$ and corresponding eigenfunc-

tions $\psi_k^{\omega_i} \in V_{\text{snap}}^{\omega_i}$ satisfying

$$a_i(\psi_k^{\omega_i}, v) = \lambda_k^{\omega_i} s_i(\psi_k^{\omega_i}, v), \quad \forall v \in V_{\text{snap}}^{\omega_i}, \tag{2.9}$$

where the bilinear forms in the spectral problem are defined as

$$\begin{aligned} a_i(u, v) &= \int_{\omega_i} \kappa \nabla u \cdot \nabla v, \\ s_i(u, v) &= \int_{\omega_i} \tilde{\kappa} u v, \end{aligned} \tag{2.10}$$

and $\tilde{\kappa} = \sum_j \kappa |\nabla \chi_j|^2$, while χ_j denotes the multiscale partition of unity function for each coarse block $K_j \subset \omega_i$. We arrange the eigenvalues $\lambda_k^{\omega_i}$ of the spectral problem (2.9) in ascending order, and select the first L_{ω_i} eigenfunctions $\{\psi_k^{\omega_i}\}_{k=1}^{L_{\omega_i}}$ corresponding to the small eigenvalues as the multiscale basis functions.

An alternative way to construct the multiscale basis function is using the idea of simplified basis functions. This approach assumes the number of channels and positions of the channalized permeability field are known. Therefore, we can obtain multiscale basis functions $\{\psi_k^{\omega_i}\}_{k=1}^{L_{\omega_i}}$ using this information without solving the spectral problem [37].

2.2.2 Construction of Basis Functions for NLMC

In the NLMC approach, the multiscale basis functions are selected such that the degrees of freedom have physical meanings and correspond to average solutions[50]. This method derives its foundation from CEM-GMsFEM [38], and starts with the definition of the auxiliary space. The idea here is to use a constant as auxiliary basis for the matrix in each coarse block, and another constant for each separate fracture network within the block. The simplified auxiliary space uses only essential degrees of freedom in each continua, thus one can obtain an upscaled equation with a minimal size where each degree of freedom represents the average of the solution over each continua.

In this section, we describe NLMC in detail following [37]. We consider the time dependent

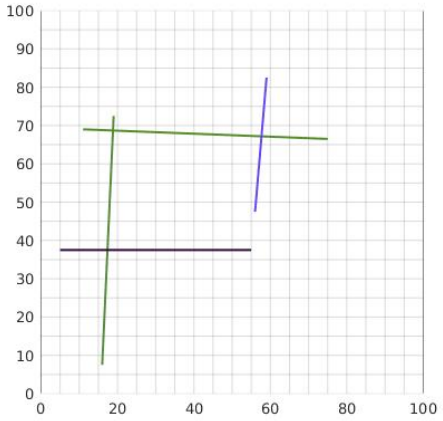


Figure 2.3: Example of a fractured media.

flow equation (2.2) in a fractured media (see Figure 2.3 for an illustration).

- Step 1: Partition of the domain

$$\Omega = \Omega_M \bigoplus_s d_s \Omega_{F,s}. \tag{2.11}$$

where M and F correspond to matrix and fracture respectively, and d_s is the aperture of fracture $\Omega_{F,s}$. Denoted by $\kappa(x) = \kappa_m$ the permeability in the matrix, and $\kappa(x) = \kappa_s$ the permeability in the s-th fracture. The permeabilities of matrix and fractures can differ by orders of magnitude.

Assume $\mathcal{T}^H$ is a coarse-grid partition of the domain Ω with a mesh size H which is further refined into a fine mesh $\mathcal{T}^h$ (see Fig3re 2.4 for an illustration of the fine and coarse mesh, where coarse elements are blue rectangles and fine elements are unstructured black triangles). Denoted by $\{K_i| \quad i = 1, \cdots, N_c\}$ the set of coarse elements in $\mathcal{T}^H$, where N_c is the number of coarse blocks. We also define the oversampled region K_i^+ for each K_i, with a few layers of neighboring coarse blocks, see Figure 2.4 for the illustration of K_i and K_i^+. We further define the set of all fracture segments within the coarse block K_j as $F^j = \{f_s^{(j)} | 1 \leq n \leq L_j\} = \{\cup_s \Omega_{F,s}\} \cap K_j$ where $L_j = dim(F^j)$.

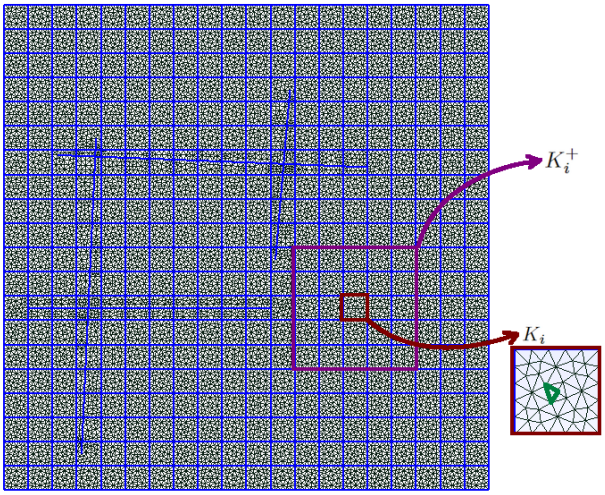


Figure 2.4: Illustration of coarse and fine meshes.

- Step 2: Computation of local basis functions in K_i^+.

  The basis for each over-sampled region $\psi_m^{(i)}$ solves the following local constraint minimizing problem on the fine grid

$$
\begin{aligned}
&a(\psi_m^{(i)}, v) + \sum_{K_j \subset K_i^+} \left(\mu_0^{(j)} \int_{K_j} v + \sum_{1 \le s \le L_j} \mu_s^{(j)} \int_{f_s^{(j)}} v \right) = 0, \quad \forall v \in V_0(K_i^+), \\
&\int_{K_j} \psi_m^{(i)} = \delta_{ij}\delta_{0m}, \quad \forall K_j \subset K_i^+, \\
&\int_{f_n^{(j)}} \psi_m^{(i)} = \delta_{ij}\delta_{nm}, \quad \forall f_n^{(j)} \in F^{(j)}, \ \forall K_j \subset K_i^+,
\end{aligned}
\tag{2.12}
$$

  where $a(u,v) := \int_{\Omega_M} \kappa_m \nabla u \cdot \nabla v + \sum_s \int_{\Omega_{F,s}} \kappa_s \nabla_f u \cdot \nabla_f v$. $\mu_0^{(j)}$ and $\mu_s^{(j)}$ are Lagrange multipliers while $V_0(K_i^+) := \{v \in V(K_i^+) | v = 0 \text{ on } \partial K_i^+\}$ and $V(K_i^+)$ is the fine grid space over an over-sampled region K_i^+. By this way of construction, the average of the basis $\psi_0^{(i)}$ equals 1 in the matrix part of the coarse element K_i, and equals 0 in other parts of the coarse blocks, $K_j \subset K_i^+$ as well as any fracture inside K_i^+. As for $\psi_m^{(i)}, m \ge 1$, it has an average of 1 on the m-th fracture continua inside the coarse element K_i, and an average of 0 in other fracture continua as well as the matrix continua of any coarse block $K_j \subset K_i^+$. It

indicates that the basis functions separate the matrix and fractures, and each basis represents one continua.

2.2.3 Multiscale Approximation Space

Once the multiscale basis functions $\{\psi_k^{\omega_i}\}$ are constructed, the span of the multiscale basis functions will form the offline space V_{ms}

$$\begin{aligned} V_{ms}^{(i)} &= \text{span}\{\psi_k^{\omega_i}\}_{k=1}^{L_{\omega_i}}, \\ V_{ms} &= \oplus_i V_{ms}^{(i)}. \end{aligned} \tag{2.13}$$

The multiscale solution $u_{\text{ms}} \in V_{\text{ms}}$ is then defined such that it satisfies

$$a(u_{\text{ms}}, v) = (g, v), \quad \text{for all } v \in V_{\text{ms}}, \tag{2.14}$$

or

$$(\frac{\partial u_{\text{ms}}}{\partial t}, v) + a(u_{\text{ms}}, v) = (g, v), \quad \text{for all } v \in V_{\text{ms}}. \tag{2.15}$$

These multiscale approximation spaces are justified by their construction such that they sustain the fine-level information while having reduced dimensions. Therefore, the multiscale solutions can be accurately approximated in such space with fewer degrees of freedom. However, difficulties arise in situations with uncertainties in the media properties in some local regions. To quantify the uncertainties, one needs to sample realizations of media properties and construct a distinct approximation spaces for each realization. The computational cost can thus grow very huge. To this end, building a functional relationship between the media properties and the multiscale model in an offline stage can avoid repeating computations and thus vastly reduce the computational complexity. Cases are common in which one needs to construct a nonlinear map when building a multiscale model. Modelling such a relationship typically involves high-order approximations. Therefore, it is natural to use machine learning techniques to derive such complex models. In particular, we adopt deep learning to facilitate the construction.

2.3 Deep Learning

As discussed in Section 1.1, deep learning techniques are introduced to construct data-aware multiscale models by using a training data set that is constituted by both simulation and observation data. Therefore, the network will learn an interpolated map from the existing data sampled from both models. Investigations have also been made in using deep learning for fast computation of multiscale basis [51]. Using the existing data, we can learn the complicated forward map between heterogeneous coefficients and the corresponding basis. Thanks to the ability of the neural networks to generalize, predictions of basis functions from multiscale coefficients become effortless once the network is well-trained.

In either case, when adopting deep learning in multiscale problems, we take advantage of the neural network that it is able to express complicated maps. This is guaranteed by its unique structure together with supporting algorithms for optimization. In this section, we will focus on these aspects of the neural networks and provide a general procedure to construct and tune a general deep neural network. Our later discussion will be based on the general cases provided in this section.

Specifically, if we are given samples $\{(x_i, y_i)\}_{i=1}^{L}$ from the map $\mathcal{F} : X \to Y$, i.e., $\mathcal{F}(x_i) = y_i$ for $1 \leq i \leq L$, and would like to learn the map from existing data and further predict the values of $\mathcal{F}(x_i)$ for $i = L+1, \cdots, L+M$, we first reformulate this problem as an optimization problem with the help of neural network. The optimization takes $\{(x_i, y_i)\}_{i=1}^{L}$ as training samples and produces a proper network coefficient θ^* starting from some initialization chosen at random such that $\mathcal{NN}(\cdot;\theta^*) \approx \mathcal{F}(\cdot)$. Moreover, if the neural network $\mathcal{NN}(\cdot)$ has a feed-forward fully-connected structure, then

$$\mathcal{NN}(x;\theta) := W_n\sigma(\cdots\sigma(W_2\sigma(W_1x + b_1) + b_2)\cdots) + b_n. \tag{2.16}$$

Here, θ represents all tuneable coefficients in the neural network $\mathcal{NN}(\cdot)$ and $\sigma(\cdot)$ is some nonlinear activation function. There are many choices of such nonlinear functions [52], while the most

common ones used are ReLU and $\tanh$.

For the network $\mathcal{NN}(\cdot)$ defined above, we will use the output $\mathcal{NN}(x_i)$ to approximate the desired output y_i. The difference between them will be measured using a cost function $\mathcal{C}(\cdot)$. For example, we can take the mean square error as the loss function:

$$\mathcal{C}(\theta) := \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathcal{NN}(x_i;\theta))^2, \tag{2.17}$$

which measures the average squared difference between the estimated values and the observed values. The neural network is then optimized by seeking θ^* to minimize the loss function, i.e.

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{C}(\theta). \tag{2.18}$$

Numerically, this optimization problem can be solved with a stochastic gradient descent (SGD) type method [53]. By calculating the gradient of the loss function, the coefficient θ is iteratively updated in an attempt to minimize $\mathcal{C}(\theta)$. This process is also referred to "training." Once the loss is minimized to a satisfactory small level, the neural network parameters θ^* is decided, and further, the overall neural network architecture $\mathcal{NN}(\cdot;\theta^*)$ is constructed. The predictions can then be given by $\mathcal{NN}(x_i;\theta^*)$ for $i = L+1, \cdots, L+M$.

In this dissertation, we not only utilize deep learning as a powerful tool to approximate sophisticated maps but also aim to understand why it works. We further develop the neural networks such that they are tailored to the targeted multiscale problems based on the underlying multiscale concepts. On the other hand, by deepening the understanding of the mechanism behind, we utilize neural networks to direct the model reduction in return. The in-depth discussions over these topics are presented in Chapter 5 and Chapter 6.

3. GENERALIZED MULTISCALE MULTICONTINUUM MODEL FOR FRACTURED VUGGY CARBONATE RESERVOIRS

In this chapter, a coarse solver is designed for flow simulations in a fractured and vuggy domain. We especially focus on the case that multiple discrete fractures locate in a single coarse neighborhood when MsFEM fails. Fractures and vugs are treated hierarchically with DFM and multicontinuum model. Highly developed fractures with only global effects are modeled as a fracture continuum, while fractures that have local effects are embedded as discrete fracture networks. For independent vugs, a continuum is used to represent their effects with specific configurations such that no intra-flow is considered. The heterogeneous media is then described as a coupled system of three continuum: matrix, fractures, and vugs. The system coupling DFM and three continuum is discretized spatially following GMsFEM to reduce the degrees of freedom while sustain its accuracy.

This chapter is organized as follows: In Section 3.1, the problem under discussion is clarified, followed by Section 3.2 which briefly reviews the multi-continuum model. In Section 3.3, a step-by-step illustration on GMsFEM together with a priori error estimate is provided. The details of time discretization of our problem is also discussed in this section. In Section 3.4, we present multiple numerical results to verify the effectiveness of the proposed methods. Lastly, this chapter is concluded by Section 3.5.

3.1 Problem Setting

In this chapter, we consider a 2-dimensional flow problem in a multiscale porous media. In specific, we consider a Darcy flow. For simplicity, we ignore the gravity and the capillary pressure effects.

3.1.1 Equation for Slightly Compressive Flow in Porous Media

In specific, we consider the following equation for slightly compressive flow in a heterogeneous porous media, we remark that this equation is just a special case of (2.1),

$$\frac{\phi c}{B^{\circ}}\frac{\partial u}{\partial t} - \frac{1}{\mu}\nabla \cdot (\frac{\kappa}{B}\nabla u) = g \quad \text{in } \Omega. \tag{3.1}$$

Here, Ω is the computational domain. c is compressibility and μ is viscosity of the liquid. B° is the formation volume factor (FVF) at reference pressure u^0 and B is a FVF at reservoir condition. They are used to quantify the compressibility of the target liquid. ϕ represents porosity of the fractured vuggy media, while κ is a permeability function that bears multiscale features (See Figure 3.1 for an illustration). The solution to be sought is pressure u, given a production rate g.

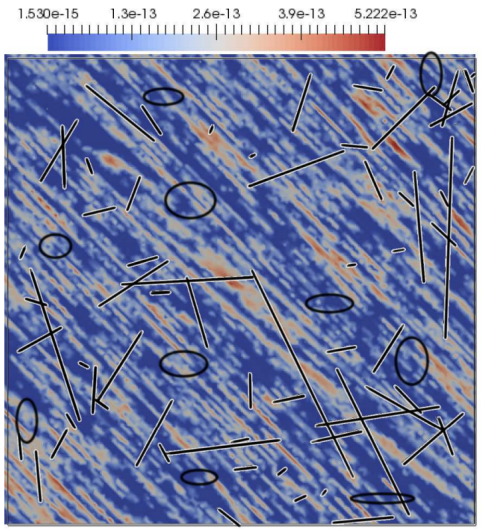


Figure 3.1: Permeability field $\kappa(x)$ with multiscale features.

Limiting our interests to slightly compressible liquid, we can further employ the simplified

correlation between the formation volume factor B and the pressure u that

$$B = \frac{B^\circ}{1 + c(u - u^0)} \tag{3.2}$$

to rewrite (3.1) and get

$$\frac{\phi c}{B^\circ}\frac{\partial u}{\partial t} - \frac{1}{\mu}\nabla \cdot (\kappa \frac{1 + c(u - u^0)}{B^\circ}\nabla u) = g, \quad \text{in } \Omega. \tag{3.3}$$

In the following sections, we will derive our method based on (3.3) along with Dirichlet or Neumann boundary conditions on $\partial\Omega$.

Throughout this chapter, we assume c, ϕ μ and B° are constants. (3.2) can then be reformulated as

$$b\frac{\partial u}{\partial t} - \nabla \cdot (\kappa(x)\alpha(u)\nabla u) = g \tag{3.4}$$

where

$$b = \frac{\phi c}{B^\circ}$$

is a constant, while

$$\alpha(u) = \frac{1}{\mu} \cdot (1 + c(u - u^0))$$

is a linear map in u.

3.1.2 Fine-scale Spatial Discretization

For flow in a fractured and vuggy media, the multiscale flow problem described in (3.4) becomes more complicated as the fractures and vugs have very different hydraulic properties from its background matrix. They can bring in extra transfer and storage mechanics to the flow. The fractures amplify the complexity of modeling as they can have a wide range of scales and topology. In order to delicately model the their effects on flow, we apply a hierarchical approach. Fractures that have only local effects can be resolved by fine mesh. Thus, the computational domain can be

partitioned as in (2.11)

$$\Omega = \Omega_M \bigoplus_s d_s \Omega_{F,s}.$$

Those fractures that have global effects and are not resolved by mesh can later be handled by representing them as one continua. So are the effects of vugs, which will be discussed in details in next section.

For a fine-scale approximation of u, we discretize the PDE on a fine grid, and apply Finite Element Method as well as the Discrete Fracture Model (DFM). Specifically, all integrations in the weak form of (3.4), will now be taken separately in both Ω_M and $\Omega_{F,s}$ with distinct hydraulic parameters. To compromise arbitrary fractures $\Omega_{F,s}$, one need to adopt an unstructured fine-scale mesh. The resulting semi-discrete numerical system is

$$\begin{aligned} &\int_{\Omega_M} b_M \frac{\partial u_h}{\partial t} v_h \, dx + \sum_s \int_{\Omega_{F,s}} b_{F,s} \frac{\partial u_h}{\partial t} v_h \, dx \\ &+ \int_{\Omega_M} \kappa_M(x) \alpha(p_h) \nabla u_h \nabla v_h \, dx + \sum_s \int_{\Omega_{F,s}} \kappa_{F,s}(x) \alpha(u_h) \nabla_F u_h \nabla_F v_h \, dx \qquad (3.5) \\ = &\int_\Omega g v_h \, dx, \quad \forall v_h \in V_h. \end{aligned}$$

Here, v_h is a standard FEM basis function. ∇_F means taking directional derivative along the degenerated 1-D fracture $\Omega_{F,i}$. Note that the aperture effects are considered in $\kappa_{F,s}(x)$. $b_M = \dfrac{\phi_M \, c}{B^\circ}$ and $b_{F,s} = \dfrac{\phi_{F,s} \, c}{B^\circ}$ are constants.

3.2 Multi-continuum Model

To explicitly represent the global effects of unresolved fractures, vugs and matrix, we introduce the multi-continuum methods. We consider the media as a coupled system of three parallel continua: matrix, unresolved fractures (usually natural fractures), and vugs. They coexist everywhere in our computational domain, while they interact with each other via mass transfer (see Figure 3.2 for an illustration). Without of loss of generality, we assume that all continuum interact with each other. If we denote the flow pressure for continua i as u^i, and write the interaction between continua i and j as $Q^{i,j}$, we can then establish a system of PDEs following (3.4) to describe the

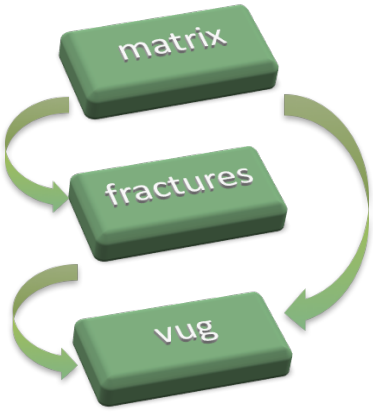


Figure 3.2: Illustration of triple-continuum model.

flow mechanism in each continua as:

$$b^i \frac{\partial u^i}{\partial t} - \nabla \cdot (\kappa^i(x)\alpha(u^i)\nabla u^i) = g^i - \sum_{j \neq i} Q^{i,j} \tag{3.6}$$

with $b^i = \dfrac{\phi^i c}{B^\circ}$. Here i can be m, f,or v which stands for matrix, unresolved fractures and vugs respectively. We further assume that there is no intra-flow inside the vugs and all vugs only act as a storage in this system. That is to say, we only consider the case when all vugs are independent from each other. Mass transfer due to inflow of liquid along vugs can be disregarded in any element of the domain. Therefore, flow equation for vugs can be written as

$$b^v \frac{\partial u^v}{\partial t} = g^v + Q^{f,v} + Q^{m,v}. \tag{3.7}$$

The term $Q^{i,j}$ represents the mass transfer from continua i to continua j. This transfer can be modeled using [54, 55]

$$Q^{i,j} = \sigma \frac{\kappa^{i,j}}{\mu}(u^i - u^j) = q^{i,j}(u^i - u^j),$$

where $q^{i,j} = q^{j,i}$. Here σ is a shape factor, and $\kappa^{i,j}$ is taken to be the harmonic mean of the permeability κ^i and κ^j.

With (3.6) and (3.7), we can derive the weak formulation of our proposed triple-continuum

system. For matrix and unresolved fractures, we have

$$\int_\Omega b^i \frac{\partial u^i}{\partial t} v^i \, dx + \int_\Omega \kappa^i(x)\alpha(u^i)\nabla u^i \nabla v^i \, dx + \sum_{j\neq i} \int_\Omega Q^{i,j} v^i \, dx = \int_\Omega g^i v^i dx, \quad i = m, f. \tag{3.8}$$

For vugs, we have

$$\int_\Omega b^v \frac{\partial u^v}{\partial t} v^v \, dx - \int_\Omega Q^{m,v} v^v \, dx - \int_\Omega Q^{f,v} v^v \, dx = \int_\Omega g^v v^v dx. \tag{3.9}$$

Here, v^i is any testing function in the same space as u^i. We mention that equation of u^m, u^f and u^v are coupled through term $Q^{i,j}$. Thus, this coupled system should be solved in a Cartesian product space $(u^m, u^f, u^v) \in V^m \times V^f \times V^v$. In our proposed approach, we take $V^i = H_0^1(\Omega)$ for all continuum.

To express the effects of both unresolved and resolved fractures on flow dynamics, we manage to incorporate DFM when solving this multicontinuum equation system (3.8)–(3.9). Like what we have in (3.5), we assume $\Omega_{F,s}$ corresponds to a 1-D domain that serves as a resolved fracture region. All integrations on Ω can thus be rewritten as $\int_{\Omega_m} + \sum_s \int_{\Omega_{F,s}}$. For example, (3.8) can be rewritten as

$$\begin{aligned} &\int_{\Omega_M} b^i \frac{\partial u^i}{\partial t} v^i \, dx + \sum_s \int_{\Omega_{F,s}} b_{F,s} \frac{\partial u^i}{\partial t} v^i \, dx \\ &+ \int_{\Omega_M} \kappa^i(x)\alpha(u^i)\nabla u^i \nabla v^i \, dx + \sum_s \int_{\Omega_{F,s}} \kappa_{F,s}(x)\alpha(u^i)\nabla_F u^i \nabla_F v^i \, dx \\ &+ \sum_{j\neq i} \int_\Omega Q^{i,j} v^i \, dx = \int_\Omega g^i v^i \, dx, \end{aligned} \tag{3.10}$$

for $i = m, f$, after applying DFM to its original form. Similarly, incorporating DFM in (3.9) yields

$$\int_{\Omega_M} b^v \frac{\partial u^v}{\partial t} v^v \, dx + \sum_s \int_{\Omega_{F,s}} b_{F,s} \frac{\partial u^v}{\partial t} v^v \, dx - \int_\Omega Q^{m,v} v^v \, dx - \int_\Omega Q^{f,v} v^v \, dx = \int_\Omega g^v v^v \, dx. \tag{3.11}$$

The fine-scale FEM solution (u^m, u^f, u^v) should be sought in $V_h = V_h^m \times V_h^f \times V_h^v$, where the V_h^i

is a conforming finite element space of the continuum i on a fine partition $\mathcal{T}^h$ of the domain. We also remark that the shape factor σ is taken to be proportional to h^{-2}.

For the purpose of simpler notations in the analysis presented in Appendix A.1, we rewrite the derived system (3.10)– (3.11) in a more general N-continuum setting. First, we denote the Sobolev space $V = [H_0^1(\Omega)]^N$. On each continuum, given a fixed $w^i \in H_0^1(\Omega)$, we define the bilinear forms:

$$\begin{aligned} b^i(u^i, v^i) &= \int_{\Omega_M} b^i u^i v^i \, dx + \sum_s \int_{\Omega_{F,s}} b_{F,s} u^i v^i \, dx, \\ a^i(u^i, v^i; w^i) &= \int_{\Omega_M} \kappa^i \alpha(w^i) \nabla u^i \cdot \nabla v^i \, dx + \sum_s \int_{\Omega_{F,s}} \kappa_{F,s} \alpha(w^i) \nabla_F u^i \cdot \nabla_F v^i \, dx. \end{aligned} \tag{3.12}$$

Given a fixed $w \in V$, we further define the following coupled bilinear forms on V

$$\begin{aligned} b(u, v) &= \sum_i b^i(u^i, v^i), \\ a(u, v; w) &= \sum_{1 \le i < N} a^i(u^i, v^i; w^i), \\ q(u, v) &= \sum_i \sum_{j \ne i} q^{i,j} \int_\Omega (u^i - u^j) v^i \, dx. \end{aligned} \tag{3.13}$$

Then the generalized weak formulation (3.10)–(3.11) can be written as: find $u = (u^1, u^2, \cdots, u^N)$, where $u(t, \cdot) \in V$, we have such that for all $v = (v^1, v^2, \cdots, v^N)$, where $v(t, \cdot) \in V$,

$$b\left(\frac{\partial u}{\partial t}, v\right) + a(u, v; u) + q(u, v) = (g, v), \qquad t \in (0, T). \tag{3.14}$$

Specifically, for $N = 3$ case in this paper, the continuum indices representing the matrix, fractures and vugs components in order.

3.3 GMsFEM

In order to reduce the computational cost, we would like to solve the equation system (3.6) and (3.7) on a coarse mesh. However, permeability coefficient $\kappa(x)$ is heterogeneous in space, thus a

standard FEM solution on coarse mesh will be inaccurate as it loses subgrid information. Therefore, we use GMsFEM to construct multiscale basis that contains local heterogeneous permeability information. By replacing the standard FEM basis with GMsFEM basis, we are able to obtain a better accuracy and sustain an affordable computational cost.

In this section, we briefly review the procedure for GMsFEM. Roughly speaking, the construction of GMsFEM basis consists of two stages: solving snapshot problems and conducting spectral decomposition. Both steps are performed locally as discussed in Section 2.2.1. The fine and coarse partition of the computational domain Ω is same as in 2.2.1. We remark that the fine mesh should be taken in surrendering the discrete fracture network (see Figure 3.3b for an illustration).

3.3.1 Snapshot Space

Recall that the coarse neighborhood ω_i of node x_i on a coarse mesh $\mathcal{T}^H$ is defined as:

$$\omega_i = \bigcup_j \{K_j \in \mathcal{T}^H \,|x_i \in \bar{K}_j\}.$$

A snapshot space is an auxiliary space constructed within each coarse neighborhood ω_i. We omit the subscript i for simplicity. There are a few different ways to construct snapshot space [33]. In this chapter, we take solutions to the following coupled harmonic extension problems as the snapshot basis functions.

The snapshot problems are designed analogue to the steady state equation of (3.6) and (3.7). We consider a coupled snapshot system in a coarse neighborhood ω, in which we find

$$\phi_{k,s}^{\text{snap},\omega} = \left(\phi_{k,s}^{1,\text{snap},\omega}, \phi_{k,s}^{2,\text{snap},\omega}, \cdots, \phi_{k,s}^{N,\text{snap},\omega}\right) \in V_h(\omega)$$

such that

$$
\begin{aligned}
-\nabla\cdot\frac{\kappa^i(x)}{\mu}\nabla\phi_{k,s}^{i,\text{snap},\omega}+\sum_{j\neq i}q^{i,j}(\phi_{k,s}^{i,\text{snap},\omega}-\phi_{k,s}^{j,\text{snap},\omega})=0,\quad &\text{in }\omega\quad i=1,2,\cdots,N-1,\\
\sum_{j\neq v}q^{i,j}(\phi_{k,s}^{i,\text{snap},\omega}-\phi_{k,s}^{j,\text{snap},\omega})=0,\quad &\text{in }\omega,\quad i=N,\\
\phi_{k,s}^{\text{snap},\omega}=\delta_{k,s},\quad &\text{on }\partial\omega.
\end{aligned}
\tag{3.15}
$$

$\delta_{k,s}$ is defined on all fine-scale nodes of $\partial\omega$. If the set $\{x_i^\omega|1\leq i\leq N_v^\omega\}$ represents all fine-scale nodes on boundary, we have

$$
\delta_{k,s}(x_i^\omega)=\begin{cases}e_s & i=k,\\ 0 & i\neq k.\end{cases}
$$

Here, $\{e_s\}_{s=1}^N$ are standard basis in $\mathbb{R}^N$. So far, we construct the local snapshot space as:

$$
V_{\text{snap}}^\omega=\text{span}\{\phi_{k,s}^{\text{snap},\omega}\mid 1\leq k\leq N_v^\omega,1\leq s\leq N\}.
$$

The global snapshot space is defined as the sum of all local snapshot spaces, i.e.

$$
V_{\text{snap}}=\text{span}\{\phi_{k,s}^{\text{snap},\omega_i}\mid 1\leq i\leq N_v,1\leq k\leq N_v^{\omega_i},1\leq s\leq N\}.
$$

Remark When solving local snapshot problem (3.15) on the fine mesh within ω, one should also apply the idea of DFM and replace all integral $\int_\omega$ by $\int_{\omega_M}+\sum_s\int_{\omega_{F,s}}$ and all coefficients correspondingly.

3.3.2 Spectral Problem

To further reduce the dimension of the resulting system, we conduct a spectral decomposition on V_{snap}^ω. More precisely, we sought eigenpairs $(\lambda_m^\omega,\psi_m^\omega)\in\mathbb{R}\times V_{\text{snap}}^\omega$ for the following local spectral problem

$$
a_\omega(\psi_k^\omega,v)=\lambda_k^\omega s_\omega(\psi_k^\omega,v),\quad\forall v\in V_{\text{snap}}^\omega,
\tag{3.16}
$$

where

$$a_\omega(u,v) = \sum_{i\in\{m,f\}} a^i_\omega(u^i,v^i) + \sum_i \sum_{j\neq i} \int_\omega q^{i,j}(u^i-u^j)v^i\,dx,$$
$$s_\omega(u,v) = \frac{1}{\mu}\sum_i \int_\omega \kappa^i(x)u^iv^i\,dx.$$

The form of $a_\omega(u,v)$ and $s_\omega(u,v)$ are inspired by analysis which will be demonstrated in next section along with Appendix A.1. We sort the eigenvalues $\{\lambda_k^\omega\}$ of (3.16) in ascending order, and we take the first L_ω eigenfunctions $\psi_k^\omega = (\psi_k^{1,\omega}, \psi_k^{2,\omega}, \cdots, \psi_k^{N,\omega})$. Then the k-th multiscale basis function $\psi_k^{\text{ms},\omega} = (\psi_k^{1,\text{ms},\omega}, \psi_k^{2,\text{ms},\omega}, \cdots, \psi_k^{N,\text{ms},\omega})$ in ω is defined by

$$\psi_k^{i,\text{ms},\omega} = \chi^\omega \psi_k^{i,\omega}, \quad i = 1,2,\cdots,N,$$

where χ^ω is a partition of unity function for coarse grid $\mathcal{T}^H$ on a coarse neighborhood ω. By multiplying χ^ω, we obtained a set of conforming multiscale basis functions supported in ω. Using the multiscale basis functions $\{\psi_k^{\text{ms},\omega_i}\}$ for all coarse regions ω_i, we construct the multiscale space

$$V_{\text{ms}} = \text{span}\{\psi_k^{\text{ms},\omega_i} \mid 1 \le k \le L_{\omega_i}, 1 \le i \le N_v\}.$$

We remark that $dim(V_{\text{ms}}) \ll dim(V_h)$, where $V_h = [V_h^i]^N$ is the standard FEM approximation space on $\mathcal{T}^h$. When the multiscale space is established, we can then find a coarse-scale solution on V_{ms} with less computational effort.

Once the multiscale space is constructed, the general GMsFEM solution is given by: find $u_{\text{ms}} = (u_{\text{ms}}^1, \cdots, u_{\text{ms}}^N)$, where $u_{\text{ms}}(t,\cdot) \in V_{\text{ms}}$, such that for all $v = (v^1, v^2, \cdots, v^N)$, where $v(t,\cdot) \in V_{\text{ms}}$,

$$b\left(\frac{\partial u_{\text{ms}}}{\partial t}, v\right) + a(u_{\text{ms}}, v; u_{\text{ms}}) + q(u_{\text{ms}}, v) = (f,v), \quad t \in (0,T). \tag{3.17}$$

3.3.3 A-priori Error Estimates

In this section, we present some a-priori error estimates of the semi-discrete problem. The proofs of these estimates will be left to Appendix A.1.

We suppose κ has a upper bound κ^+ and a lower bound κ^- in Ω. We further assume that $\alpha(u^i)$ and $\alpha(u^i_{\text{ms}})$ has a uniform upper bound α^+ and a uniform lower bound α^-, i.e.

$$0 < \alpha^- \leq \alpha(u^i), \alpha(u^i_{\text{ms}}) \leq \alpha^+. \tag{3.18}$$

Next, we introduce some metrics on V. The bilinear form $b(\cdot,\cdot)$ can further induce a norm

$$\|u\|_b := (b(u,u))^{1/2}.$$

We also define a norm $\|\cdot\|_{a_Q}$ by

$$\|u\|_{a_Q} := (|u|_a^2 + |u|_q^2)^{\frac{1}{2}}, \tag{3.19}$$

where $|u|_a^2 := \sum_{1\leq i<N} \left(\int_{\Omega_M} \kappa^i |\nabla u^i|^2 \, dx + \sum_s \int_{\Omega_{F,s}} \kappa_{F,s} |\nabla_F u^i|^2 \, dx \right)$ and $|u|_q^2 := q(u,u)$.

The first theorem provides an estimate of the error between the weak solution u and the multiscale solution u_{ms} by the projection error of u onto the multiscale space V_{ms} in various metrics.

Theorem 3.3.1. *Let u be the weak solution in* (3.14) *and u_{ms} be the multiscale numerical solution in* (3.17)*. Assume $\nabla u \in L^4(\Omega_M)$ and $\nabla_F u \in L^2(\Omega_{F,s})$. Then we have*

$$\begin{aligned} \|u(t,\cdot) - u_{ms}(t,\cdot)\|_b^2 + \int_0^T \|u - u_{ms}\|_{a_Q}^2 \, dt \\ \leq C \inf_{w\in V_{ms}} \Big(\int_0^T \|\frac{\partial(w-u)}{\partial t}\|_b^2 \, dt + \int_0^T \|w-u\|_{a_Q}^2 \, dt + \|w(0,\cdot) - u(0,\cdot)\|_b^2 \Big). \end{aligned} \tag{3.20}$$

In light of Theorem 3.3.1, we have to establish an estimate of the projection error of u onto

the multiscale space V_{ms} on the right hand side of (3.20), in order to complete the convergence analysis. With the assumption that the irreducible error between the Sobolev space V and the snapshot space V_{snap} is small, which holds when a sufficiently large number of snapshot solutions is taken, we define an approximation $u_{\text{snap}}(\cdot,t) \in V_{\text{snap}}$ of $u(\cdot,t)$ in the snapshot space by

$$u_{\text{snap}}(x,t) = \sum_{i=1}^{N_v} \sum_{k=1}^{N_v^{\omega_i}} \sum_{s=1}^{N} u(x_k,t)\chi^{\omega_i}(x_k)\phi_{k,s}^{\text{snap},\omega_i}(x), \tag{3.21}$$

and provide an estimate of the projection error of u_{snap} onto the snapshot space V_{ms}.

Theorem 3.3.2. *Let u and u_{snap} be the reference solution and the snapshot projection of u as defined in* (3.14) *and* (3.21)*. We then have*

$$\begin{aligned}\inf_{w\in V_{ms}} \int_0^T \|\frac{\partial(w-u_{snap})}{\partial t}\|_b^2\, dt + \int_0^T \|w-u_{snap}\|_{a_Q}^2\, dt + \|w(0,\cdot)-u_{snap}(0,\cdot)\|_b^2 \\ \leq \frac{C}{\Lambda}(\int_0^T \|\frac{\partial u}{\partial t}\|_{a_Q}^2\, dt + \int_0^T \|u\|_{a_Q}^2\, dt + \|u(0,\cdot)\|_{a_Q}^2)\end{aligned} \tag{3.22}$$

with $\Lambda = \min_j\{\lambda_{L_{\omega_j}+1}^{\omega_j}\}$.

3.3.4 An Implementation View

In this section, we derive the fully discrete system and present the implementation details. We adopt the implicit Euler scheme for time discretization to the semi-discrete GMsFEM system (3.17). Suppose the time domain $(0,T)$ is partitioned into equal subintervals of length Δt, and denote the n-th time instant by $t_n = n\Delta t$. Using backward difference, the fully discrete GMsFEM scheme is to, successively for $n = 1, 2, \ldots$, find $u_{\text{ms}}^n \in V_{\text{ms}}$ such that

$$b\left(\frac{u_{\text{ms}}^n - u_{\text{ms}}^{n-1}}{\Delta t}, v\right) + a(u_{\text{ms}}^n, v; u_{\text{ms}}^n) + q(u_{\text{ms}}^n, v) = (g^n, v), \quad \forall v \in V_{\text{ms}}, \tag{3.23}$$

where the subscript n denotes the evaluation of a time-dependent function at the time instant t_n and for an given initial condition u_{ms}^0. At each time instant t_n, (3.23) gives rise to a nonlinear algebraic system with respect to the multiscale basis functions. With a sufficiently small time step size, we

can adopt a direct linearization approach by replacing the field $\alpha(u_{\text{ms}}^n)$ by $\alpha(u_{\text{ms}}^{n-1})$ and derive

$$b\left(\frac{u_{\text{ms}}^n - u_{\text{ms}}^{n-1}}{\Delta t}, v\right) + a(u_{\text{ms}}^n, v; u_{\text{ms}}^{n-1}) + q(u_{\text{ms}}^n, v) = (g^n, v), \quad \forall v \in V_{\text{ms}}. \tag{3.24}$$

Alternatively, we can use an iterative approach. More precisely, we can construct a sequence $\{u_{\text{ms},m}^n\}_{m=0}^{\infty} \subset V_{\text{ms}}$ whose fixed point is the solution u_{ms}^n and truncate the successive iterations when a stopping criterion is fulfilled. In this case, we start with an initial guess $u_{\text{ms},0}^n = u_{\text{ms}}^{n-1}$ and solve for

$$b\left(\frac{u_{\text{ms},m}^n - u_{\text{ms}}^{n-1}}{\Delta t}, v\right) + a(u_{\text{ms},m}^n, v; u_{\text{ms},m-1}^n) + q(u_{\text{ms},m}^n, v) = (g^n, v), \forall v \in V_{\text{ms}}. \tag{3.25}$$

We remark that it is equivalent to the linearization approach if we stop after one iteration.

3.4 Numerical Results

In this section, we apply our proposed methods to a realistic fractured and vuggy reservoir. All three continuum have heterogeneous permeability background (see Figure 3.1 for the permeability of matrix) and discrete fracture networks are embeded in this reservoir like in Figure 3.3a. An unstructured fine mesh is used to resolve the discrete fractures networks (see Figure 3.3b). The descriptive parameters of this reservoir are listed in Table 3.1. All numerical results are implemented using FEniCS Library.

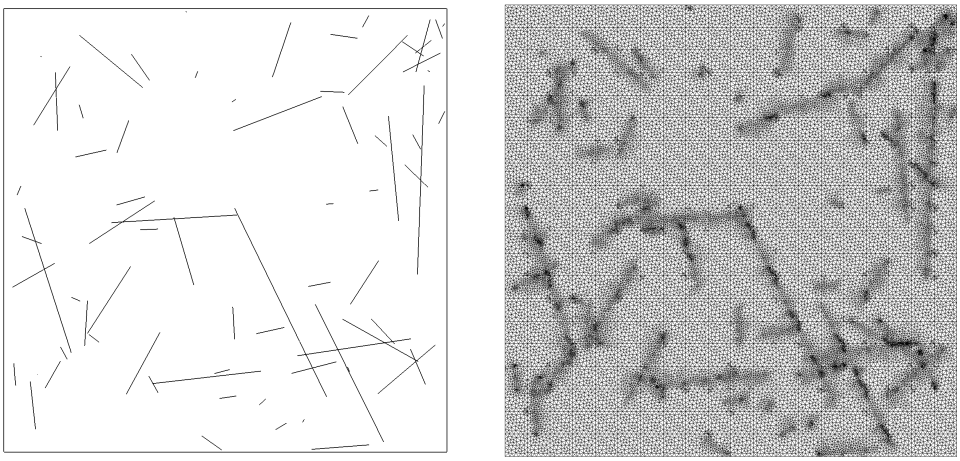
The numerical experiments are conducted from different aspects. Performances are compared between MsFEM and GMsFEM, nonzero source term and nonzero mixed boundary condition. We also discuss the impact of the number of basis functions selected to the solution accuracy. We remark that all examples are conducted using direct linearization approach as the iterative approach do not significantly improve the results for our problem.

3.4.1 Comparison of MsFEM and GMsFEM

In this subsection, we discuss the necessity to apply GMsFEM. From Figure 3.4, we can tell that, even with similar number of degrees of freedom, the MsFEM is not able to recover the true

Quantity	Value
Size of model	$15000ft \times 15000ft$
ϕ^m	0.2
ϕ^f	0.01
ϕ^v	0.1
$\phi_{F,j}$	1
κ^f	10^{-12}
κ^v	10^{-13}
$\kappa_{F,j}$	8.2606×10^{-8}
c	1.4504×10^{-8}
μ	8×10^{-3}
u^0	2.0684×10^7
B°	1.1
δ	$1/h^2_{\text{min}}$

Table 3.1: Values of all quantities.



(a) Idealized discrete fracture network(DFN)

(b) Unstructured fine mesh

Figure 3.3: DFN and fine scale mesh.

solution, thus GMsFEM must be applied to generate meaningful results. This is especially true when there are multiple discrete fracture networks coexist in a single coarse neighborhood. Many numerical experiments have shown that MsFEM basis functions are not able to handle homogeneous background and multiple discrete fracture networks simultaneously. Figure 3.4 shows the

solution we obtained using MsFEM and GMsFEM respectively when a single source is placed at the bottom left corner. The relative error of MsFEM solution can be as large as 30%.

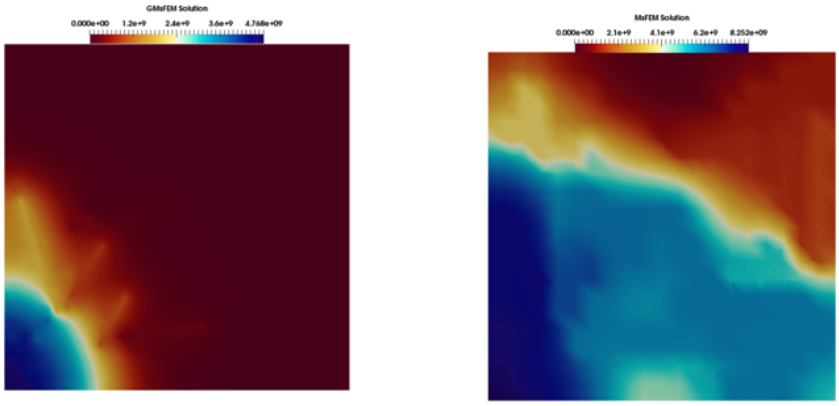


Figure 3.4: Comparison between GMsFEM and MsFEM solution with heterogeneous background and discrete fracture network. Left: GMsFEM solution with DOF=2646. Right MsFEM solution with DOF =2400.

3.4.2 GMsFEM Solution for Different Boundary Condition and Source

In this subsection, we demonstrate the performance of our proposed triple continuum GMsFEM methods applied to problem (3.6) and (3.7), where lagging coefficient scheme is used to linearize the problem. Different boundary conditions and source term settings are tested for coupled GMsFEM approach.

Number of Basis	Day 1	Day 10	Day 20
2	17.21	27.22	66.44
4	14.88	17.27	43.65
8	4.72	11.86	13.31
16	4.24	12.05	12.58

Table 3.2: L^2 relative errors(%) of numerical results for mixed boundary condition. Nonzero Dirichlet boundary condition is imposed on top and bottom boundary. Zero Neumann boundary is applied to left and right boundary.

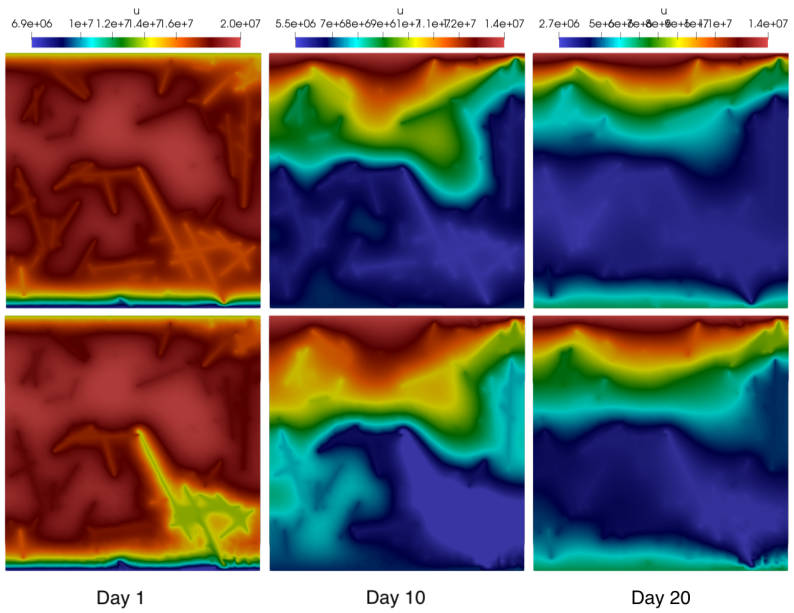


Figure 3.5: Triple-Continuum, heterogeneous background flow simulation of matrix with top and bottom nonzero Dirichlet boundary condition. Zero Neumann boundary is applied to left and right boundary. First row: fine-scale reference solution, DOF = 80229. Second row: Coupled coarse-scale GMsFEM solution with 8 basis, DOF = 3528.

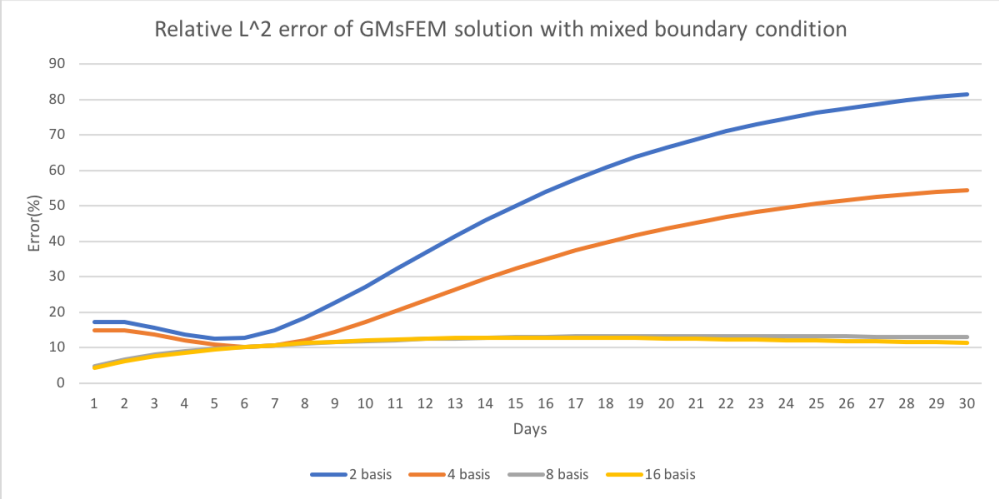


Figure 3.6: Illustration of error trend with time for different number of basis for dirichlet boundary condition case.

From both error tables and solution figures , we come to the conclusion that: 1) For nonzero

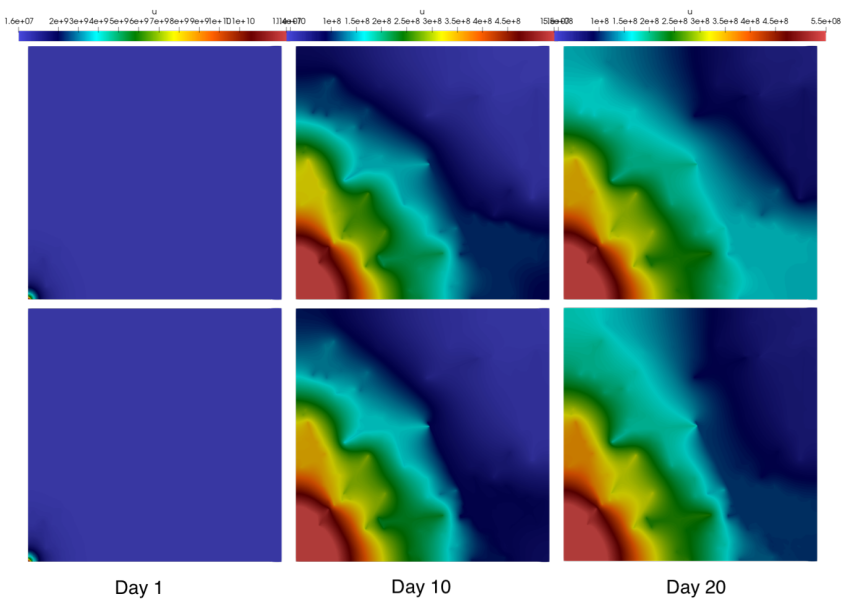


Figure 3.7: Flow simulation results for a triple continuum heterogeneous background matrix with no flow boundary condition. Injector locates at bottom left corner. First row : Fine-scale reference solution. Second row: Coupled coarse-scale GMsFEM solution. DOF is same as in Figure 3.5.

Number of Basis	Day 1	Day 10	Day 20
2	15.79	10.05	11.42
4	5.48	5.89	8.53
8	2.84	6.20	8.51
16	1.12	6.30	8.49

Table 3.3: L^2 relative errors(%) of numerical results for zero Neumann boundary condition.

mixed boundary condition case, the GMsFEM solution can obtain a good result when using 8 basis or more. 2) For zero Neumann boundary and single point source term case, the coupled approach can obtain good approximation of fine-scale solution with 4 basis or more. 3) For both cases, the coupled approach can give us an acceptable solution.

From Figure 3.8 , Figure 3.6, Table 3.3 and Table 3.2, we can tell that the error of solution decreases when we increase the number of eigenfunctions used.

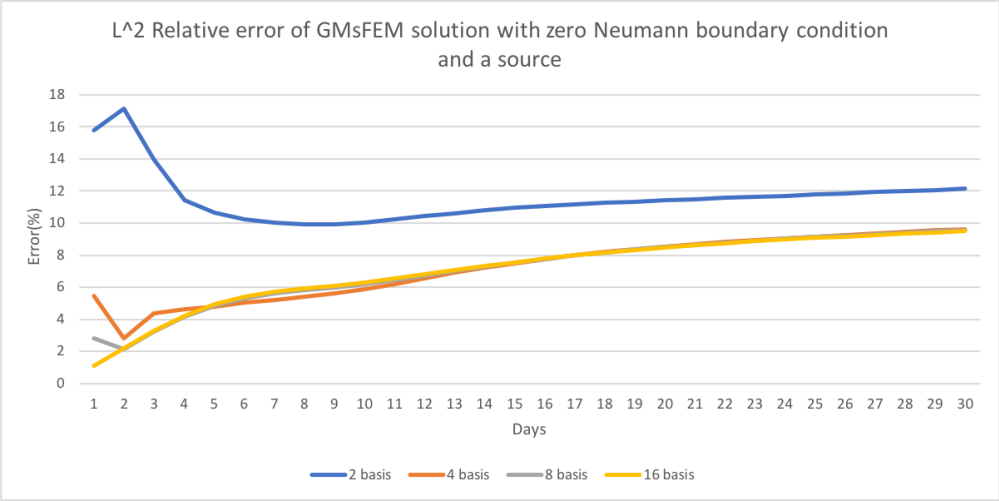


Figure 3.8: Illustration of error trend with time for different number of basis for single source case.

3.5 Conclusion

In this chapter, we proposed a triple continuum GMsFEM method as a fast solver of flow problems in a heterogeneous domain. A fractured and vuggy reservoir is modeled as a system of three continuum. Coupled assembling is provided to construct GMsFEM multiscale space. The convergence of our proposed method is proved strictly following mild assumptions. Later, the performance is tested using multiple examples with different settings. We conclude that with a GMsFEM framework, a multicontinuum model, and the discrete fracture network, our proposed method can inherit the merits of the three. By coupling them together, we improved the capability as well as accuracy of our simulation and obtained competitive approximations for both mixed boundary conditions and a single source case.

In short, we claim that our proposed method can accomplish the flow simulation task with both accuracy and efficiency. Nevertheless, we notice that our proposed method is only good for the case when discrete fracture network is known. For reservoirs containing uncertainties, further exploration is desired. Besides, for vugs with turbulent flow inside, one will end up with a coupled PDE system containing Navier-Stokes equation. Future investigations are required to expand our work to such cases.

4. GMSFEM FOR MIXED ELLIPTIC EQUATION IN A NARROW DOMAIN

When simulating flow in capillary or fissured systems, homogenization method is commonly used. However, when there is no scale separation of the media, for example, modeling blood circulation within a nonuniform vessel, using only one basis function will result in imprecise solutions due to the loss of local multiscale information [56]. Thus, we would like to address such problems using GMsFEM instead. By constructing multiscale basis functions, we pass microscopic information from the local system to the global system instead of averaging it out. Moreover, by the design of GMsFEM, we are able to construct an approximation space with rich multiscale information where the leading basis correspond to the dominant feature modes.

Additionally, we want to study how the geometry of the computational domain influences the approximation accuracy of our proposed method through rigorous analysis.

4.1 Problem Setting

In this chapter, we aim to solve a mixed elliptic equation in a narrow domain, especially for the case when a single basis in each coarse neighborhood is not enough to represent the underlying multiscale features of the domain.

4.1.1 Mixed Formulation of Flow Equation

We consider the steady state flow equation (2.1),

$$-\text{div}\big(\kappa\,\nabla u\big) = g, \quad \text{in } \Omega.$$

Here, κ represents the heterogeneous permeability of the medium, u is pressure, Ω is a narrow domain with large length/width ratio and g is a source or sink function. If we assume the flow is governed by Darcy's Law, and the viscosity of the target liquid is 1, then the flux σ satisfies

$$\sigma = -\kappa\nabla u. \tag{4.1}$$

Now we introduce the new variable σ (which is vector-valued), we have the mixed formulation of the above PDE:

$$\kappa^{-1}\sigma + \nabla u = 0, \tag{4.2a}$$

$$\nabla \cdot \sigma = g. \tag{4.2b}$$

More specifically, Dirichlet Boundary condition is imposed to pressure u for left and right boundary while a no flow boundaries condition is imposed to flux σ on both top and bottom boundaries, i.e.

$$u = -1, \quad \text{on } \partial\Omega_1, \tag{4.3a}$$

$$u = 1, \quad \text{on } \partial\Omega_3, \tag{4.3b}$$

$$\sigma \cdot \mathbf{n} = 0, \quad \text{on } \partial\Omega_{2,4}. \tag{4.3c}$$

Here, $\mathbf{n}$ denotes the outward normal vector on $\Omega_{2,4}$. For source term g, we let:

$$g = 0. \tag{4.3d}$$

We will derive GMsFEM solution based on (4.2) for the rest of this chapter.

4.1.2 Variational Forms and Fine-scale Discretization

We first derive the weak form of the system (4.2). After multiplying Equation (4.2) by test functions τ and v, integrating over the domain, and integrating the gradient term by parts, one obtains the following variational formulation of the problem: find σ and u satisfying

$$\int_\Omega \kappa^{-1}\sigma \cdot \tau dx - \int_\Omega u\nabla \cdot \tau \, dx = -\int_{\partial\Omega} u\tau \cdot n ds, \quad \forall \tau \in \Sigma^0, \tag{4.4a}$$

$$\int_\Omega \nabla \cdot \sigma v dx = \int_\Omega g v dx, \quad \forall v \in U. \tag{4.4b}$$

By summing up (4.4) and applying the boundary conditions, we obtain a bilinear form acting on the trial function (σ, u) and test function (τ, v). The problem is then formulated as: find $\sigma \in \Sigma^0$ and $u \in U$ such that

$$a((\sigma, u), (\tau, v)) = L(\tau, v), \quad \forall (\tau, v) \in \Sigma^0 \times U, \tag{4.5}$$

where the bilinear form $a(\cdot, \cdot)$ is defined as

$$a((\sigma, u), (\tau, v)) := \int_\Omega \kappa^{-1} \sigma \cdot \tau \, dx - \int_\Omega u \nabla \cdot \tau \, dx + \int_\Omega \nabla \cdot \sigma v \, dx$$

and $L(\cdot)$ is defined as

$$L(v, q) := -\int_{\partial\Omega} u\tau \cdot n \, ds + \int_\Omega gv \, dx = \int_{\partial\Omega_1} \tau \cdot n \, ds - \int_{\partial\Omega_3} \tau \cdot n ds + \int_\Omega gv \, dx.$$

We also define the space $\Sigma^0 := \left\{ \tau \in H(div) \,\middle|\, \tau \cdot n|_{\partial\Omega_{2,4}} = 0 \right\}$, and $U := L^2(\Omega)$.

To discretize the above formulation in fine mesh $\mathcal{T}^h$, two discrete function spaces $\Sigma_h^0 \subset \Sigma^0$ and $U_h \subset U$ are needed to form a mixed function space $\Sigma_h^0 \times U_h$. A common choice of finite element space is Raviart–Thomas space . Specifically, we use a standard lowest order RT_0 space [57]. Then, the fine-scale problem can be written as:

$$\int_\Omega \kappa^{-1} \sigma_h \cdot \tau_h \, dx - \int_\Omega u_h \nabla \cdot \tau_h \, dx = -\int_{\partial\Omega} u_h \tau_h \cdot n \, ds, \quad \forall \tau_h \in \Sigma_h^0, \tag{4.6a}$$

$$\int_\Omega \nabla \cdot \sigma_h v_h \, dx = \int_\Omega g v_h \, dx, \quad \forall v_h \in U_h. \tag{4.6b}$$

or similar to (4.5):

$$a_h((\sigma_h, u_h), (\tau_h, v_h)) = L_h(\tau_h, v_h), \quad \forall (\tau_h, v_h) \in \Sigma_h^0 \times U_h, \tag{4.7}$$

where

$$a_h((\sigma_h, u_h), (\tau_h, v_h)) := \int_\Omega \frac{1}{\kappa}\sigma_h \cdot \tau_h \, dx - \int_\Omega u_h \nabla \cdot \tau_h \, dx + \int_\Omega \nabla \cdot \sigma_h v_h \, dx$$

and

$$L_h(v_h, v_h) := -\int_{\partial\Omega} u_h \tau_h \cdot n \, ds + \int_\Omega g v_h \, dx = \int_{\partial\Omega_1} \tau_h \cdot n \, ds - \int_{\partial\Omega_3} \tau_h \cdot n \, ds + \int_\Omega g v_h \, dx.$$

We finally obtain a discrete fine-scale system:

$$\begin{bmatrix} A_h & B_h \\ B_h^T & 0 \end{bmatrix} \begin{bmatrix} \sigma_h \\ u_h \end{bmatrix} = \begin{bmatrix} b_h \\ 0 \end{bmatrix}. \tag{4.8}$$

4.2 Coarse Problem

4.2.1 Coarse Partition of Domain

We let $\mathcal{T}^H$ be a coarse partition of the computational domain Ω with a mesh size H. The fine grid $\mathcal{T}^h$ should be a refinement of $\mathcal{T}^H$ with a mesh size h. Generally, $h \ll H$. Let $\mathcal{E}^H$ be the set of all faces in the coarse grid $\mathcal{T}^H$, and the number of coarse faces is defined to be N_e. A coarse neighborhood of $E_i \in \mathcal{E}^H$ is then defined as:

$$\omega_i := \bigcup_j \{K_j \in \mathcal{T}^H \mid E_i \in \partial K_j\}. \tag{4.9}$$

See Figure 4.1 for an illustration. With a coarse partition of the domain, we aim to construct a GMsFEM space $\Sigma_{\text{ms}} = \operatorname{span}_{i,k}\{\psi_k^{\text{ms},\omega_i}\}$ for σ corresponds to $\mathcal{T}^H$ which has a smaller dimensional size while containing the dominant information of the local region. Note that we will construct $\psi_k^{\text{ms},\omega_i}$ in a way that it is supported only in the coarse neighborhood ω_i.

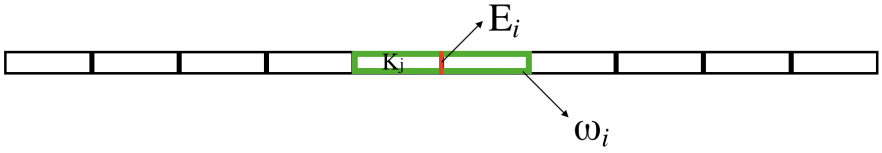


Figure 4.1: Illustration of the coarse partition $\mathcal{T}^H$ of Ω and the coarse neighborhood ω_i associated with coarse face E_i.

4.2.2 Variational Form for Coarse Problem

Similar to the fine-scale problem (4.5), the coarse problem can be formulated as: find $(\sigma_{\text{ms}}, u_{\text{ms}}) \in \Sigma_{\text{ms}} \times U_{\text{ms}}$ such that:

$$a((\sigma_{\text{ms}}, u_{\text{ms}}), (\tau, v)) = L(\tau, v), \quad \forall (\tau, v) \in \Sigma_{\text{ms}} \times U_{\text{ms}}. \tag{4.10}$$

Here, we take $U_{\text{ms}} := \left\{ u \in L^2(\Omega) \middle| \; u|_K \in \mathcal{P}^0(K), \forall K \in \mathcal{T}^H \right\}$ to be a discontinuous piecewise constant space, and Σ_{ms} will be constructed using GMsFEM in the following sections. The corresponding numerical system then writes as :

$$\begin{bmatrix} R_\sigma A_h R_\sigma^T & R_\sigma B_h R_u^T \\ R_u B_h^T R_\sigma^T & 0 \end{bmatrix} \begin{bmatrix} \sigma_{ms} \\ u_{ms} \end{bmatrix} = \begin{bmatrix} R_\sigma b_h \\ 0 \end{bmatrix},$$

where

$$R_\sigma = [\psi_1, \cdots, \psi_{N_\sigma}], \quad R_u = [r_1, \cdots, r_{N_u}].$$

Here, $\{\psi_i\}_{i=1}^{N_\sigma}$ are multiscale basis functions for flux σ and $\{r_i\}_{i=1}^{N_u}$ are restriction basis functions from U_h into U_H. N_σ and N_u are the dimensions of space Σ_{ms} and space U_{ms}, respectively.

4.2.3 Construction of Snapshot Space

The key feature of GMsFEM is that it can construct a set of basis functions that capture the micro-fine characteristics of the media. We first construct a snapshot space that can reflect the multiscale features of this particular coarse region. As discussed in (2.2.1), we can obtain

the snapshot basis functions by solving the local harmonic extension problems. For mixed formulation of the flow equation, the snapshot problem is defined locally on ω_i as follows: find $(\phi_k^{\text{snap},\omega_i}, \xi_k^{\text{snap},\omega_i}) \in \Sigma_h(\omega_i) \times U_h(\omega_i)$ such that

$$\kappa^{-1}\phi_k^{\text{snap},\omega_i} + \nabla \xi_k^{\text{snap},\omega_i} = 0, \quad \text{in } \omega_i, \tag{4.11a}$$

$$\nabla \cdot \phi_k^{\text{snap},\omega_i} = c_k^i, \quad \text{in } \omega_i, \tag{4.11b}$$

$$\phi_k^{\text{snap},\omega_i} \cdot m_i = \delta_k^i, \quad \text{on } E_i, \tag{4.11c}$$

$$\phi_k^{\text{snap},\omega_i} \cdot n_i = 0 \quad \text{on, } \partial\omega_i. \tag{4.11d}$$

Here, n_i denotes the outward unit-normal vector on $\partial\omega_i$ and m_i is a fixed unit-normal vector on E_i. Constant c_k^i is defined as

$$c_k^i := \frac{1}{|K_j^i|}\int_{E_i} \phi_k^{\text{snap},\omega_i} \cdot m_i \, ds = \frac{1}{|K_j^i|}\int_{E_i} \delta_k^i \, ds, \quad j = 1, 2.$$

In this scenario, $\omega_i = K_1^i \cup K_2^i$. Notice that the coarse edge E_i is constituted by fine edges e_k, i.e. $E_i = \cup_{k=1}^{J_i} e_k$. δ_k^i is defined as a piece-wise constant function on E_i such that

$$\delta_k^i := \begin{cases} 1, & \text{on } e_k, \\ 0, & \text{on } e_s, s \neq k. \end{cases}$$

The snapshot space on ω_i is then defined as

$$\Sigma_{\text{snap}}^{\omega_i} := \operatorname*{span}_k \{\phi_k^{\text{snap},\omega_i}\}. \tag{4.12}$$

Moreover, the global snapshot space is defined

$$\Sigma_{\text{snap}} := \oplus_i \Sigma_{\text{snap}}^{\omega_i}. \tag{4.13}$$

4.2.4 Construction of GMsFEM Basis

To further reduce the space dimension, we want to select those multiscale modes that directly reflect the major features of the media. By conducting a spectral decomposition to the snapshot space, we are able to construct the offline multiscale space Σ_{ms}. We first seek eigen-functions $\psi_k^{\text{ms},\omega_i} \in \Sigma_{\text{snap}}^{\omega_i}$ to the spectral problem

$$a_i(\psi_k^{\text{ms},\omega_i}, v) = \lambda_k^{\omega_i} s_i(\psi_k^{\text{ms},\omega_i}, v), \quad \forall v \in \Sigma_{\text{snap}}^{\omega_i}, \tag{4.14}$$

where

$$a_i(u, v) := \int_{E_i} \kappa^{-1}(u \cdot m_i)(v \cdot m_i)\, ds$$

and

$$s_i(u, v) := \int_{\omega_i} uv\, dx + \int_{\omega_i} \nabla \cdot u \, \nabla \cdot v\, dx.$$

We then sort the eigenvalues in ascending order and choose the first L_{ω_i} eigenfunctions that correspond to the smallest eigen-values. The multiscale space is then formed as

$$\Sigma_{\text{ms}} := \text{span}\{\psi_k^{\text{ms},\omega_i} \mid 1 \leq k \leq L_{\omega_i}, 1 \leq i \leq N_e\}. \tag{4.15}$$

4.3 Stability Analysis

In this subsection, we would like to present a precise analysis of the approximation error of our proposed multiscale space. The large length/width ratio of the domain is also taken into consideration. We first define the following norms and symbols:

$$||u||_{L^2(\Omega)}^2 := \int_\Omega u^2\, dx,$$

for a scalar function $u \in L^2(\Omega)$:

$$||\sigma||_{\kappa^{-1},\Omega}^2 := \int_\Omega \kappa^{-1}|\sigma|^2\, dx,$$

for vector function v;

$$||\sigma||^2_{H(\text{div},\Omega);\kappa^{-1}} := ||\sigma||^2_{\kappa^{-1},\Omega} + ||\nabla \cdot \sigma||^2_{L^2(\Omega)}\, dx,$$

for vector function σ; we further denote

$$\alpha \preceq \beta$$

if there exist a constant $C > 0$ such that $\alpha \leqslant C\beta$.

Let K be any coarse block from $\mathcal{T}^H$. Then, the projection of σ onto Σ_{snap} restricted to each K is defined to be $\hat{\sigma}$ as following:

Lemma 4.3.1. *Let* $(\sigma_h, u_h) \in \Sigma_h \times U_h$ *be fine solution of* (4.7) *and* $\hat{\sigma}$ *be the weak fine-scale solution of* (4.16)

$$\kappa^{-1}\hat{\sigma} + \nabla \hat{u} = 0, \quad \textit{in } K, \tag{4.16a}$$

$$\nabla \cdot (\hat{\sigma}) = \bar{g}, \quad \textit{in } K, \tag{4.16b}$$

$$\hat{\sigma} \cdot n = \sigma_h \cdot n, \quad \textit{on } \partial K, \tag{4.16c}$$

subject to the condition

$$\int_K \hat{u} = \int_K u_h. \tag{4.17}$$

Here, n *is the outward normal vector on* ∂K *and* $\bar{g} := \dfrac{1}{|K|}\displaystyle\int_K g$. *We then have*

$$\int_\Omega \kappa^{-1}|\sigma_h - \hat{\sigma}|^2\, dx \preceq \max_{K \in T_H} (\kappa_{min,K^{-1}}) \sum_{i=1}^{N_e} ||g - \bar{g}||^2_{L^2(K_i)}. \tag{4.18}$$

Proof. We first notice that by the construction of $\hat{\sigma}$ and that of Σ_{snap}, we have $\hat{\sigma} \in \Sigma_h \cap \Sigma_{\text{snap}}$.

Then, in each corase block K, by subtracting the variational form (4.5) from (4.7), we have

$$\int_K \kappa^{-1}(\sigma_h - \hat{\sigma}) \cdot \tau_h \, dx - \int_K \nabla \cdot (\tau_h)(u_h - \hat{u}) \, dx = 0, \quad \forall \tau_h \in \Sigma_h^0(K), \tag{4.19a}$$

$$\int_K \nabla \cdot (\sigma_h - \hat{\sigma}) v_h \, dx = \int_K (g - \bar{g}) v_h \, dx, \quad \forall v_h \in U_h(K). \tag{4.19b}$$

Taking $\tau_h = \sigma_h - \hat{\sigma}$ and $v_h = u_h - \hat{u}$ in (4.19), we can then get

$$\int_K \kappa^{-1}(\sigma_h - \hat{\sigma})^2 \, dx = \int_K (g - \bar{g})(u_h - \hat{u}) \, dx. \tag{4.20}$$

Since Raviart-Thomas space enjoys an inf-sup condition, i.e. it satisfies

$$||v_h||_{L^2(K)} \preceq \sup_{\tau_h \in \Sigma_h^0(K)} \frac{\int_K \nabla \cdot (\tau_h) v_h \, dx}{||\tau_h||_{H(\text{div},K);\kappa^{-1}}}, \qquad \forall v_h \in U_h(K). \tag{4.21}$$

Combining condition (4.21) with (4.19), we can get:

$$||u_h - \hat{u}||_{L^2(K)} \preceq \kappa_{\text{min},K}^{-\frac{1}{2}} ||\sigma_h - \hat{\sigma}||_{\kappa^{-1},K}, \tag{4.22}$$

where, $\kappa_{\text{min},K}$ is the minimum value of κ over K. Then, by (4.20)

$$||u_h - \hat{u}||_{L^2(K)} \preceq \kappa_{\text{min},K}^{-\frac{1}{2}} ||g - \bar{g}||_{\kappa^{-1},K}. \tag{4.23}$$

Now, summing up the results for all coarse blocks, we obtain the desired conclusion. □

To simplify, we only consider the type of "thin domain" $\Omega = [0,1] \times [0, Ly]$, where $\epsilon := \frac{Ly}{1} \ll 1$. We generate the coarse grid by dividing the "x-axis" of the rectangle. Since $Ly \ll 1$, $H = Ly = \epsilon$. No-flow condition is imposed merely on the upper and bottom sides of the rectangle. Thus, the multiscale basis is only taken at the vertical edges of the coarse grid.

Before we state and prove inf-sup condition for our approximation space $\Sigma_{\text{ms}} \times U_{\text{ms}}$, we consider a snapshot problem on a reference coarse neighborhood $\tilde{\omega}_i = [x_{i-1}, x_{i+1}] \times [0, Ly]$. We aim to find

$(\tilde{\phi}_k^{\text{snap},\tilde{\omega}_i}, \tilde{\xi}_k^{\text{snap},\tilde{\omega}_i}) \in H(div, \tilde{\omega}_i) \times L_2(\tilde{\omega}_i)$ that solves the following snapshot problem. For simplicity, we denoted $(\tilde{\phi}_k^{\text{snap},\tilde{\omega}_i}, \tilde{\xi}_k^{\text{snap},\tilde{\omega}_i})$ as $(\tilde{\phi}_k^i, \tilde{\xi}_k^i)$. The reference snapshot problem is defined as

$$\tilde{\kappa}^{-1}\tilde{\phi}_k^i + \nabla\tilde{\xi}_k^i = 0, \quad \text{in } \tilde{\omega}_i, \tag{4.24a}$$

$$\nabla \cdot \tilde{\phi}_k^i = \tilde{c}_k^i, \quad \text{in } \tilde{\omega}_i, \tag{4.24b}$$

$$\tilde{\phi}_k^i \cdot \tilde{m}_i = \tilde{\delta}_k^i, \quad \text{on } \tilde{E}_i, \tag{4.24c}$$

$$\tilde{\phi}_k^i \cdot \tilde{n}_i = 0, \quad \text{on } \partial\tilde{\omega}_i. \tag{4.24d}$$

where

$$\tilde{\kappa}(x, y) = \begin{pmatrix} \kappa(x, \epsilon y) & 0 \\ 0 & \epsilon^{-2}\kappa(x, \epsilon y) \end{pmatrix},$$

$\tilde{c}_k^i, \tilde{n}_i$ and $\tilde{m}_i$ are defined similar to c_k^i, n_i and m_i in the reference coarse neighborhood $\tilde{\omega}_i$. We solve this problem weakly, i.e., we seek the solutions to

$$\tilde{a}_i^{\text{snap}}((\tilde{\phi}_k^i, \tilde{\xi}_k^i), (\tilde{\tau}, \tilde{v})) = \tilde{L}_i^{\text{snap}}(\tilde{\tau}, \tilde{v}), \quad \forall (\tilde{\tau}, \tilde{v}) \in \Sigma_h(\tilde{\omega}_i) \times U_h(\tilde{\omega}_i), \tag{4.25}$$

where

$$\tilde{a}_i^{\text{snap}}((\phi, \xi), (\tau, v)) = \int_{\tilde{\omega}_i} \tilde{\kappa}^{-1}\phi \cdot \tau \, dx - \int_{\tilde{\omega}_i} \xi \cdot \nabla\tau \, dx + \int_{\tilde{\omega}_i} \nabla \cdot \phi v \, dx,$$
$$\tilde{L}_i^{\text{snap}}(\tau, v) = \int_{\tilde{\omega}_i} \tilde{c}_k^i v \, dx.$$

Let

$$\tilde{V}_{\text{snap}}^i := \operatorname*{span}_k \{\tilde{\phi}_k^i\}.$$

By proper scaling, we can get

$$\tilde{\phi}_k^i(x,y) = \begin{pmatrix} (\phi_k^{\text{snap},\omega_i})_1(x,\epsilon y) \\ \epsilon^{-1}(\phi_k^{\text{snap},\omega_i})_2(x,\epsilon y) \end{pmatrix}, \tag{4.26a}$$

$$\tilde{\xi}_k^i(x,y) = \xi_k^{\text{snap},\omega_i}(x,\epsilon y). \tag{4.26b}$$

Similarly, we have a reference spectral problem in $\tilde{\omega}_i$:

$$\tilde{a}_i(\tilde{\psi}_k^i,\tilde{\tau}) = \tilde{\lambda}_k^i \tilde{s}_i(\tilde{\psi}_k^i,\tilde{\tau}), \quad \forall \tilde{v} \in \tilde{V}_{\text{snap}}^i, \tag{4.27}$$

where $\tilde{s}_i(\sigma,\tau) := \int_{\tilde{\omega}_i} \tilde{\kappa}^{-1}\sigma\cdot\tau\, dx + \int_{\tilde{\omega}_i}(\nabla\cdot\sigma)(\nabla\cdot\tau)\, dx$, and $\tilde{a}_i(\sigma,\tau) := \int_{\tilde{E}_i} \tilde{\kappa}^{-1}(\sigma\cdot\tilde{m}_i)(\tau\cdot\tilde{m}_i)\, ds$. Moreover, by scaling, we are able to get

$$\epsilon\tilde{a}_i(\tilde{\psi}_k^i,\tilde{\psi}_k^i) = a_i(\psi_k^{\text{ms},\omega_i},\psi_k^{\text{ms},\omega_i}). \tag{4.28}$$

Theorem 4.3.2. *Assume* $\cup[x_i,x_{i+1}]$ *is an partition of* $[0,1]$. *Thus,* $\Omega = \bigcup_{1\le i\le N_e}[x_i,x_{i+1}]\times[0,Ly]$. *For any given* $u \in U_H$ *with* $\int_\Omega u = 0$, *we have*

$$||u||_{L^2(\Omega)} \le C \sup_{\sigma\in\Sigma_{ms}} \frac{\int_\Omega \nabla\cdot\sigma u\, dx}{||\sigma||_a} \tag{4.29}$$

for some constant C. *Here,* $||\cdot||_a$ *is defined as*

$$||\sigma||_a := (\sum_i a_i(\sigma,\sigma))^{\frac{1}{2}}.$$

Proof. We consider

$$\tilde{\sigma}_i = \operatorname*{argmin}_{\tilde{\psi}\in\tilde{\Sigma}_{\text{ms}}(\tilde{\omega}_i),\int_{\tilde{E}_i}\tilde{\psi}\cdot m_i=1} ||\tilde{\psi}||_{\tilde{a}}^2,$$

where $||\cdot||_{\tilde{a}}$ is defined similar to $||\cdot||_a$, i.e. $||\sigma||_{\tilde{a}} := (\sum_i \tilde{a}_i(\sigma,\sigma))^{\frac{1}{2}}$. We then let

$$\sigma_i := \begin{pmatrix} (\tilde{\sigma}_i(x,\epsilon^{-1}y))_1 \\ \epsilon(\tilde{\sigma}_i(x,\epsilon^{-1}y))_2 \end{pmatrix},$$

and

$$\sigma := \sum_i -[u]_{x_i}\sigma_i,$$

where $[u]_{x_i} = u_{i+1} - u_i$. Therefore, we have

$$\int_\Omega \nabla\cdot\sigma u\, dx = \sum_i -[u]_{x_i}\int_{\omega_i}\nabla\cdot\sigma_i\, dx = \sum_i [u]_{x_i}\int_{E_i}\sigma_i\cdot m_i\, ds = \sum \epsilon[u]^2_{x_i} = \sum_i\int_{E_i}[u]^2_{x_i}\, ds.$$

Similarly to the proof of Poincare Inequality, we can easily prove that

$$\frac{1}{H}\sum_i\int_{E_i}[u]^2_{x_i}\, ds \geq \int_\Omega u^2\, dx,$$

thus we have

$$\int_\Omega \nabla\cdot\sigma u\, dx \geq ||u||^2_{L^2(\Omega)}$$

and

$$\begin{aligned} ||\sigma||^2_a &\leq 2\sum_i [u]^2_{x_i}||\sigma_i||^2_a \\ &= 2\sum_i [u]^2_{x_i}\epsilon^{-1}||\tilde{\sigma}_i||^2_{\tilde{a}} \\ &\leq 2\sum_i\int_{E_i}[u]^2_{x_i}C^2_{\text{infsup}}\, ds, \end{aligned}$$

where C_{infsup} is defined to be $C_{\text{infsup}} = \frac{1}{H^{\frac{1}{2}}}(\sup_i \inf_{\tilde{\psi}\in\tilde{V}_{\text{ms}},\int_{\tilde{E}_i}\tilde{\psi}\cdot m_i=1}||\tilde{\psi}||^2_{\tilde{a}})^{\frac{1}{2}}$. Therefore, we finally have

$$\frac{\int_\Omega\nabla\cdot\sigma u\, dx}{||\sigma||_a} = \frac{\sum_i\int_{E_i}[u]^2_{x_i}\, ds}{||\sigma||_a} \geq \frac{(\sum\int_{E_i}[u]^2_{x_i}\, ds)^{\frac{1}{2}}}{\sqrt{2}C_{\text{infsup}}} \geq \int_\Omega u^2\, dx.$$

□

Theorem 4.3.3. *Let σ_h be the fine-grid solution obtained in* (4.6) *and σ_H be the mixed GMsFEM solution obtained in* (4.10)*. Then, the following estimate holds:*

$$\int_\Omega \kappa^{-1}|\sigma_h - \sigma_H|^2 \, dx \preceq C_{infsup}^2 \Lambda^{-1} \sum_i a_i(\hat{\sigma}, \hat{\sigma}) + \max_{K \in T_H} (\kappa_{min,K}^{-1}) \sum_i ||g - \bar{g}||_{L^2(K_i)}^2 \tag{4.30}$$

where $\Lambda = \min_i \lambda_{L^{\omega_i}+1}^{\omega_i}$ and $\hat{\sigma}$ is the projection problem solution of (4.16)*.*

Proof. By (4.6), (4.10) and the fact that $\Sigma_{\text{ms}} \subset \Sigma_h^0$ we get that

$$\begin{aligned} \int_\Omega \kappa^{-1}(\sigma_h - \sigma_H) \cdot \tau_H \, dx + \int_\Omega \nabla \cdot (\tau_H)(u_h - u_H) \, dx = 0, \qquad & \forall \tau_H \in \Sigma_{\text{ms}}^0, \\ \int_\Omega \nabla \cdot (\sigma_h - \sigma_H) v_H \, dx = 0, \quad & \forall v_H \in U_H. \end{aligned} \tag{4.31}$$

Recall (4.19a), we have

$$\int_K \nabla \cdot (\sigma_h - \hat{\sigma}) \tau_H \, dx = \int_K (g - \bar{g}) \tau_H \, dx = 0, \quad \forall v_H \in U_H.$$

Let $\hat{u}_H \in U_H$, such that

$$\int_K (\hat{u}) \, dx = \int_K (\hat{u}_H) \, dx, \quad \forall K \in \mathcal{T}_H.$$

Since v_H is a constant in each coarse block K, and so is $\nabla \cdot \tau_H$, by (4.16c) and (4.17), we have

$$\int_\Omega \nabla \cdot (\tau_H) u_h \, dx = \int_\Omega \nabla \cdot (\tau_H) \hat{u} \, dx = \int_\Omega \nabla \cdot (\tau_H) \hat{u}_H \, dx.$$

Therefore, (4.31) can be written as

$$\begin{aligned} \int_\Omega \kappa^{-1}(\sigma_h - \sigma_H) \cdot \tau_H \, dx - \int_\Omega \nabla \cdot \tau_H (\hat{u}_H - u_H) \, dx = 0, \qquad & \forall \tau_H \in \Sigma_{\text{ms}}^0, \\ \int_\Omega \nabla \cdot (\hat{\sigma} - \sigma_H) v_H \, dx = 0, \qquad & \forall v_H \in U_H. \end{aligned} \tag{4.32}$$

Since $\hat{\sigma} \in \Sigma_{\text{snap}}$, we can rewrite $\hat{\sigma}$ as

$$\hat{\sigma} = \sum_{i=1}^{N_e} \sum_{k=1}^{J_i} \hat{\sigma}_{ik} \psi_k^{\text{ms},\omega_i},$$

where $J_i = dim(\Sigma_{\text{snap}}^{\omega_i})$. We then define $\hat{\sigma}_{\text{ms}} \in \Sigma_{\text{ms}}$ as

$$\hat{\sigma}_{\text{ms}} = \sum_{i=1}^{N_e} \sum_{k=1}^{L_{\omega_i}} \hat{\sigma}_{ik} \psi_k^{\text{ms},\omega_i}. \tag{4.33}$$

Here, L_{ω_i} is the number of eigenfunctions we select for coarse neighborhood ω_i . We can further write (4.32) as

$$\begin{aligned} \int_\Omega \kappa^{-1}(\sigma_h - \sigma_H) \cdot \tau_H \, dx - \int_\Omega \nabla \cdot \tau_H (\hat{u}_H - u_H) \, dx = 0, \quad \forall \tau_H \in \Sigma_{\text{ms}}^0, \\ \int_\Omega \nabla \cdot (\hat{\sigma}_{\text{ms}} - \sigma_H) v_H \, dx = \int_\Omega \nabla \cdot (\hat{\sigma}_{\text{ms}} - \hat{\sigma}) v_H \, dx, \quad \forall v_H \in U_H. \end{aligned} \tag{4.34}$$

Let $\tau_H = \hat{\sigma}_{\text{ms}} - \sigma_H$ and $v_H = \hat{u}_H - u_H$ and plug back to (4.34). If we add up the equations, we get

$$\int_\Omega \kappa^{-1}(\sigma_h - \sigma_H) \cdot (\hat{\sigma}_{\text{ms}} - \sigma_H) \, dx = \int_\Omega \nabla \cdot (\hat{\sigma}_{\text{ms}} - \hat{\sigma})(\hat{u}_H - u_H) \, dx. \tag{4.35}$$

Since $\hat{u}_H - u_H \in U_H$ and by inf-sup condition (4.29) and (4.34), we have

$$\begin{aligned} ||\hat{u}_H - u_H||_{L^2(\Omega)} &\preceq C_{\text{infsup}} \sup_{\sigma \in \Sigma_{\text{ms}}} \frac{\int_\Omega \nabla \cdot \sigma (\hat{u}_H - u_H) \, dx}{||\sigma||_a} \\ &= C_{\text{infsup}} \sup_{\sigma \in \Sigma_{\text{ms}}} \frac{\int_\Omega \kappa^{-1}(\sigma_h - \sigma_H) \cdot \sigma \, dx}{||\sigma||_a} \\ &\preceq C_{\text{infsup}} \sup_{\sigma \in \Sigma_{\text{ms}}} \frac{||\sigma_h - \sigma_H||_{\kappa^{-1},\Omega} ||\sigma||_{\kappa^{-1},\Omega}}{||\sigma||_a}. \end{aligned}$$

Since Σ_{ms} is a finite dimensional space, all norms are equivalent. Therefore,

$$||\hat{u}_H - u_H||_{L^2(\Omega)} \preceq C_{\text{infsup}} ||\sigma_h - \sigma_H||_{\kappa^{-1},\Omega}. \tag{4.36}$$

By the definition of bilinear form $s_i(\cdot,\cdot)$, we have

$$\int_\Omega (\nabla\cdot(\hat{\sigma}_{\text{ms}}-\hat{\sigma}))^2\,dx \preceq \sum_{i=1}^{N_e}\int_{\omega_i}(\nabla\cdot(\hat{\sigma}_{\text{ms}}-\hat{\sigma}))^2\,dx \preceq \sum_{i=1}^{N_e} s_i(\hat{\sigma}_{\text{ms}}-\hat{\sigma},\hat{\sigma}_{\text{ms}}-\hat{\sigma}).$$

Then, by (4.35) and the Cauchy–Schwarz Inequality, we can obtain

$$\begin{aligned}
||\sigma_h-\sigma_H||^2_{\kappa^{-1},\Omega} &= \int_\Omega \kappa^{-1}(\sigma_h-\sigma_H)^2\,dx \\
&\leq |\int_\Omega \kappa^{-1}(\sigma_h-\sigma_H)(\sigma_h-\hat{\sigma}_{\text{ms}})| + |\int_\Omega \kappa^{-1}(\sigma_h-\sigma_H)(\hat{\sigma}_{\text{ms}}-\sigma_H)| \\
&= |\int_\Omega \kappa^{-1}(\sigma_h-\sigma_H)(\sigma_h-\hat{\sigma}_{\text{ms}})| + |\int_\Omega \nabla\cdot(\hat{\sigma}_{\text{ms}}-\hat{\sigma})(\hat{u}_H-u_H)| \\
&\preceq ||\sigma_h-\sigma_H||_{\kappa^{-1},\Omega}\cdot||\sigma_h-\hat{\sigma}_{\text{ms}}||_{\kappa^{-1},\Omega} + ||\nabla\cdot(\hat{\sigma}_{\text{ms}}-\hat{\sigma})||_{L^2(\Omega)}\cdot||\hat{u}_H-u_H||_{L^2(\Omega)} \\
&\preceq ||\sigma_h-\sigma_H||_{\kappa^{-1},\Omega}\cdot||\sigma_h-\hat{\sigma}_{\text{ms}}||_{\kappa^{-1},\Omega} + (\sum_{i=1}^{N_e} s_i(\hat{\sigma}_{\text{ms}}-\hat{\sigma},\hat{\sigma}_{\text{ms}}-\hat{\sigma}))^{\frac{1}{2}}\cdot C_{\text{infsup}}||\sigma_h-\sigma_H||_{\kappa^{-1},\Omega}.
\end{aligned}$$

By deviding $||\sigma_h-\sigma_H||_{\kappa^{-1},\Omega}$ for both sides of the inequality, we then have

$$||\sigma_h-\sigma_H||^2_{\kappa^{-1},\Omega} \preceq ||\sigma_h-\hat{\sigma}_{\text{ms}}||_{\kappa^{-1},\Omega} + (\sum_{i=1}^{N_e} s_i(\hat{\sigma}_{\text{ms}}-\hat{\sigma},\hat{\sigma}_{\text{ms}}-\hat{\sigma}))^{\frac{1}{2}}\cdot C_{\text{infsup}}. \tag{4.37}$$

For the first term on the right-hand, by triangle inequality, we have

$$||\sigma_h-\hat{\sigma}_{\text{ms}}||_{\kappa^{-1},\Omega} \leq ||\sigma_h-\hat{\sigma}||_{\kappa^{-1},\Omega} + ||\hat{\sigma}_{\text{ms}}-\hat{\sigma}||_{\kappa^{-1},\Omega}.$$

Moreover, by the definition of the spectral problem we have

$$||\hat{\sigma}_{\text{ms}}-\hat{\sigma}||_{\kappa^{-1},\Omega} \preceq \sum_{i=1}^{N_e}||\hat{\sigma}_{\text{ms}}-\hat{\sigma}||_{\kappa^{-1},\omega_i} \preceq \sum_{i=1}^{N_e} s_i(\hat{\sigma}_{\text{ms}}-\hat{\sigma},\hat{\sigma}_{\text{ms}}-\hat{\sigma}).$$

Thus, the following inequality holds:

$$||\sigma_h-\sigma_H||^2_{\kappa^{-1},\Omega} \preceq ||\sigma_h-\hat{\sigma}||_{\kappa^{-1},\Omega} + (\sum_{i=1}^{N_e} s_i(\hat{\sigma}_{\text{ms}}-\hat{\sigma},\hat{\sigma}_{\text{ms}}-\hat{\sigma}))^{\frac{1}{2}}\cdot C_{\text{infsup}}. \tag{4.38}$$

By (4.18) in Lemma 4.3.1, we can estimate the first term on the right hand side and determine the second term as all eigenfunctions are orthogonal. We then have

$$\begin{aligned} s_i(\hat{\sigma}_{\text{ms}} - \hat{\sigma}, \hat{\sigma}_{\text{ms}} - \hat{\sigma})) &= s_i(\sum_{i=1}^{N_e} \sum_{k=L_{\omega_i}+1}^{J_i} \hat{\sigma}_{ik} \psi_k^{\text{ms},\omega_i}, \sum_{i=1}^{N_e} \sum_{k=L_{\omega_i}+1}^{J_i} \hat{\sigma}_{ik} \psi_k^{\text{ms},\omega_i}) \\ &= \sum_{k=L_{\omega_i}+1}^{J_i} (\lambda_k^{\omega_i})^{-1} (\hat{\sigma}_{ik}^2) a_i(\psi_k^{\text{ms},\omega_i}, \psi_k^{\text{ms},\omega_i}) \\ &\leq (\lambda_{L_{\omega_i}+1}^{\omega_i})^{-1} a_i(\hat{\sigma}_{\text{ms}} - \hat{\sigma}, \hat{\sigma}_{\text{ms}} - \hat{\sigma}) \\ &\leq (\lambda_{L_{\omega_i}+1}^{\omega_i})^{-1} a_i(\hat{\sigma}, \hat{\sigma}). \end{aligned} \tag{4.39}$$

We thus proved the conclusion in the theorem. □

From this theorem, we actually conclude that the approximation error can be reduced if more eigenfunctions are used in each coarse neighborhood.

4.4 Numerical Results

In this section, we conduct a few numerical examples to test the performance of the proposed GMsFEM on the thin domains where the length/width ratio is taken to be $40 : 1$ in Example 1 and $80 : 1$ in Example 2. Additionally, we also test the proposed methods on a wavy thin domain to assess its performance in a more realistic case. Notice that some channelized heterogeneous permeability will be used in Example 1 and Example 2. Besides, no sink or source is placed in the computational domain for all examples.

4.4.1 Example 1

In this example we let the length/width ratio of the domain to be 40/1, the domain is partitioned into 10 coarse blocks as shown in Figure 4.1. In this example, the permeability is taken as in Figure 4.2. $\kappa(x) = 1.0$ for the background while $\kappa(x) = 0.01$ in the channelized region. Figure 4.3 and Figure 4.4 demonstrate the comparison of fine and coarse solutions for equation (4.4). Different numbers of GMsFEM basis functions are selected in each attempt.

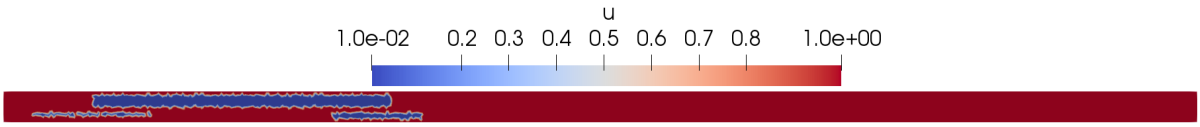


Figure 4.2: Example 1: heterogeneous permeability $\kappa(x)$.

L^2 relative error (%)	σ_H	u_H
1 basis	30.55	12.77
2 basis	5.90	12.01
4 basis	1.79	12.01
8 basis	1.01	12.01

Table 4.1: Example 1: L_2 relative errors (%) of σ_H and u_H with different numbers of GMsFEM basis functions.

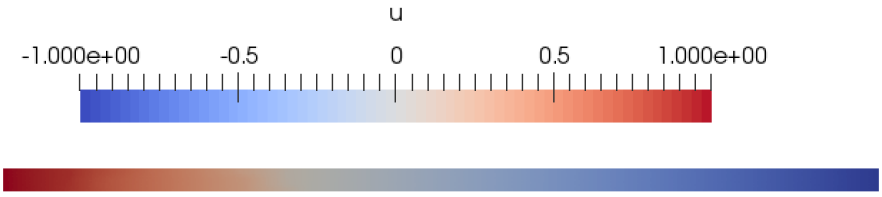
4.4.2 Example 2

In this example we let the length/width ratio of the domain to be 80/1, and the domain is again partitioned into 10 coarse blocks. In this example, the permeability is taken as in Figure 4.5.

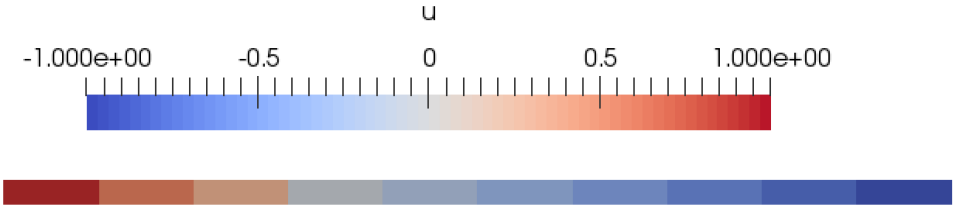
Figure 4.6 and Figure 4.7 show GMsFEM solutions for different numbers of basis functions.

L^2 relative error (%)	σ_H	u_H
1 basis	7.72	11.27
2 basis	0.93	11.22
4 basis	0.72	11.22
8 basis	0.52	11.22

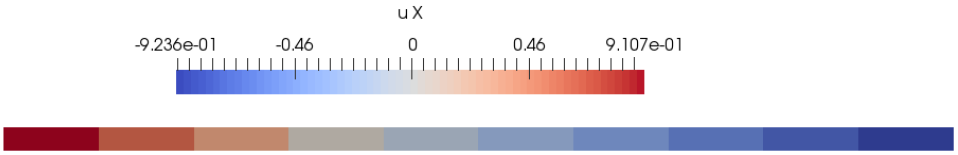
Table 4.2: Example 2: L_2 relative error(%) of σ_H and u_H with different numbers of GMsFEM basis.



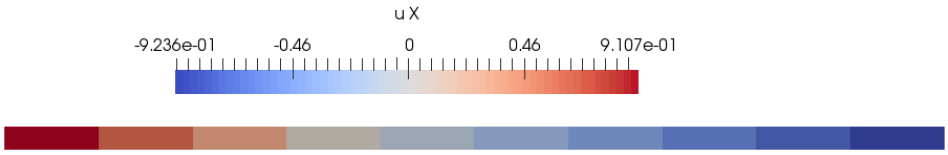
(a) Fine-scale reference pressure u_h, DOF = 11278



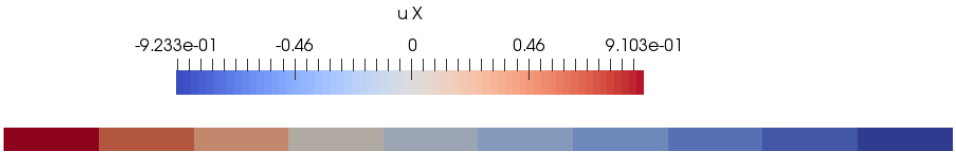
(b) GMsFEM pressure solution u_H with 1 basis in each coarse neighborhood, DOF = 10



(c) GMsFEM pressure solution u_H with 2 basis in each coarse neighborhood, DOF = 10

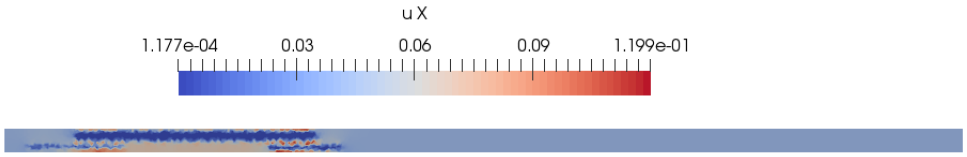


(d) GMsFEM pressure solution u_H with 4 basis in each coarse neighborhood, DOF = 10

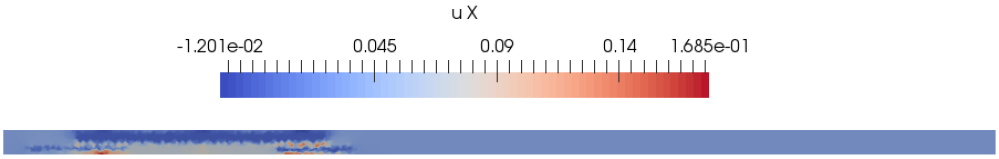


(e) GMsFEM pressure solution u_H with 4 basis in each coarse neighborhood, DOF = 10

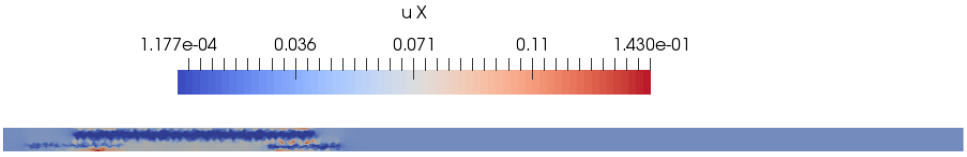
Figure 4.3: Example 1: pressure solution with different numbers of basis functions.



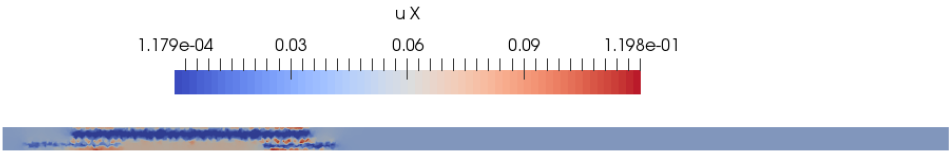
(a) Fine-scale flux solution σ_h, DOF = 5580



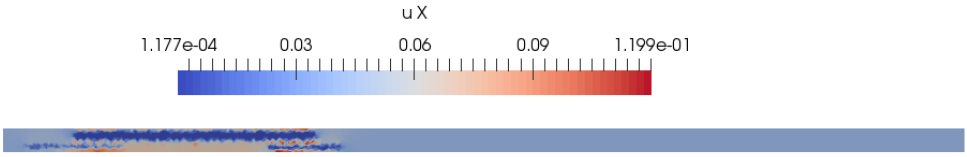
(b) GMsFEM flux solution σ_H with 1 basis in each coarse neighborhood, DOF = 9



(c) GMsFEM flux solution σ_H with 2 basis in each coarse neighborhood, DOF = 18



(d) GMsFEM flux solution σ_H with 4 basis in each coarse neighborhood, DOF = 36



(e) GMsFEM flux solution σ_H with 8 basis in each coarse neighborhood, DOF = 72

Figure 4.4: Example 1: flux solutions with different numbers of basis functions.

4.4.3 Example 3

In this example, we let the computation domain to have curvy boundaries as shown in Figure 4.8. Here, we make use of a homogeneous permeability $\kappa(x) = 1.0$ everywhere of the domain

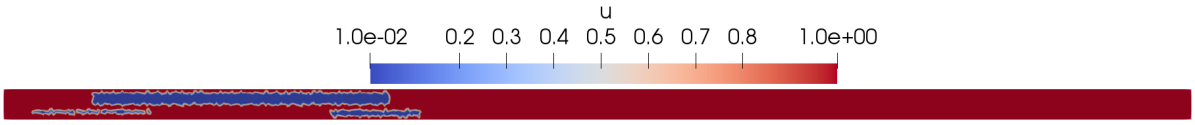


Figure 4.5: Example 2: heterogeneous permeability $\kappa(x)$.

to focus on studying the impacts of the domain geometry. The solution comparisons are then presented as in Figure 4.9, Figure 4.10 and Table 4.3.

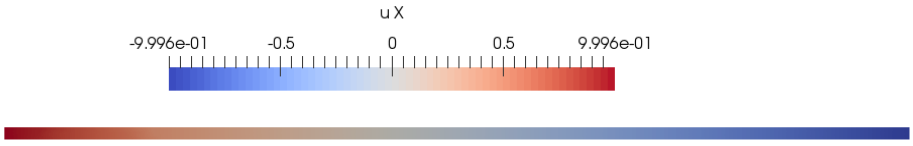
L^2 relative error (%)	σ_H	u_H
1 basis	1.38	10.00
2 basis	0.23	10.00

Table 4.3: Example 3: L_2 relative error(%) of σ_H and u_H with different numbers of GMsFEM basis in a thin domain with wavy boundaries.

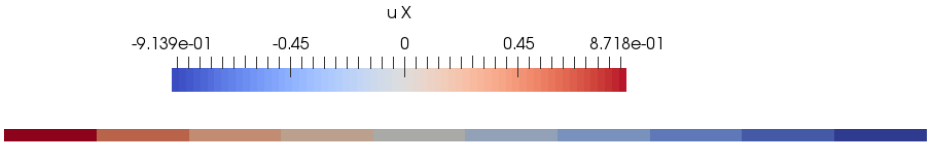
4.5 Conclusion

From both error tables and solution figures of Example 1, 2, and 3, we come to the conclusion that: 1) When we have heterogeneous coefficients, GMsFEM solutions with a single basis in each coarse neighborhood will lead to large errors. This necessitates using multiscale methods. 2) We can obtain a good approximation of fine-scale solutions with 2 basis or more. 3) GMsFEM can be easily extended to thin domains with more complicated geometry.

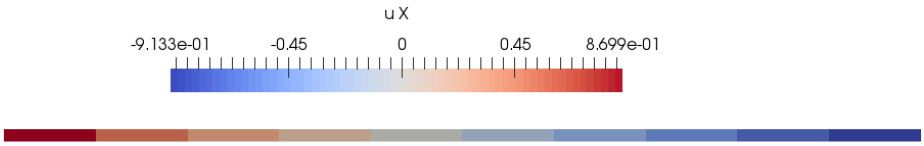
Both numerical and analytic results have shown the effectiveness of applying GMsFEM in such problems. However, challenges can be foreseen if a more complex geometry of vessel (e.g. thin domain with junctions) is considered in future works.



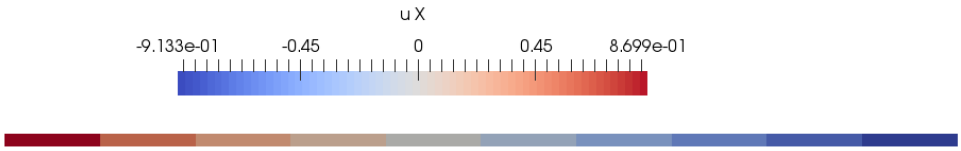
(a) Fine scale reference pressure u_h, DOF = 22524



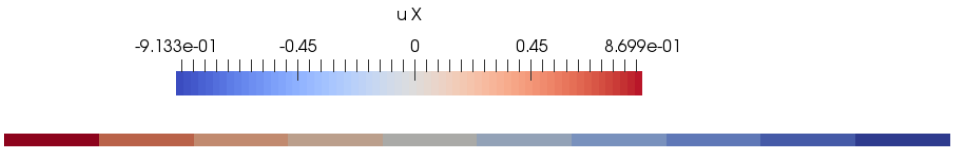
(b) GMsFEM pressure solution u_H with 1 basis in each coarse neighborhood, DOF = 10



(c) GMsFEM pressure solution u_H with 2 basis in each coarse neighborhood, DOF = 10

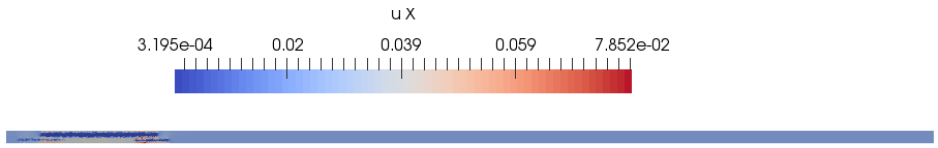


(d) GMsFEM pressure solution u_H with 4 basis in each coarse neighborhood, DOF = 10

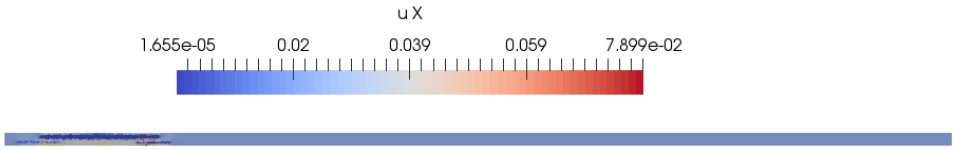


(e) GMsFEM pressure solution u_H with 8 basis in each coarse neighborhood, DOF = 10

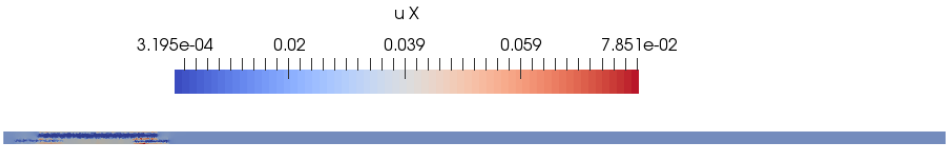
Figure 4.6: Example 2: pressure solution with different numbers of basis functions.



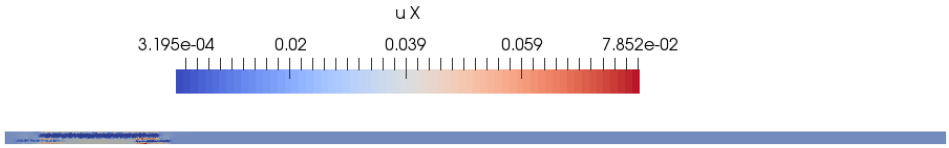
(a) Fine scale flux solution σ_h, DOF = 11263



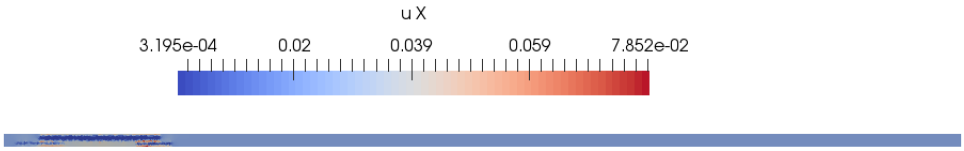
(b) GMsFEM flux solution σ_H with 1 basis in each coarse neighborhood, DOF = 9



(c) GMsFEM flux solution σ_H with 1 basis in each coarse neighborhood, DOF = 18



(d) GMsFEM flux solution σ_H with 1 basis in each coarse neighborhood, DOF = 36



(e) GMsFEM flux solution σ_H with 1 basis in each coarse neighborhood, DOF = 72

Figure 4.7: Example 2: flux solution with different numbers of GMsFEM basis.

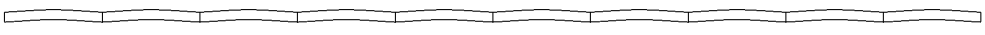
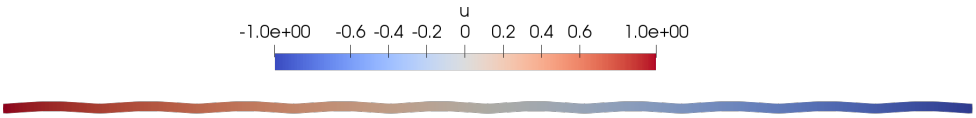
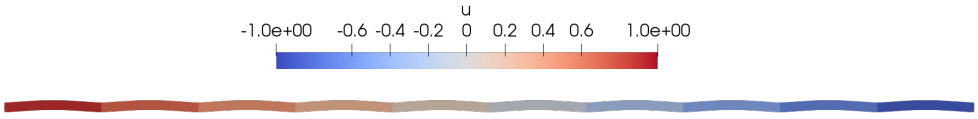


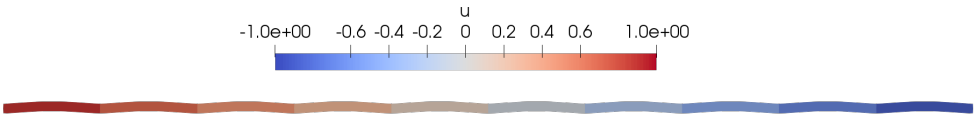
Figure 4.8: Coarse mesh of a wavy thin domain.



(a) Fine-scale reference solution u_h, DOF = 108316

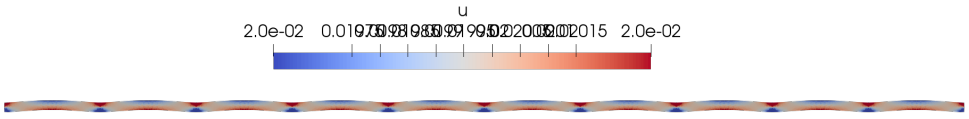


(b) GMsFEM pressure solution u_H with 1 basis in each coarse neighborhood, DOF = 10

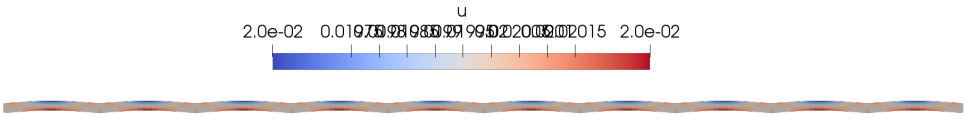


(c) GMsFEM pressure solution u_H with 2 basis in each coarse neighborhood, DOF = 10

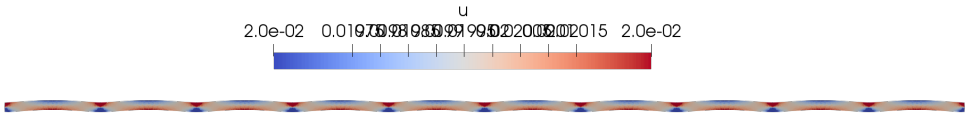
Figure 4.9: Example 3: pressure solution with different numbers of GMsFEM basis.



(a) Fine scale reference solution σ_h, DOF = 54159



(b) GMsFEM flux solution σ_H with 1 basis in each coarse neighborhood, DOF = 9



(c) GMsFEM flux solution σ_H with 2 basis in each coarse neighborhood, DOF = 18

Figure 4.10: Example 3: flux solution with different numbers of GMsFEM basis.

5. DEEP MULTISCALE MODEL LEARNING

In this chapter, we aim to construct a robust deep learning architecture for reduced-order model of fluid dynamics. More precisely, the flow dynamics can be thought of as multi-layer networks, where each layer, in general, is a nonlinear forward map and the number of layers relates to the internal time steps. The network structure is designed in a sparsely-connected manner. We utilize a reduced order model to determine the connectivity between the input layer and the first hidden layer while each neuron in the input layer stands for a degree of freedom in the coarse model. Specifically, we utilize the non-local multi-continuum approach [37] as our upscaled model.

Due to the lack of available observation data, the training data is supplemented with computational data as needed. Therefore, the trained network provides a model that inherits the merits from both the computational model and the true model underlying the physical observations.

In this chapter, we will present the main ingredients of our approach as well as the numerical results. In Section 5.1, we present the general multiscale concepts. Section 5.2 is dedicated to neural network construction while in Section 5.3, we present numerical results.

5.1 Preliminaries

In general, we study

$$\frac{\partial u}{\partial t} = F(x, t, u, \nabla u, I), \tag{5.1}$$

where I denotes the input, which can include the media properties, such as the permeability field, source terms (well rates), or initial conditions. $F(\cdot)$ can have a multiscale dependence with respect to space and time. The coarse-grid equation for (5.1) can have a complicated form for many problems (cf. [37]). This involves multiple coarse-grid variables in each computational coarse grid, non-local connectivities between the coarse-grid variables, and complex nonlinear local problems with constraints. In a formal way, the coarse-grid equations in the time interval $[t_n, t_{n+1}]$ can be written as $u_i^{j,n}$, where i is the coarse-grid block, j is a continuum representing the coarse-grid variables, and n is the time step. More precisely, for each coarse-grid block i, one may need several

coarse-grid variables, which will be denoted by j. The equation for $u_i^{j,n}$, in general, has a form

$$\overline{u}_i^{j,n+1} - \overline{u}_i^{j,n} = \sum_{i,j} \overline{F}_{i,j}(x, t, \triangle t, \overline{u}_i^{j,n+1}, \nabla \overline{u}_i^{j,n+1}, I), \tag{5.2}$$

where $\overline{u}_i^{j,n}$ is the average solution at time t^n, for the j_{th} continuum within i_{th} coarse block, and the sum is taken over some neighborhood cells and corresponding connectivity continuum. The computation of $\overline{F}$ can be expensive and involves local nonlinear problems with constraints. In many cases, researchers use general concepts from upscaling, for example, the number of continua, the dependence of $\overline{F}$, and the non-locality, to construct multiscale models. We propose to use the overall concept of the complex upscaled models in conjunction with deep learning strategies to design novel data-aware coarse-grid models.

In this chapter, we consider a special case of (5.1), the diffusion equation in a fractured media

$$\frac{\partial u}{\partial t} - \mathrm{div}(\kappa(x)\lambda(t, x)\nabla u) = g(t), \quad \text{in} \quad \Omega, \tag{5.3}$$

subject to some boundary conditions. Our numerical examples consider the zero Neumann boundary condition $\nabla u \cdot n = 0$. Here, Ω is the computational domain, u is the pressure of flow, $g(t)$ is a time dependent source term, and $\kappa(x)$ is a fixed heterogeneous fractured permeability field. The $\lambda(t, x)$ is some given mobility which is time-dependent and represents the nonlinearities in the two-phase flow. Our approach can be applied to nonlinear equations. Regarding the input parameter I, we will consider source terms $g(t, I)$, which correspond to well rates. In general, we can also consider permeability and initial conditions as the input parameters. In this chapter, we will only solve the PDE for different source term configurations.

5.1.1 Multiscale Model: Non-local Multi-continuum Approach

The fine-scale solution of (5.3) on the fine mesh $\mathcal{T}^h$ can be obtained using the standard finite element scheme, with the backward Euler method for time discretization:

$$\left(\frac{u_f^{n+1}-u_f^n}{\Delta t}, v\right) + (\kappa\lambda^{n+1}\nabla u_f^{n+1}, \nabla v) = (g^{n+1}, v). \tag{5.4}$$

Here, $(\cdot,\cdot)$ denotes the L^2 inner product. In the matrix form, we have

$$M_f u_f^{n+1} + \Delta t A_f u_f^{n+1} = \Delta t b_f + M_f u_f^n, \tag{5.5}$$

where M_f and A_f are fine-scale mass matrix and stiffness matrix respectively, and b_f is the right hand side vector.

With a proper construction of NLMC basis functions $\{\psi_m^{(i)}\}$ as discussed in Section 2.2.2, we then define the transmissibility matrix T by

$$T_{mn}^{(i,j)} = a(\psi_m^{(i)}, \psi_n^{(j)}). \tag{5.6}$$

We note that m and n denote different continua, and i, j are the indices for coarse blocks. Since the multiscale basis are constructed in oversampled regions, the support of multiscale basis for different coarse degrees of freedom will overlap, and this results in non-local transfers. The mass transfer between continua m in coarse block i and continua n in coarse block j is $T_{mn}^{(i,j)}([u_T]_n^{(j)} - [u_T]_m^{(i)})$, where $[u_T]$ is the coarse scale solution.

With a simple index, we can write T (transmissibilities) in the following form

$$\begin{bmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ t_{21} & t_{22} & \dots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \dots & t_{nn} \end{bmatrix}, \tag{5.7}$$

where $n = \sum_{i=1}^{N}(1 + L_i)$ is the total number of coarse degrees of freedom. L_i is the number of discrete fracture segments in the coarse block K_i, 1 stands for the degree of freedom in the matrix and N is the number of coarse blocks.

The upscaled model for the diffusion problem (5.3) will be formed as follows

$$M_T u^{n+1} + \Delta t A_T u^{n+1} = \Delta t b_T + M_T u^n, \tag{5.8}$$

where A_T is the NLMC coarse scale transmissibility matrix, i.e.

$$\begin{pmatrix} -\sum_j t_{1j} & t_{12} & \dots & t_{1n} \\ t_{21} & -\sum_j t_{2j} & \dots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \dots & -\sum_j t_{nj} \end{pmatrix},$$

and M_T is an approximation of the coarse scale mass matrix [37]. We note that both A_T and M_T are non-local and defined for each continua.

To this point, we obtain an upscaled model from the NLMC method. We remark that the results in [37] indicate that the upscaled equation in our modified method can use small local regions.

5.2 Deep Multiscale Model Learning (DMML)

We will utilize a rigorous NLMC model as stated in the previous section and Section 2.2.2 to solve the coarse scale problems and use the resulting solutions in deep learning framework to approximate the operator $F(\cdot)$ in (5.1). The advantages of NLMC approach are that one can get not only very accurate approximations compared to the reference fine grid solutions, but also coarse grid solutions that have important physical meanings. That is, the coarse grid parameters are the average pressure in the corresponding matrix or fracture in a coarse block. Usually, $\overline{F}$ is difficult to compute and it is conditioned to data. The idea of this work is to use the coarse grid information and available real data in combination with deep learning techniques to overcome the computation difficulty of $\overline{F}$.

It is clear that the solution at the time instant $n+1$ depends on the solution at the time instant n as well as input parameters, such as permeability/geometry of the fractured media and source terms. Here, we would like to learn the relationship of the solutions between two consecutive time instants using a multi-layer network. If we simply take only computational data in the training process, the neural network will provide a forward map to approximate our reduced-order models.

To be specific, let m be the number of samples in the training set. Suppose for a given set of various input parameters, we use NLMC method to solve the problem and obtain the coarse grid solutions

$$\begin{aligned}\{&u_1^1, \cdots, u_1^{n+1},\\ &u_2^1, \cdots, u_2^{n+1},\\ &\cdots, \cdots,\\ &u_m^1, \cdots, u_m^{n+1}\}\end{aligned}$$

at all time steps for these m samples. Our goal is to use deep learning to learn fluid dynamics from the coarse grid solutions and find a network $\mathcal{N}(\cdot)$ to describe the push-forward map between u^n and u^{n+1} for any training sample such that:

$$u^{n+1} \sim \mathcal{N}(u^n, I^n), \tag{5.9}$$

where I^n is some input parameter which can also change with respect to time, and $\mathcal{N}(\cdot)$ is a multi-layer network to be trained.

Remark: The proposed framework also considers nonlinear elliptic PDEs, where the map $\mathcal{N}(\cdot)$ corresponds to the linearized discrete system.

In deep network, we call u^n and I^n the inputs, and u^{n+1} the output. One can take the coarse solutions at time step 1 (initial time instant) to time step n as the inputs, and solutions at time 2 to $n+1$ (final time instant) as corresponding outputs in the training process. In this case, a universal neural net $\mathcal{N}(\cdot)$ can be obtained. The solution at time 1 can be forwarded all the way to time $n+1$

by repeatedly applying the universal network n times, that is,

$$u^{n+1} \sim \mathcal{N}(\mathcal{N} \cdots \mathcal{N}(u^1, I^1) \cdots, I^{n-1}), I^n). \tag{5.10}$$

Then, in the future testing/predicting procedure, given a new coarse scale solution at initial time u^1_{new}, we can also easily obtain the solution at final time step by the deep neural network

$$u^{n+1}_{\text{new}} \sim \mathcal{N}(\mathcal{N} \cdots \mathcal{N}(u^1_{\text{new}}, I^1) \cdots, I^{n-1}), I^n). \tag{5.11}$$

One can also train independent forward maps for any two consecutive time instants as needed. That is, we will have $u^{j+1} \sim \mathcal{N}_j(u^j, I^j)$, for $j = 1, \cdots, n$. In this case, to predict the final time solution u^{n+1}_{new} given the initial time solution u^1_{new}, we use n different networks $\mathcal{N}_1(\cdot), \cdots, \mathcal{N}_n(\cdot)$

$$u^{n+1}_{\text{new}} \sim \mathcal{N}_n(\mathcal{N}_{n-1} \cdots \mathcal{N}_1(u^1_{\text{new}}, I^1) \cdots, I^{n-1}), I^n).$$

Besides the previous time step solutions, the other input parameters I^n such as permeability or source terms can be different when entering the network at different time steps.

As mentioned previously, we can also take the input in the sense of "region of influence". It is important to use reduced-order model, since it will identify the regions of influence and an appropriate number of variables. In NLMC approach, we construct a non-local multi-continuum transmissibility matrix, which provides us information about the connections between coarse parameters. It has been shown in [37] that, due to the exponential decay of the global basis function away from the target coarse region, one can use the constructed local basis functions to solve the problem. These basis functions are only supported in a small oversampling local region and can still provide us with a good accuracy in solutions compared with those using global basis functions. Moreover, the transmissibility matrix formed by these local basis functions also indicates local connections (in a slightly larger oversampling region) between a target coarse degree of freedom (dofs) with others. Taking advantage of the underlying NLMC model, we can simplify the problem

when designing neural networks. Typically, for specific coarse degrees of freedom (corresponding to a coarse block matrix or a fracture in the coarse block) of the solution at time instant $n+1$, we can only activate the connections between this coarse degree of freedom and the coarse scale degrees of freedom in some oversampling neighborhood at time instant n. The advantage of defining regions of the influence is to reduce the complexity of the deep network. An illustration of the comparison between deep neural nets with full connections and a network with local connections indicated by region of influence is shown in Figure 5.1. In Figure 5.1 (a), we illustrate the deep network using full connections, where the input layer and the first hidden layer of the network are fully connected. This means, for example, the matrix continua parameter in any coarse block K_i is always connected with all the dofs in the matrix continua (blue shaded coarse blocks/neurons) and all the dofs on the fracture continua (purple shaded coarse blocks/neurons). On the other hand, in Figure 5.1 (b), the network applies some local connections in the first layer. For the coarse block K_i, it only connects with a few matrix continua dofs (pink shaded areas/neurons) and a limited number of fracture continua dofs (yellow shaded areas/neurons).

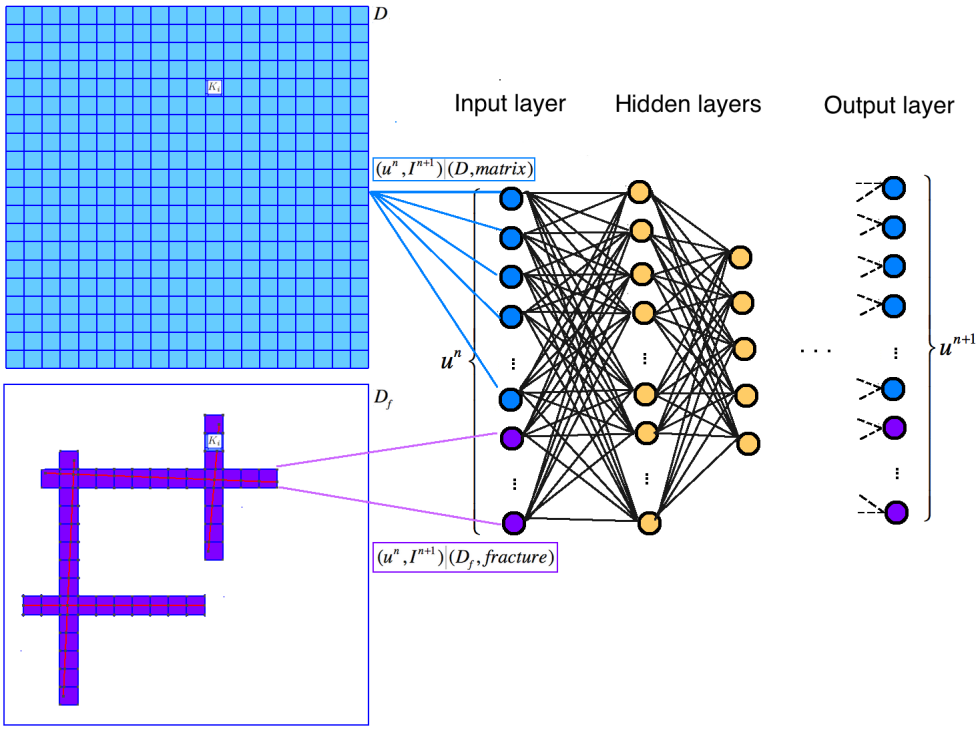
Besides all the ideas stated above, in this work, we also aim to incorporate available observed data in the neural net, which will modify the reduced order model and improve the performance of the model such that the new model will take into account real data effects. First, we introduce some notations. We denote the simulation data by

$$\{u_s^1, \cdots, u_s^{n+1}\}$$

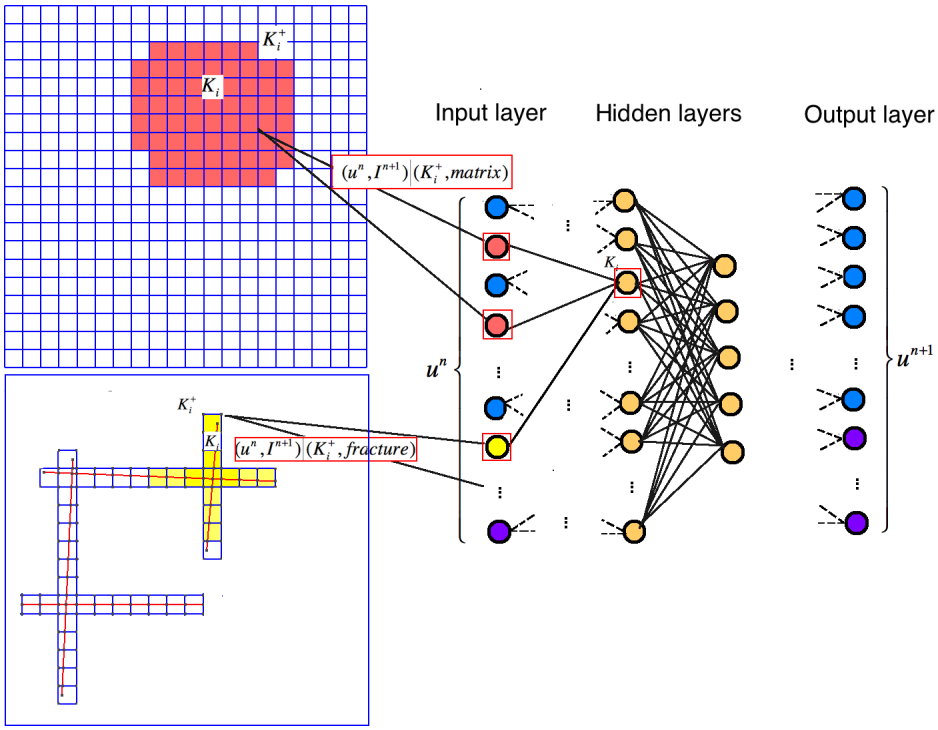
and denote the "observation" data by

$$\{u_o^1, \cdots, u_o^{n+1}\}$$

at all time steps for these m samples. To get the observed data, we can (1) perturb the simulation data, (2) perturb the permeability or geometry of the fractured media, run a new simulation and use the results as observed data, (3) use available experimental data. We want to investigate the effects



(a) Deep network using full connections



(b) Deep network using local connections indicated by the region of influence.

Figure 5.1: Comparison of deep nets with full connections and neural net with local connections indicated by "region of influence".

of including observation data in the output of the deep neural nets.

As a comparison, we will consider three networks:

- Network A: $\mathcal{N}_o(\cdot)$ uses all observation data as outputs,

$$u_o^{n+1} \sim \mathcal{N}_o(u_o^n, I^n). \tag{5.12}$$

- Network B: $\mathcal{N}_m(\cdot)$ uses a mixture of observation data and simulation data as output,

$$u_{\text{mixed}}^{n+1} \sim \mathcal{N}_m(u_{\text{mixed}}^n, I^n). \tag{5.13}$$

- Network C: $\mathcal{N}_s(\cdot)$ uses all simulation data (no observation data) as output,

$$u_s^{n+1} \sim \mathcal{N}_s(u_s^n, I^n). \tag{5.14}$$

where u_{mixed} is a mixture of simulation data and observed data.

In Network A, we assume the observation data is sufficient, and consider the observation data at time $n+1$ as a function of the observation data at time n. In this case, the map fits the data well but will ignore the simulation model if the data are obtained without using underlying simulation model in any sense. This is usually not the case in reality, since the observation data are expensive to get and deep learning requires a large amount of data to make the training effective. In Network C, we simply take all data from simulation in the training process. One will get a network describing the simulation model (in our example, the NLMC model) as best as it can but ignoring the observational data effects. This network can serve as an emulator (simplified forward map, which avoids deriving/solving coarse-grid models) to do a fast simulation. We will utilize Network A and C results as references and investigate more about Network B. Network B is the one where we take a combination of computational data and observational data to train. It will not only take into account the underlying physics but also use the real data to modify the model, thus resulting in a

data-driven approach.

We expect that the proposed algorithm to provide a new upscaled model that can honor the data while it follows our general multiscale concepts.

5.2.1 Network Structures

In our example, without loss of generality, we suppose that there are uncertainties in the injection rates g, i.e., the value or the position of the sources can vary among samples. Suppose we have a set of different realizations of the source $\{g_1, g_2, \cdots, g_m\}$, where m is a sufficiently large number, we need to run simulations based on NLMC model and take the solutions as data for deep learning. To obtain the observation data, we will solve the problem with the same set of source realization using the fine scale model.

As discussed in the previous section, we consider three different networks, namely $\mathcal{N}_o(\cdot)$, $\mathcal{N}_m(\cdot)$ and $\mathcal{N}_s(\cdot)$. For each of these networks, we take the vector $x = (u_\alpha^n, g^n)$ ($\alpha = o, m, s$) constituted by the coarse scale solution vector and the source term vector at a particular time step as the input. As discussed before, we can take the input coarse scale parameters in the whole domain Ω or in the region of influence K^+. Based on the availability of the observational data in the example pairs, we will define an appropriate network among (5.12), (5.13) and (5.14) accordingly. The output $y = u_\alpha^{n+1}$ is taken to be a coarse scale solution in the next time step, where $\alpha = o, m, s$ corresponds to three networks. Assume for a large set of samples of the source terms, there exist both corresponding computational data u_s and observation data u_o. We will use these data to train deep neural networks $\mathcal{N}(\cdot)$, such that they can approximate the function $F(\cdot)$ in (5.1) well, with respect to the loss functions. Then for some new source term g_{m+1}, given the coarse scale solution at time instant n, we expect the networks output $\mathcal{N}(u_\alpha^n, g_{m+1}^n; \theta^*)$ is close to the output data u_α^{n+1}.

Here, we briefly summarize the architecture of the network $\mathcal{N}_\alpha$, where $\alpha = o, m, s$ for three networks we defined in (5.12), (5.13) and (5.14), respectively.

- Input: $x = (u_s^n, g^n)$ is the vector containing the coarse scale solution vector and the source term at a particular time step t_n.

- Output: $y = u_\alpha^{n+1}$ is the coarse scale solution at the next time step.

- Sample pairs: $N = mn$ example pairs of (x_j, y_j) are collected, where m is the number of samples of flow dynamics and n is the number of time steps.

- Loss function: The typical cost function for regression problems is the mean squared error function: $\frac{1}{N}\sum_{j=1}^{N} \|y_j - \mathcal{N}_\alpha(x_j;\theta)\|_2^2$. In our numerical examples, our output is the coarse grid solution vector, so we would like to consider the relative L^2 error as the loss function, i.e. $\frac{1}{N}\sum_{j=1}^{N} \frac{||y_j - \mathcal{N}_\alpha(x_j;\theta)||_{L^2}}{||y_j||_{L^2}}$.

- Weighted loss function: In building a network in $\mathcal{N}_m(\cdot)$ by using a mixture of N_1 pairs of observation data $\{(x_j, y_j)\}_{j=1}^{N_1}$ and N_2 pairs of computational data $\{(x_j, y_j)\}_{j=N_1+1}^{N}$, where $N_1+N_2 = N$, we may consider using weighted loss function, i.e., $w_1 \sum_{j=1}^{N_1} \frac{||y_j - \mathcal{N}_\alpha(x_j;\theta)||_{L^2}}{||y_j||_{L^2}} + w_2 \sum_{j=N_1+1}^{N} \frac{||y_j - \mathcal{N}_\alpha(x_j;\theta)||_{L^2}}{||y_j||_{L^2}}$, where $w_1 > w_2$ are user-defined weights.

- Activation function: The popular ReLU function (the rectified linear unit activation function) is a common choice for activation function in training deep neural network architectures [58]. In our numerical examples, we use ReLU activation in the hidden layers and use a linear activation in the output layer.

- DNN structure: The number of layers and the number of neurons in each layer are specified in the following numerical examples. For the connection between the input neurons and first layer neurons, we will use dense connections or our self-designed local connections, and compare their performance in the numerical examples.

- Training Optimizer: We use AdaMax [59], a stochastic gradient descent (SGD) type algorithm well-suited for high-dimensional parameter space, in minimizing the loss function.

5.3 Numerical Results

In this section, we present some representative numerical results. We consider the fractured media as shown in the Figure 5.2, where the red lines denotes the fractures. The permeability of the

matrix is $\kappa_m = 1$, and the permeability of the fractures are $\kappa_f = 10^3$. We assume that the observed data samples are obtained from the fine scale model solutions for different source terms. As for the corresponding computational data, we compute the NLMC solutions on a 20 by 20 coarse grid as shown in Figure 5.2, using the same set of source terms and permeability coefficients. As discussed in [37], the local basis are constructed in the oversampled regions of each coarse block. In our numerical examples, we choose two layers of oversampling. With this small oversampling size, the NLMC simulation is fast, but as a trade-off, it sacrifices a little accuracy. Generally, in practice, the relative upscaling errors for the NLMC solution and averaged fine scale solution are $5\% - 15\%$. This guarantees the NLMC solutions acceptable errors and room for improvement. We will train the networks $N_s(\cdot)$, $N_o(\cdot)$ and $N_m(\cdot)$ using the computational samples (from NLMC model), observation samples (from fine scale model), and the mixture of them, respectively.

All the network training are performed using the Python deep learning API Keras [60] with TensorFlow framework.

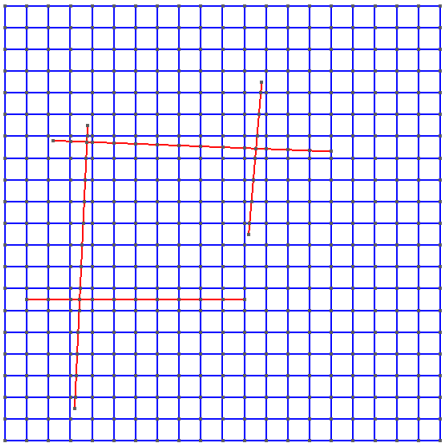


Figure 5.2: Geometry (permeability) for obtaining both simulation and observation data.

5.3.1 Example 1

In our first example, we use a time independent, constant mobility coefficient $\lambda = 1$. For the source term, we use a piece-wise constant function. Namely, in one of the coarse blocks, the value

of g is a positive number c. In another coarse block, the value of g is $-c$, and $g = 0$ elsewhere. This represents a two-well source; one of the blocks is the injection well, the other is the production well. Randomly selecting the location of the two wells, with the constraint that the two wells are well separated in the domain, we get source samples $g_1, \cdots, g_{1000}$. In this example, we set the values of the source to be time independent. The equation (5.3) is solved using both the fine scale and NLMC model, where we set $T = 0.1$, and divide T into 10 time steps.

For the 1000 source terms, we use the solutions correspond to the last 100 for validation, i.e, they will not be seen in the training process. We will use 300, 500 and 900 out of the first 900 solutions to train the network separately.

As discussed before in (5.9), we would like to find a universal deep network to describe the map between two consecutive time steps. We use the solution at time step n as input, and solution at time step $n + 1$ as corresponding output, where $n = 1, 2, 3, 4.....9$. That is, each sample pair is in the form of $(u^n, u^{n+1}), n = 1, 2, ...9$. For example, the solutions corresponding to 300 different training source terms result in $300 * 9 = 2700$ samples, and the solutions corresponding to 100 testing source terms result in $100 * 9 = 900$ testing samples, where the multiplier 9 stands for 9 time steps (time steps 1 to 9, or time 2 to 10).

We will test the performance of the three networks (5.12), (5.13), and (5.14). For $N_s(\cdot)$, we take all training samples to be the NLMC solutions. In this case, there is no observation data in training, and the network will only approximate the NLMC model. For $N_o(\cdot)$, we use all the fine scale solutions as training samples, thus the network aims to approximate the fine grid model and will be used as reference. As for $N_m(\cdot)$, we take half training samples from u_s and the other half from u_o. Specifically, we assume the observation data are given for some well configurations while for other configurations observation data can only be replaced by simulation data due to their rarity. This is the case when the network is trained partially with true data. We expect the trained network $N_m(\cdot)$ to produce an improved model compared with $\dot{N_s()}$. In the training process, we also consider both the full connections and local connections indicated by the region of influence input (see Figure 5.1), where we use multiscale concepts to reduce the region of influence (connections)

between the nodes.

5.3.1.1 Full connection

The three networks are firstly constructed adopting the structure of DNN with densely connected layers. The dimension of the input is 445, which is the degree of freedom in the NLMC model associated to the 20 by 20 coarse grid and the fracture configuration as shown in Figure 5.2. For this example, since the mobility coefficient is a constant which makes the map linear, we only take 1 layer with 445 neurons. The activation function is chosen to be linear at the output layer. The training was performed over 50 epochs, and the batch size is chosen to be 100. We use the Adam algorithm as the optimizer, and the learning rate is 0.002. The number of trainable parameters in this network is $198,470$. For the loss function, we use the relative L^2 error between the true data (samples computed from fine grid model) and predicted data (obtained from the output of the neural networks) , i.e.

$$\frac{||u_o^{n+1} - N_\alpha(u_s^n, I^{n+1})||_{L^2}}{||u_o^{n+1}||_{L^2}}, \quad \alpha = o, s, m.$$

The training and validation losses for $N_o(\cdot)$ are plotted in the left of Figure 5.3, and they have similar behavior for the other two networks. We remark that, in order to compare the performance for different sizes of the training data set (namely $2700, 4500, 8100$), in the figure, the training losses (vs epochs) are depicted at the end of learning all samples in the whole data set instead of after every batch. We observe that, for each data set, the validation loss is very close to the training loss, which indicates that the network performs and generalizes well.

Next, in the testing procedure, we input an identical input data set to three networks defined in (5.12), (5.13), (5.14), and compare the predicted outputs from the three networks with the corresponding observation data. The errors are computed in relative L^2 norm. The results are shown in Table 5.1. We can see that, by mixing the computational and observation data (the second column in the table), we can get a better model, since the mean error of $\|\mathcal{N}_m(u_s^n, I^n) - u_o^{n+1}\|$ among testing samples is closer to that of $\|\mathcal{N}_o(u_s^n, I^n) - u_o^{n+1}\|$.

	Relative Errors (%)		
Number of Training Samples	$\left\|u^j_{pred,o} - u^j_o\right\|$	$\left\|u^j_{pred,m} - u^j_o\right\|$	$\left\|u^j_{pred,s} - u^j_o\right\|$
2700	6.5	6.7	7.1
4500	4.8	5.1	5.7
8100	3.7	4.0	4.7

Table 5.1: Example 1, fully connected network. Number of network parameters: $198,470$. Mean error between prediction and true solutions for two consecutive time steps, 900 samples are tested for three different networks.

5.3.1.2 Sparse connection: region of influence

Though we have obtained promising results using fully connected networks, the number of trainable parameters is quite large. In this section, we would like to design a locally-connected layer in the neural network by taking advantage of the region of influence in the underlying model. This can help to reduce the trainable parameters in the network.

The idea is to reduce the full connections among the neurons between two layers. Thanks to the NLMC model, we derive the transmissibility matrix T that describes the nonlocal connections of coarse degrees of freedom, not only spatially but also across continua. This provides us a way to define the connections between the input neurons to the first layer. That is, we design a layer with the number of trainable weight parameters equal to the nonzero entries in the matrix T. This can reduce the number of parameters due to the sparsity of T. The defined sparse weight will only activate the connections between the input and nodes in the next layer as indicated in the transmissibility matrix. The following hidden layers will still be fully connected if they exist.

Remark: We remark that a direct application of Convolutional Neural Network (CNN) is not trivial for our problem. Since the samples we used in the network training are the coarse scale solutions for the multi-continuum model, which contains the degrees of freedom from both matrix and fracture continuum. In our example, the degrees of freedom for the matrix continuum lies in the 20 by 20 coarse grid (which is 400), but the additional fracture degrees of freedom only lies in the coarse blocks which contain the fractures (which is 45 in the geometry as shown in Figure 5.2). The multi-continuum solutions thus can not be directly represented by a square image which is needed

for CNN. Furthermore, according to the NLMC model, the transmissibility connections exist not only among coarse blocks but also among different continua, and the effects of the fracture continua should not be ignored. We then try to extend the fracture continua solution to the 20 by 20 grid using zero padding, and input the matrix and fracture continua solutions as two-channel images to train CNN network. But the zero padding procedure increases the number of training parameters (instead of decreasing as expected when using CNN), since we enlarge the input dimension from 445 to 800. And the training accuracy is also affected in a bad sense because the network has the additional burden to learn the zeros in the second channel. Thus, a self defined locally connected layer is needed.

We would like to compare the predicted results from the neural networks trained with different types of coarse parameters. Specifically, two types of networks are trained with the coarse parameters in the whole domain and parameters only in the region of influence, respectively. Comparing Table 5.1 and Table 5.2, we can see that, using the region of influence idea can result in similar results for all three networks $\mathcal{N}_o(\cdot)$, $\mathcal{N}_m(\cdot)$ and $\mathcal{N}_s(\cdot)$ when we use similar network parameters such as the number of layers, number of neurons (445) in each layer, training epochs (50), learning rate (0.002), loss functions (relative l^2 error) and activation functions (linear). The training/validation losses are plotted in the right of Figure 5.3 for Locally Connected Networks (LCN). We observe that the losses of LCN decay faster compared with those in DNN. As for the number of trainable parameters, it is $198,470$ for the fully connected network, but is only $28,107$ for the sparsely connected network. This suggests that, the data in the region of influence of the underlying model is of dominant importance in deciding the outputs and thus enables reduction in training effort.

5.3.2 Example 2

In our second example, we use heterogeneous time-dependent mobility and source term. Here, we fix the location of the source term and vary the value of the source. The distributions of the mobility in some time steps are shown in Figure 5.4, which is from two-phase flow mobility. The source term in the right hand side is defined as follows. At $0 \leq x \leq 0.1, 0 \leq y \leq 0.1$, we have $g = 10[(\sin(\alpha x))^2 + (\sin(\beta y))^2]$ denotes an injection well, and at $0.9 \leq x \leq 1.0, 0.9 \leq y \leq 1.0$

	Relative Errors (%)		
Number of Training Samples	$\left\|u^j_{pred,o} - u^j_o\right\|$	$\left\|u^j_{pred,m} - u^j_o\right\|$	$\left\|u^j_{pred,s} - u^j_o\right\|$
2700	6.2	6.4	6.7
4500	5.1	5.3	5.8
8100	4.3	4.6	5.1

Table 5.2: Example 1, locally connected network. Number of network parameters: $28,107$. Mean errors between prediction and true solutions for two consecutive time steps, 900 samples are tested for three different networks.

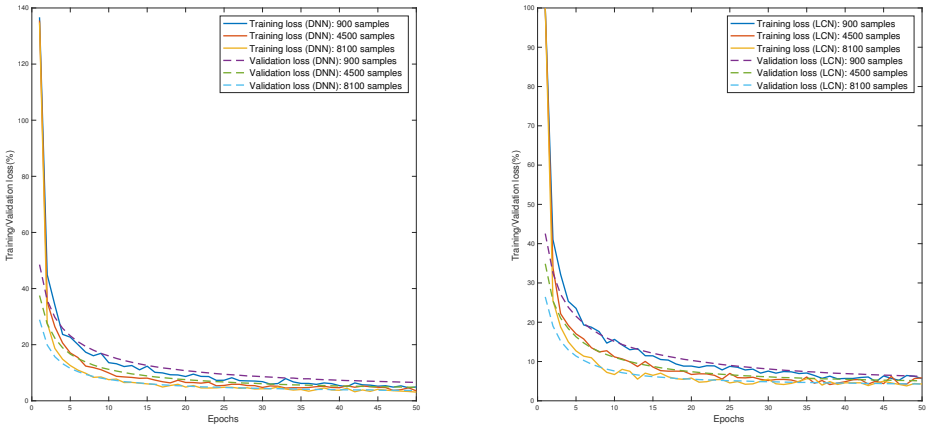
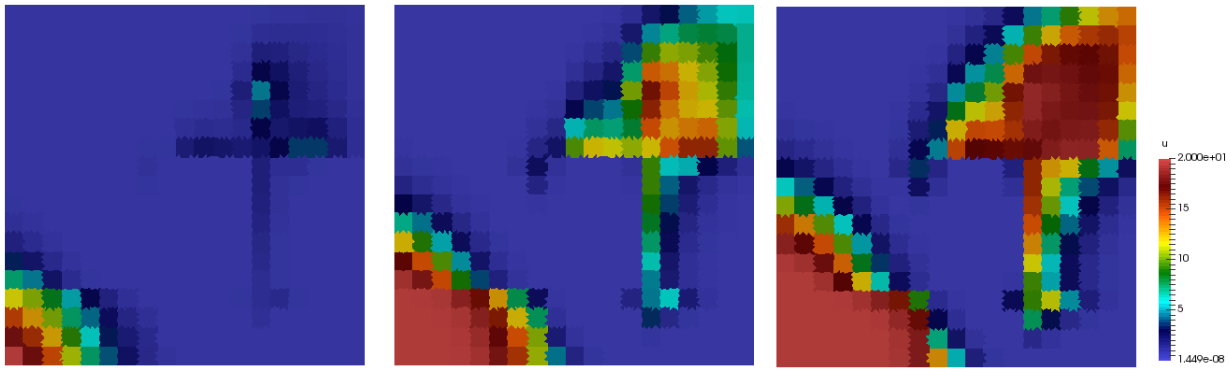


Figure 5.3: DNN (left) and Locally-connected Network (right) training/validation losses over epochs for $N_o(\cdot)$, with different number of samples.

we have $g = -10[(\sin(\alpha x))^2 + (\sin(\beta y))^2]$ denotes an production well, where the parameters α and β are randomly chosen in each time step, and are different among samples (which are obtained using these different source terms g). So, for each sample, we have different values of the source term, and, in each sample, the source term is time dependent.

For the computational data u_s, we use the solution obtained from NLMC model for n random source terms. We note that, $n = 200, 600$ and 1000 in this example, and we take solutions associated with $100, 500, 900$ sources out of them for training correspondingly, and the other 100 for validation/testing.



(a) Mobility at time $t = 0.01$ (b) Mobility at time $t = 0.05$ (c) Mobility at time $t = 0.1$

Figure 5.4: Illustration of mobility $\lambda(x, t)$.

For the observation data u_o, we first solve the system from the fine grid model using the same set of source terms, and use the coarse degrees of freedom fine solution average as the observation population. As for the mixture u_m of computational and observation data, we take the samples relating to half of the sources from u_s, and the samples relating to another half of the sources from u_o. In practice, to explain the mixture data u_m, we can assume we have the observation data in the whole domain given some well configurations, but for some other well configurations, we only have simulation results.

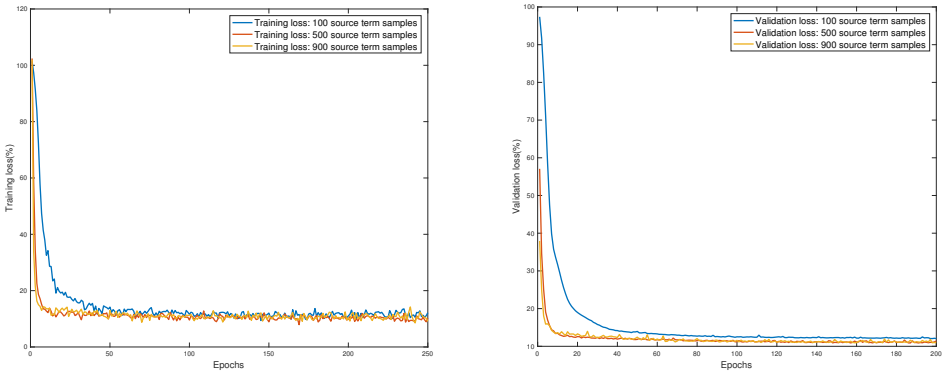
5.3.2.1 Full connection

We first build the three deep neural networks with densely connected layers. Due to the nonliearity of the underlying problem in this example, we will take 4 hidden layers with ReLU activation function, and use linear activation at the output layer. The input vector has dimension 445 as before, which is the degree of freedom in the NLMC model. The first hidden layer also has 445 neurons, and they are fully connected with the input. We take 50 neurons in the other three hidden layers. And the output has the dimension 445.

We solve the equation (5.3) with $T = 0.1$. Similarly, we use the solutions at time step 1 to time step 9 as input data, and solutions at time step 2 to time step 10 as output data. In this example, we have $900, 4500, 8100$ training sample pairs, respectively. The validation set then has 900 samples

in each case.

In the training process, we take the batch size to be 100. For the loss function, we use the relative L^2 error between the samples (computed from fine grid model/ NLMC model) and the predicted data (the output of the neural networks), same as in Example 1. We remark that (u^n, I^{n+1}) is taken to be $(u^n, \triangle t \cdot g^{n+1})$ in this examples, where g is the time dependent source term. The training and validation loss for the network $N_o(\cdot)$ are shown in Figure 5.5, the loss history for $N_s(\cdot)$ and $N_m(\cdot)$ are similar.



(a) Training loss (b) Validation loss

Figure 5.5: DNN training/validation loss vs. epochs for $N_o(\cdot)$, with different number of samples.

As we discussed before, we can use (5.10) or (5.11) to forward the solution from the initial time step to the final time step using the "universal" deep neural nets. Assume we have 10 time steps in total, for a given solution u_1 at the initial time, we will apply $\mathcal{N}_\alpha(\cdot)$ for $\alpha = o, m, s$ for 9 times to obtain the final time predictions. That is,

$$u^{10}_{pred,\alpha} = \underbrace{\mathcal{N}_\alpha \circ \mathcal{N}_\alpha \circ \cdots \circ \mathcal{N}_\alpha}_{\text{9 times}}(u^1)$$

for $\alpha = o, m, s$.

Finally, we compare the final time predictions $u^{10}_{pred,\alpha}$ (for $\alpha = o, m, s$) with the observation data u^{10}_o at the final time step given u^1_s. The results are shown in Table 5.3. There are 100 samples to test in total. As we increase the number of training samples, it is clear that $\left\|u^{10}_{pred,o} - u^{10}_o\right\|$ is decreasing. For $\left\|u^{10}_{pred,m} - u^{10}_o\right\|$, with the increasing size of the training sample, it will give better and better prediction results. When we take 900 samples, the mean of $\left\|u^{10}_{pred,m} - u^{10}_o\right\|$ over testing samples is 16.8%, which is very close to the reference case where $\left\|u^{10}_{pred,o} - u^{10}_o\right\|$ has a mean error of 13.5%. This indicates that using a mixture of computational data and observation data can enhance the performance of NLMC model induced by deep learning, as we compared the predictions to the observation data (which is the fine grid solution). We also show the comparison for one of the samples in Figure 5.6, where the solution produced by the network $N_m(\cdot)$ has almost the same accuracy as the solution produced by $N_o(\cdot)$, and is much more accurate than that of $N_s(\cdot)$.

	Relative Errors (%)		
Number of Training Samples	$\left\|u^{10}_{pred,o} - u^{10}_o\right\|$	$\left\|u^{10}_{pred,m} - u^{10}_o\right\|$	$\left\|u^{10}_{pred,s} - u^{10}_o\right\|$
900	20.2	30.6	31.1
4500	14.6	17.8	37.1
8100	13.5	16.8	45.9

Table 5.3: Example 2, fully connected network. Mean error between final time step prediction and true solutions over 100 testing samples for three different networks.

5.3.2.2 Sparse connection: Region of influence

In this section, we examine the region of influence input for this example. Different from Example 1, the DNN network here has more layers. We will just replace the first fully connected layer with a locally connected layer. In this section, we only show the results for the case that the training samples are generated corresponding to 500 different source terms. The other two sample sets perform similarly. The training and validation loss over some epochs are shown in Figure 5.7. Compared the fully connected network and locally connected network, we can see that the training

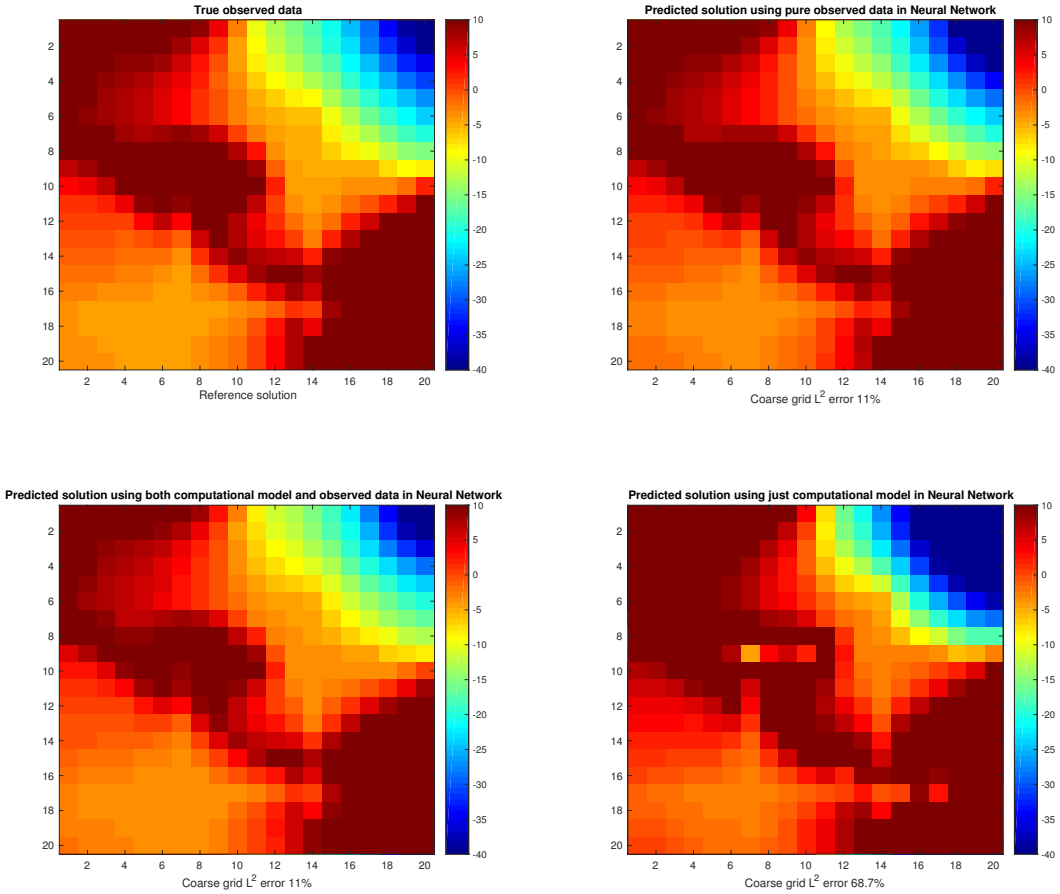
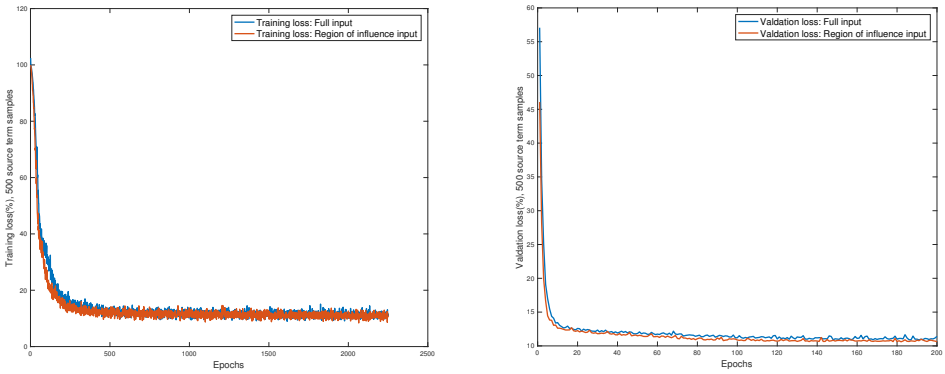


Figure 5.6: Example 2, fully connected network. DNN predicted solutions' comparison for one of the samples.

loss decays faster for locally connected network, and the validation loss have very similar behavior for both networks. We remark that, the number of trainable parameters in fully connected network is $318,315$, and that number in locally connected network is $154,422$, where we only replace the first layer by a self defined locally connected layer. All the other hyperparameters in both networks are chosen to be the same. The mean errors for the same testing samples as in the previous section are presented in Table 5.4. Compared with Table 5.3, we observe slightly better results are obtained by locally connected network.



(a) Training loss (b) Validation loss

Figure 5.7: Comparison of fully connected network and locally connected network. Training/validation loss vs. epochs for $N_o(\cdot)$, number of source samples is 500.

	Relative Errors (%)		
Number of Training Samples	$\|u^{10}_{pred,o} - u^{10}_o\|$	$\|u^{10}_{pred,m} - u^{10}_o\|$	$\|u^{10}_{pred,s} - u^{10}_o\|$
4500	13.7	17.9	33.5

Table 5.4: Example 2, locally connected network. Mean error between final time step prediction and true solutions over 100 testing samples for three different networks.

5.4 Conclusion

In this chapter, we use deep learning techniques to derive and modify upscaled models for nonlinear PDEs. In particular, we combine multiscale model reduction (NLMC) with deep learning techniques in obtaining better approximations of the underlying models, which takes into account observed data. We show that the regions of influence derived from upscaling concepts can lighten the neural network. Because of the coarseness of the upscaled model, the prediction is more robust and computationally inexpensive. On the other hand, incorporating observation data in the training can improve the coarse grid model. Numerical examples shows that DMML can obtain accurate approximations, which also honors the observed data.

In conclusion, we believe DMML can be used as a new coarse-grid model for complex nonlinear problems with observed data, where upscaling of the computational model is expensive and may not accurately represent the true observed model.

6. REDUCED-ORDER DEEP LEARNING FOR FLOW DYNAMICS

6.1 Introduction

In this chapter, we investigate applying neural networks to multiscale simulations and discuss a design of a novel deep neural network model reduction approach for multiscale problems.

The main contributions of the chapter are the following: (1) We study how neural network captures the important multiscale modes related to the features of the solution. (2) We relate ℓ_1 minimization to model reduction and derive a more robust network for our problems using a soft thresholding. (3) We suggest an efficient strategy for some class of nonlinear problems that arise in porous media applications. (4) We use multi-layer networks for combined time stepping and reduced-order modeling, where at each time step the appropriate important modes are selected. We remark that we can use observed data to learn multiscale model as in [61].

The chapter is organized as follows. Section 6.2 will be a preliminary introduction of the multiscale problem. Section 6.3 mainly focuses on discussions on the reduced-order neural network. The structure of the neural network is presented. Section 6.4 later discusses the proposed neural network from different aspects and proposes a way to conduct model reduction with the neural network coefficient. We also present the relation between the soft thresholding neural network and a ℓ_1 minimization problem in this section. Section 6.5 provides various numerical examples to verify the predictive power of our proposed neural network and provide support to the claims in Section 6.4. Lastly, the chapter is concluded with Section 6.6.

6.2 Preliminaries

We consider a nonlinear flow equation

$$\frac{\partial u}{\partial t} - \text{div}(\kappa(t, x, u)\nabla u) = g, \tag{6.1}$$

in the domain $(0,T)\times\Omega$. Here, Ω is the spatial domain, κ is the permeability coefficient which can have a multiscale dependence with respect to space and time, and g is a source function. The weak formulation of (6.1) involves finding $u\in H^1(\Omega)$ such that

$$\left(\frac{\partial u}{\partial t},v\right)+(\kappa(t,x,u)\nabla u,\nabla v)=(g,v),\quad \forall v\in H^1(\Omega). \tag{6.2}$$

If we numerically solve this problem in a m-dimensional approximation space $V_h\subset H^1(\Omega)$, and use an Euler temporal discretization, the numerical solution can be written as

$$u_h(t_n,x)=u_h^n=\sum_{j=1}^{m}\alpha_j^n\psi_j(x), \tag{6.3}$$

where $\{\psi_j\}_{j=1}^m$ is a set of basis for V_h. Moreover, the problem (6.2) can then be reformulated as: find $u_h\in V_h$ such that

$$\left(\frac{u_h^{n+1}-u_h^n}{\Delta t},v_h\right)+(\kappa(t_{n+1},x,u_h^{\nu})\nabla u_h^{n+1},\nabla v_h)=(f^{n+1},v_h),\quad \forall v_h\in V_h, \tag{6.4}$$

where $\nu=n$ or $n+1$ corresponds to linear and nonlinear system respectively. Here, $(\cdot,\cdot)$ denotes the L_2 inner product.

For problems with multiple scales, a multiscale basis function for each coarse node is computed following the idea of upscaling, i.e., the problem can be solved with a local model reduction. Instead of using the classic piece-wise polynomials as the basis functions, we construct the local multiscale basis following NLMC [61] and use the span of all such basis functions as the approximation space V_h. More specifically, for a fractured media (Figure 2.4), the basis functions of (6.1) can be constructed following NLMC as discussed in Section 2.2.2.

Thus, the solution coefficient $U^n=(\alpha_1^n,\alpha_2^n,\ldots,\alpha_m^n)^T$ satisfies the recurrence relation

$$U^{n+1}=(M+\Delta tA^{\nu})^{-1}(MU^n+\Delta tF^n), \tag{6.5}$$

where M and A^ν are the mass matrix and the stiffness matrix with respect to the NLMC basis $\{\psi_j\}_{j=1}^m$, ν can be n or $n+1$ depending on the temporal scheme that we use. We have

$$[M]_{ij} = \int_\Omega \psi_i(x)\psi_j(x)\,dx,$$
$$[A^\nu]_{ij} = \int_\Omega \kappa(t_n, x, u_h^\nu)\nabla\psi_i(x)\cdot\nabla\psi_j(x)\,dx.$$

We claim that a global model reduction can be conducted to the problem described above, as solution $u_h^n(x)$ in many cases can be sparse in V_h even if V_h is already a reduced-order space. For instance, $u_h(t_n, x)$ is strongly bonded to initial condition $u_h(t_0, x)$. It can be foreseen that if initial conditions are chosen from a small subspace of $V_h(\Omega)$, $u_h^n(x) = u_h(t_n, x)$ is also likely to accumulate somewhere in V_h. In other words, the distribution of coefficients U^n can hardly expand over the entire $\mathbb{R}^m$ space but only lies in a far smaller subspace.

Other physical restrictions to the problem could also narrow down the space of solution. This indicates that $u_h(t_n, x)$ can be closely approximated with less degrees of freedom compared to $dim(V_h)$. Section 6.3 and Section 6.4 will be discussing how to identify dominant modes in the space of U^n using a neural network.

6.3 Reduced-order Neural Network

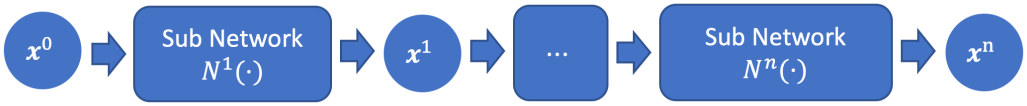
In this section, we present the construction of the reduced-order neural network. We propose a reduced-order neural network that can model a time series. Moreover, if there exists a basis that can represent the solutions for each time step with sparse coefficients, then the proposed neural network can identify such basis from the training samples. Specifically, Subsection 6.3.1 will discuss the macroscopic structure of the proposed neural network while Subsection 6.3.2 will discuss two designs of the sub-network of the network with more details. Subsection 6.3.3 later assembles the full multi-layer neural network.

6.3.1 Reduced-order Neural Network Structure

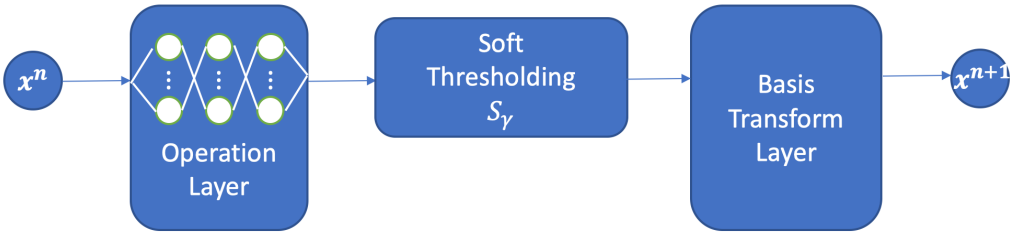
We propose a reduced-order neural network as shown in Figure 6.1a. Here, the full network is constituted by several sub-networks. Each sub-network $\mathcal{N}^n(\cdot)$ is expected to model a one-step

temporal evolution from $\mathbf{x}^n$ to $\mathbf{x}^{n+1}$ in a time series $\vec{\mathbf{x}} = [\mathbf{x}^0, \mathbf{x}^1, \cdots, \mathbf{x}^n]$.

Sub-networks should have a general structure as shown in Figure 6.1b. The specific design will vary depending on the problem we are modeling (see discussions in Subsection 6.3.2). The sub-network is built in a way that the input $\mathbf{x}^n$ will first be fed into a multi-layer fully-connected network named as "operation layer". This layer is intended for the neural network to capture the map between two solutions at consecutive time steps. The output of the operation layer is then fed into a soft thresholding function to impose sparsity to the solution coefficient. Lastly, the data will be processed with a "basis transform layer" in which a new basis set will be learned. With the new basis set, one can represent the solution with sparse coefficients assuming such representation exists.



(a) Multi-layer reduced-order neural network $\mathcal{NN}(\cdot)$



(b) Sub-network $\mathcal{N}^n(\cdot)$

Figure 6.1: Reduced-order neural network structure.

6.3.2 Sub-network

In this subsection, we present two designs of the sub-network $\mathcal{N}^n(\cdot)$. One is for modeling linear dynamics while the other is designed for nonlinear dynamics. One can choose from these

two options when assembling the full network depending on the dynamics of interest. Both sub-network designs are intended to learn a new set of basis and then impose the sparsity to the solution coefficient in the new system while learning dynamics.

6.3.2.1 Sub-network for linear process

We first present the sub-network for modeling linear dynamics. It can be used to model the one-step flow described in (6.1), where we define the sub-network for the n-th time step as

$$\mathcal{N}^n(\mathbf{x}^n;\theta_n) := W_n^2 S_\gamma(W_n^1 \cdot \mathbf{x}^n + b_n). \tag{6.6}$$

Here, for the sub-network parameter $\theta_n = (W_n^1, W_n^2, b_n)$, W_n^1 and b_n are in the operation layer and W_n^2 works as the basis transformation layer. S_γ is the soft thresholding function defined point-wise as $S_\gamma : \mathbb{R} \to \mathbb{R}$

$$S_\gamma(x) = \text{sign}(x)(|x| - \gamma)_+ \begin{cases} x - \gamma & \text{if } x \geq \gamma, \\ 0 & \text{if } -\gamma < x < \gamma, \\ x + \gamma & \text{if } x \leq -\gamma. \end{cases} \tag{6.7}$$

We further require W_n^2 to be an orthogonal matrix, i.e. $(W_n^2)^T \cdot (W_n^2) = I$. To this end, we train the network (6.6) with respect to the cost function

$$\mathcal{C}^n(\theta_n) = \|\mathbf{x}^{n+1} - \mathcal{N}^n(\mathbf{x}^n;\theta_n)\|_2^2 + \eta_n \|(W_n^2)^T \cdot (W_n^2) - I\|_1 \tag{6.8}$$

with a penalty on the orthogonality constraint of W_n^2. η is a hyper coefficient for adjusting the weight of the ℓ_1 regularization term. Here, $\mathbf{x}^n$ is the input of $\mathcal{N}^n(\cdot;\theta_n)$, and $\mathcal{C}^n(\cdot)$ measures the mismatch between true solution $\mathbf{x}^{n+1}$ and the prediction $\mathcal{N}^n(\mathbf{x}^n)$ while forcing W_n^2 to be orthogonal. We remark that this cost functions is only a part of the full cost function that we will be discussing in Subsection 6.3.3.

With such a design, the trained neural sub-network $\mathcal{N}^n(\cdot;\theta_n^*)$ will be able to model the input-output map specified by the training data while producing a matrix W_n^2 whose columns forms an

orthogonal basis in $\mathbb{R}^m$.

6.3.2.2 Sub-network for nonlinear process

Similar to the linear case, a sub-network for nonlinear process can also be designed. When the output $\mathbf{x}^{n+1}$ is non-linearly dependent on input $\mathbf{x}^n$, we make use of a d_n-layer feed-forward neural network $\tilde{\mathcal{N}}(\cdot)$ to approximate the input-output map. $\tilde{\mathcal{N}}(\cdot)$ will work as the operation layer of the sub-network. The output of $\tilde{\mathcal{N}}(\cdot)$ is then processed with soft-thresholding and a basis transformation layer W_n^2. We define the sub-network to be

$$\begin{aligned} \mathcal{N}^n(\mathbf{x}^n;\theta_n) &:= W_n^2 S_\gamma(\tilde{\mathcal{N}}(\mathbf{x}^n;\theta_n)), \\ \tilde{\mathcal{N}}(\mathbf{x}^n;\theta_n) &:= W_n^{1,d_n}(\cdots(\sigma(W_n^{1,3}\sigma(W_n^{1,2}\sigma(W_n^{1,1}\mathbf{x}^n + b_n^1) + b_n^2)\cdots + b_n^{d_n}), \end{aligned} \tag{6.9}$$

where $\theta_n = \left(W_n^{1,1}, W_n^{1,2}, \ldots, W_n^{1,d_n}, W_n^2, b_n^1, b_n^2, \ldots, b_n^{d_n}\right)$ is the sub-network parameter, and $\sigma(\cdot)$ is a nonlinear activation function. The cost function for training the sub-network parameter is again defined in (6.8). We also remark that if $d_n = 1$ and $\sigma = \mathbf{1}$ is the point-wise identify function, then the network structure (6.9) is reduced to (6.6).

Another approach to reduce the difficulty of reproducing a nonlinear process is clustering. Instead of using a single network to approximate complicated nonlinear relations, we use different networks for different data clusters as the clusters of the solutions can be predicted in many cases. Thus, separate the training samples by cluster can be an easy and effective way to accurately recover complicated process. Discussions on clustering and numerical examples are presented in Section 6.5.5.

6.3.3 Multi-layer Reduced Order Network

Now we construct the full neural network $\mathcal{NN}(\cdot;\theta)$ by stacking up the sub-network $\mathcal{N}^n(\cdot;\theta_n)$. More precisely, $\mathcal{NN}(\cdot;\theta) : \mathbb{R}^m \to \mathbb{R}^m$ is defined as:

$$\mathcal{NN}(\mathbf{x}^0;\theta) := \mathcal{N}^n(\cdots\mathcal{N}^1(\mathcal{N}^0(\mathbf{x}^0;\theta_0);\theta_1)\cdots;\theta_n), \tag{6.10}$$

where $\mathcal{N}^n(\cdot;\theta_n)$ is defined as in (6.6) or (6.9) depending the linearity of the process, and $\theta = (\theta_0, \theta_1, \ldots, \theta_n)$ is the full network parameter. We use such $\mathcal{NN}(\cdot;\theta)$ to approximate time series $\vec{\mathbf{x}} = [\mathbf{x}^0, \mathbf{x}^1, \cdots, \mathbf{x}^{n+1}]$. Denote the output of $(t+1)$-fold composition of the sub-network $\mathcal{N}^t(\cdot)$ as

$$\mathbf{o}^{t+1} := \mathcal{N}^t(\cdots \mathcal{N}^1(\mathcal{N}^0(\mathbf{x}^0;\theta_0);\theta_1)\cdots;\theta_t). \tag{6.11}$$

Then $\mathbf{o}^{t+1}$ works as a prediction of the solution at time $t+1$. The full cost function is then defined as

$$\mathcal{C}(\theta) = \sum_{t=0}^{n} \|\mathbf{o}^{t+1} - \mathbf{x}^{t+1}\|_2^2 + \eta_t \|(W_t^2)^T \cdot (W_t^2) - I\|_1, \tag{6.12}$$

where $\vec{\mathbf{x}}$ is the true time sequence, and η_t is a hyper-parameter stands for the weight of the regularizer while θ represents all tuneable parameters of $\mathcal{NN}(\cdot)$. Each layer (sub-network) corresponds to a one-step time evolution of the dynamics.

Suppose we have L training samples,

$$\vec{\mathbf{x}}_i = [(\mathbf{x}_i^0), (\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^{n+1})], \qquad 1 \le i \le L.$$

The optimal parameter θ^* of $\mathcal{NN}(\cdot)$ is then determined by optimizing the cost function $\mathcal{C}(\theta)$ subject to this training set $\{\vec{\mathbf{x}}_i\}_{i=1}^L$ as discussed Section 2.3. Once θ^* is decided, predictions can be made for testing samples $[\mathbf{x}_i^1, \cdots, \mathbf{x}_i^{n+1}]$ by $\mathcal{NN}(\mathbf{x}_i^0)$, for $i > L$.

6.4 Discussions and Applications

In this section, we discuss some theoretical aspects of the proposed neural networks. Specifically, we use the proposed network to model fluid dynamics in heterogeneous media, as described in (6.1). First, we relate the soft thresholding network with ℓ_1 optimization problem. Secondly, we explore how learned coefficients of neural network are related to the map that is being approximated. Thirdly, based on the understanding of the proposed neural network, we present a way to utilize the trained network coefficients to construct a reduced-order model.

Specifically, we consider a one-layer neural network for single-step linear dynamics

$$\mathcal{NN}(\mathbf{x};\theta) = \mathcal{N}(\mathbf{x};\theta) := W^2 S_\gamma(W^1\mathbf{x} + b), \tag{6.13}$$

omitting the indices for time step n. We train the neural network $\mathcal{NN}(\cdot;\theta)$ with data pairs $\{(U_i^n, U_i^{n+1})\}_{i=1}^L$, which are NLMC solution coefficients (see Section 2.2.2 for more details of data generation) to (6.1) at t^n and t^{n+1}, respectively. More precisely, we consider the linear case of (6.1), when $\kappa(t,x)$ is independent of u_h^n. In this case, the one step fluid dynamics (6.5) is indeed a linear revolution such that A^n is only dependent on time. Further, we denote $\hat{W}$ and $\hat{b}$ by

$$\begin{aligned} \hat{W} &:= (M + \Delta t A^n)^{-1} M, \\ \hat{b} &:= (M + \Delta t A^n)^{-1} \Delta t F^n. \end{aligned} \tag{6.14}$$

Then

$$U_i^{n+1} = \hat{W} U_i^n + \hat{b}, \quad 1 \le i \le L, \tag{6.15}$$

for all training samples that we use in this section. We further define the linear map between U_i^n and U_i^{n+1} as $\hat{\mathcal{L}}(\cdot) : \mathbb{R}^m \to \mathbb{R}^m$

$$\hat{\mathcal{L}}(\mathbf{x}) := \hat{W}\mathbf{x} + \hat{b}. \tag{6.16}$$

From now on, when there is no risk of confusion, we drop the optimal network parameter θ^* in the trained network $\mathcal{NN}(\cdot)$. We expect $\mathcal{NN}(\cdot)$ to learn the map $\hat{\mathcal{L}}(\cdot)$ from the data while extracting a system W^2 in which the data U_i^{n+1} can be represented sparsely.

6.4.1 Sparsity and ℓ_1 Minimization

To understand the trained neural network $\mathcal{NN}(\cdot)$, we first assume the following.

Assumption 1. *For a subspace $S \subset \mathbb{R}^m$, there exist some orthogonal matrix $\hat{W}^2 \in \mathbb{R}^{m\times m}$ such that $(\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})$ can be approximated by a sparse vector in $\mathbb{R}^m$ for any $\mathbf{x} \in S$. More precisely, there exist an approximation error function $\epsilon(\cdot) : \mathbb{N} \to \mathbb{R}^+$, such that for any $\mathbf{x} \in S$, there exist a*

corresponding s-sparse vector $\mathbf{y}_s \in \mathbb{R}^m$ satisfies

$$\|(\hat{W}^2)^T \hat{\mathcal{L}}(\mathbf{x}) - \mathbf{y}_s\|_2 \leq \epsilon(s). \tag{6.17}$$

We then take $\{U_i^n\}_{i=1}^L$ from S and define $U_i^{n+1,\hat{W}^2} := (\hat{W}^2)^T U_i^{n+1}$ for all training pairs(U_i^n, U_i^{n+1}) of $\mathcal{NN}(\cdot)$. By Assumption 1, there exist an s-sparse vector $U_i^s \in \mathbb{R}^m$ such that

$$||U_i^{n+1,\hat{W}^2} - U_i^s||_2 \leq \epsilon(s) \tag{6.18}$$

for $1 \leq i \leq L$.

Additionally, for any orthogonal matrix $W^2 \in \mathbb{R}^{m\times m}$, we define $U_i^{n+1,W^2} := (W^2)^T U_i^{n+1}$ and denote it as $[\alpha_{i,1}^{n+1,W^2}, \alpha_{i,2}^{n+1,W^2}, \cdots, \alpha_{i,m}^{n+1,W^2}]^T$. Recall (6.3), and the corresponding numerical solution $u_{h,i}^{n+1}$ at time step $n+1$ can be written as

$$u_{h,i}^{n+1} = \Psi U_i^{n+1},$$

letting $\Psi = [\psi_1, \psi_2, \cdots, \psi_m]$ be the multiscale basis functions, and U_i^{n+1} be the corresponding coefficient vector. By the orthogonality of W^2, $u_{h,i}^{n+1}$ can be further written as

$$u_{h,i}^{n+1} = \Psi(W^2) U_i^{n+1,W^2} = \Psi^{W^2} U_i^{n+1,W^2}, \tag{6.19}$$

with Ψ^{W^2} defined by

$$\Psi^{W^2} := \Psi W^2. \tag{6.20}$$

We further denote the columns of Ψ^{W^2} as $\psi_j^{W^2}$'s. Then $\{\psi_j^{W^2}\}_{j=1}^m$ is actually a new set of multiscale basis functions such that $u_{h,i}^{n+1}$ can be written as

$$u_{h,i}^{n+1} = \sum_{j=1}^{m} \alpha_{i,j}^{n+1,W^2} \psi_j^{W^2}. \tag{6.21}$$

If W^2 is taken to be $\hat{W}^2$ as in Assumption 1, we will obtain

$$u_{h,i}^{n+1} = \sum_{j=1}^{m} \alpha_{i,j}^{n+1,\hat{W}^2} \psi_j^{\hat{W}^2},$$

where the coefficient $U_i^{n+1,\hat{W}^2} = [\alpha_{i,1}^{n+1,\hat{W}^2}, \alpha_{i,2}^{n+1,\hat{W}^2}, \cdots, \alpha_{i,m}^{n+1,\hat{W}^2}]^T$ can be closely approximated by a sparse vector U_i^s.

We then claim that our proposed neural network $\mathcal{NN}(\cdot)$ is able to approximate such $\hat{W}^2$ and a sparse approximation of the output from data. This is guaranteed by the following lemma from [62]:

Lemma 6.4.1. *We define* $\hat{\mathcal{N}}(\cdot), \mathbb{R}^m \to \mathbb{R}^m$ *as*

$$\hat{\mathcal{N}}(\mathbf{x}) := S_\gamma(W\mathbf{x} + b).$$

The output of $\hat{\mathcal{N}}(\mathbf{x})$ *is the solution to the* ℓ_1*-regularized problem*

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathbb{R}^m}{\textit{argmin}} \frac{1}{2}\|\mathbf{y} - (W\mathbf{x} + b)\|_2^2 + \gamma\|\mathbf{y}\|_1 \tag{6.22}$$

by proximal gradient update.

Proof. It is straightforward to see that the directional derivative of the residual with respect to $\mathbf{y}$ is given by $\mathbf{y} - (W\mathbf{x} + b)$. On the other hand, the soft thresholding operator is the proximal operator of the ℓ_1 norm, i.e.

$$S_\gamma(x) = \text{prox}_{\gamma\|\cdot\|_1}(\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^m}{\text{argmin}} \left(\frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 + \gamma\|\mathbf{y}\|_1 \right).$$

With a zero initial guess, the 1-step proximal gradient update of (6.22) with a step size γ is therefore

$$\mathbf{y}^* = \hat{\mathcal{N}}(\mathbf{x}) = S_\gamma(W\mathbf{x} + b).$$

□

Thus, for the one-step neural network $\mathcal{NN}(\cdot)$ defined in (6.13), Lemma 6.4.1 implies that

$$\mathcal{NN}(\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^m}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{y} - W^2W^1\mathbf{x} + W^2b\|_2^2 + \gamma\|(W^2)^T\mathbf{y}\|_1. \tag{6.23}$$

That is to say, the output of the trained neural network $\mathcal{NN}(\cdot)$ is actually the solution to a ℓ_1 optimization problem. We further define the linear operator $\mathcal{L}(\cdot) : \mathbb{R}^m \to \mathbb{R}^m$ from the coefficients of $\mathcal{NN}(\cdot)$ as

$$\mathcal{L}(\mathbf{x}) := W^2W^1\mathbf{x} + W^2b. \tag{6.24}$$

Equation (6.23) actually implies that

$$\mathcal{L}(\cdot) \approx \mathcal{NN}(\cdot) \tag{6.25}$$

as $\mathcal{NN}(\mathbf{x})$ minimizes $\|\mathbf{y} - W^2W^1\mathbf{x} + W^2b\|_2^2$. Moreover, the output of $\mathcal{NN}(\cdot)$ is sparse in the coordinate system W^2 as it also minimizes the $\|(W^2)^T\mathbf{y}\|_1$ term.

S_γ is widely used in ℓ_1-type optimization for promoting sparsity and extracting important features as discussed above. It is therefore also brought into neural network to extract sparsity from the training data. For other network defined with activation functions such as ReLU, we also remark there's an correlation between S_γ and ReLU. Recall its definition in (6.7) :

$$S_\gamma(x) = \text{sign}(x)(|x| - \gamma)_+ \begin{cases} x - \gamma, & \text{if } x \geq \gamma, \\ 0, & \text{if } -\gamma < x < \gamma, \\ x + \gamma, & \text{if } x \leq -\gamma. \end{cases}$$

and that of ReLU : $\mathbb{R} \to \mathbb{R}$

$$\text{ReLU}(x) = \max\{x, 0\} = \begin{cases} x, & \text{if } x \geq 0, \\ 0, & \text{if } x < 0. \end{cases}$$

We can explicitly represent the soft thresholding operator S_γ by the ReLU function as

$$S_\gamma(x) = \text{ReLU}(x - \gamma) - \text{ReLU}(-x - \gamma), \quad \forall x \in \mathbb{R}, \tag{6.26}$$

or in an entry-wise sense, one can write

$$S_\gamma(\mathbf{x}) = J_m \text{ReLU}(J_m^T \mathbf{x} - \gamma \mathbf{1}_{2m}), \quad \forall \mathbf{x} \in \mathbb{R}^m, \tag{6.27}$$

where $J_m = [I_m, -I_m]$. Activation functions $S_\gamma(\cdot)$ can thus be easily implemented with the help of ReLU. Further, it also means that our proposed neural network is only a special class of neural networks that are defined with ReLU.

6.4.2 Linear Operator $\hat{\mathcal{L}} \approx \mathcal{NN}$

For neural network $\mathcal{NN}(\cdot)$ as defined in (6.13)

$$\mathcal{NN}(\mathbf{x}) = W^2 S_\gamma(W^1 \mathbf{x} + b),$$

we claim the following

Lemma 6.4.2. *We assume Assumption 1 holds. Then, there exist a set of parameters* $(W^1, W^2, b) \in \mathbb{R}^{m \times m} \times \mathbb{R}^{m \times m} \times \mathbb{R}^m$ *such that*

$$\|\mathcal{NN}(\mathbf{x}) - \hat{\mathcal{L}}(\mathbf{x})\|_2 \leq 2\epsilon(s) + s^{\frac{1}{2}}\gamma, \quad \forall x \in S. \tag{6.28}$$

Proof. By Assumption 1, there exist some orthogonal matrix $\hat{W}^2 \in \mathbb{R}^{m \times m}$ such that for all $\mathbf{x} \in S$,

we have

$$\|(\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x}) - \mathbf{y}_s\|_2 \leq \epsilon(s), \tag{6.29}$$

where $\mathbf{y}_s$ is an s-sparse vector. Next, we consider $W^1 = (\hat{W}^2)^T\hat{W}$, $W^2 = \hat{W}^2$ and $b = (\hat{W}^2)^T\hat{b}$. We recall the definition of $\hat{\mathcal{L}}(\cdot)$

$$\hat{\mathcal{L}}(\mathbf{x}) := \hat{W}\mathbf{x} + \hat{b}.$$

The difference between $\mathcal{NN}(\mathbf{x})$ and $\hat{\mathcal{L}}(\mathbf{x})$ can then be estimated by

$$\begin{aligned}
\|\mathcal{NN}(\mathbf{x}) - \hat{\mathcal{L}}(\mathbf{x})\|_2 &= \|\hat{W}^2 S_\gamma\Big((\hat{W}^2)^T\hat{W}\mathbf{x} + (\hat{W}^2)^T\hat{b}\Big) - \hat{\mathcal{L}}(\mathbf{x})\|_2 \\
&= \|\hat{W}^2\Big(S_\gamma\Big((\hat{W}^2)^T\hat{W}\mathbf{x} + (\hat{W}^2)^T\hat{b}\Big) - (\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\Big)\|_2 \\
&= \|\hat{W}^2\Big(S_\gamma\Big((\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\Big) - (\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\Big)\|_2 \\
&= \|S_\gamma\Big((\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\Big) - (\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\|_2.
\end{aligned}$$

Since $|S_\gamma(z_2) - S_\gamma(z_1)| \leq |z_2 - z_1|,\ \forall z_1, z_2 \in \mathbb{R}$, we have

$$\|S_\gamma\Big((\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\Big) - S_\gamma(\mathbf{y}_s)\|_2 \leq \|(\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x}) - \mathbf{y}_s\|_2 \leq \epsilon(s).$$

Thus, we obtain

$$\begin{aligned}
&\|S_\gamma\Big((\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\Big) - (\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\|_2 \\
\leq&\|S_\gamma\Big((\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\Big) - S_\gamma(\mathbf{y}_s)\|_2 + \|S_\gamma(\mathbf{y}_s) - \mathbf{y}_s\|_2 + \|\mathbf{y}_s - (\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})\|_2 \\
\leq&2\epsilon(s)_+ \|\mathbf{y}_s - S_\gamma(\mathbf{y}_s)\|.
\end{aligned}$$

Since $|(S_\gamma(\mathbf{y}_s) - \mathbf{y}_s)_i| \leq \gamma$, we have

$$\|S_\gamma(\mathbf{y}_s) - \mathbf{y}_s\|_2^2 = \sum_{(\mathbf{y}_s)_i \neq 0} |(S_\gamma(\mathbf{y}_s) - \mathbf{y}_s)_i|^2 \leq s\gamma^2,$$

and therefore we have

$$\|\mathcal{NN}(\mathbf{x}) - \hat{\mathcal{L}}(\mathbf{x})\|_2 \leq 2\epsilon(s) + s^{\frac{1}{2}}\gamma,$$

letting $W^1 = (\hat{W}^2)^T\hat{W}$, $W^2 = \hat{W}^2$ and $b = (\hat{W}^2)^T\hat{b}$. □

Since $\mathcal{NN}(\cdot)$ is trained with (U_i^n, U_i^{n+1}), where $U_i^n \in S$ and $\hat{\mathcal{L}}(U_i^n) = U_i^{n+1}$, we have

$$\mathcal{NN}(\mathbf{x}) \approx \hat{\mathcal{L}}(\mathbf{x}), \quad \mathbf{x} \in S. \tag{6.30}$$

More specifically, this approximation error $\|\mathcal{NN}(\mathbf{x}) - \hat{\mathcal{L}}(\mathbf{x})\|_2$ is small for all $\mathbf{x} \in S$ providing sufficient training. Therefore, by Lemma 6.4.2, we claim the trained parameters closely approximate the optimal choice to guarantee the small error indicated in (6.28) , i.e.

$$W^2 \approx \hat{W}^2, \quad W^2W^1 \approx \hat{W}, \quad \text{and} \quad W^2b \approx \hat{b}.$$

However, due to the high dimension of $\hat{W}$, full recovery of $\hat{W}$ requires enormous number of training and is thus impractical. However, by enforcing $\mathcal{NN}(\mathbf{x}) = \hat{\mathcal{L}}(\mathbf{x})$ for $\mathbf{x} \in S$, the neural network learns a set of parameters $W^2W^1 \neq \hat{W}$, and $W^2b \neq \hat{b}$ such that they function similarly as $\hat{\mathcal{L}}(\cdot)$ on the subset S in the sense of linear operator. A validation of this is later provided in Subsection 6.5.1. Recall definition of $\mathcal{L}(\cdot)$ in (6.24) and the fact that it can approximate $\mathcal{NN}(\cdot)$ as in (6.25), we claim the linear operator have the following property:

$$\mathcal{L}(\mathbf{x}) \approx \hat{\mathcal{L}}(\mathbf{x}) \quad \forall \mathbf{x} \in S. \tag{6.31}$$

In the following subsection, we further construct a reduced-order model with the help of $\mathcal{L}(\cdot)$.

6.4.3 Model Reduction with W^2

In this subsection, we further assume the s-sparse vector $\mathbf{y}_s$ in Assumption 1 has non-zero entries only at fixed coordinates for all $\mathbf{x}$ in S. That is to say, we have a fix reordering $\{j_k\}_{k=1}^m$ for $\{1, 2 \cdots m\}$, such that $(\mathbf{y}_s)_{j_k} = 0$ for $s+1 \leq k \leq m$.

Then, we will be able to utilize the coordinate system $W^2 \approx \hat{W}^2$ learned through training network to construct a reduced-order operator $\mathcal{L}_s(\cdot)$, such that it can approximate the linear map $\hat{\mathcal{L}}(\cdot)$ and maps $\mathbf{x}$ in S to a s-sparse vector in $\mathbb{R}^m$. To do so, we first define $\mathcal{L}(\cdot)$ from the learned coefficients of $\mathcal{NN}(\cdot)$ as in last subsection, and $\mathcal{L}_s(\cdot)$ will be exactly a truncation of it.

Moreover, let the new basis set $\{\psi_j^{W^2}\}_{j=1}^m$ be defined with trained coefficient W^2 as in (6.20). When truncating W^2 in $\mathcal{L}(\cdot)$, we also determine the dominant basis among $\{\psi_j^{W^2}\}_{j=1}^m$ simultaneously. Thus, we can view the model reduction from another aspect that we actually drop the basis with less significance and represent the solution with only the dominant multiscale modes.

To construct such $\mathcal{L}_s(\cdot)$, we follow the steps:

1. Find the dominant coordinates of the outputs $\mathcal{NN}(\cdot)$ in the system W^2.

   (a) Compute W^2 system coefficients of $\mathcal{NN}(U_i^n)$ for all training samples by

   $$O_i^{n+1,W^2} := (W^2)^T \mathcal{NN}(U_i^n), \quad 1 \leq i \leq L,$$

   where i refers to the sample index. Notice $(W^2)^T \mathcal{NN}(\mathbf{x})$ is sparse for $\mathbf{x} \in S$ by (6.23), therefore O_i^{n+1,W^2} is also sparse.

   (b) Calculate the quadratic mean of $\{O_i^{n+1,W^2}\}_{i=1}^L$ over all samples, coordinate by coordinate:

   $$S_j := \frac{1}{L}(\sum_{i=1}^{L} |O_{i,j}^{n+1,W^2}|^2)^{\frac{1}{2}}, \quad 1 \leq j \leq m.$$

   (c) Sort the quadratic mean value S_j in descending order and denoted the reordered sequence as $\{S_{j_k}\}_{k=1}^m$.

2. Keep the dominant j_k-th columns of W^2 for $k = 1, \cdots, s$. Then let the rest columns be zero. Thus, we construct a reduced-order coordinate system $W^{2,s} \in \mathbb{R}^{m \times m}$. Consequently, $\mathbf{y} = \hat{\mathcal{L}}(\mathbf{x})$ for any $\mathbf{x} \in S$ can be approximated with the reduced-order system W^2 as an

s-sparse vector $\mathbf{y}^{W^{2,s}} := (W^{2,s})^T\mathbf{y}$, and

$$\mathbf{y}^{W^{2,s}} \approx (W^2)^T\mathbf{y}. \tag{6.32}$$

As a result, for training/testing samples, we have $U_i^{n+1,W^{2,s}} \approx U_i^{n+1,W^2}$. Thus, the function $u_{h,i}^{n+1}$ can be approximated with only basis $\{\psi_{j_k}^{W_2}|1 \leq k \leq s\}$ correspond to dominant multiscale modes.

$$u_{h,i}^{n+1} \approx \Psi^{W^2}U_i^{n+1,W^{2,s}} = \sum_{k=1}^{s} \alpha_{i,j_k}^{n+1,W^2}\psi_{j_k}^{W^2}. \tag{6.33}$$

3. We finally define the reduced linear operator $\mathcal{L}_s(\cdot) : \mathbb{R}^m \to \mathbb{R}^m$ as

$$\mathcal{L}_s(\mathbf{x}) = W^{2,s}W^1 \cdot \mathbf{x} + W^{2,s}b, \quad \mathbf{x} \in \mathbb{R}^m. \tag{6.34}$$

Here, the output of $\mathcal{L}_s(\cdot)$ is an s-sparse vector in $\mathbb{R}^m$.

This algorithm is designed based on the fact that $O_i^{n+1,W^2} \approx (W^2)^T U_i^{n+1} = U_i^{n+1,W^2}$ as $\mathcal{NN}(\cdot)$ is fully trained with (U_i^n, U_i^{n+1}). Thus, the order of S_j can reflect not only the significance of the coordinates of the output of $\mathcal{NN}(\cdot)$ but also that of $(W^2)^T\mathcal{L}(\mathbf{x})$ for all $\mathbf{x}$ in S. Moreover, the existence of the sparse approximation $\mathbf{y}_s$ to $(\hat{W}^2)^T\hat{\mathcal{L}}(\mathbf{x})$ as described in Assumption 1 guarantees the effectiveness of the ordering.

We then claim that this reduced-order linear operator $\mathcal{L}_s(\cdot)$ can approximate the true input-output map $\hat{\mathcal{L}}(\cdot)$ on S: Since $\mathcal{L}_s(\cdot)$ is simply a truncation of $\mathcal{L}(\cdot)$, we have:

$$\mathcal{L}_s \longrightarrow \mathcal{L}, \quad \text{as } s \to m. \tag{6.35}$$

Moreover, recall (6.31)

$$\hat{\mathcal{L}}(\mathbf{x}) \approx \mathcal{L}(\mathbf{x}), \quad \forall \mathbf{x} \in S,$$

it implies the following

$$\hat{\mathcal{L}}(\mathbf{x}) \approx \mathcal{L}_s(\mathbf{x}), \quad \forall \mathbf{x} \in S. \tag{6.36}$$

This property of $\mathcal{L}_s(\cdot)$ provides us a way to represent the projected vectors $\hat{\mathcal{L}}(\mathbf{x})$ for $\mathbf{x} \in S$ using a vector with only s nonzero coefficients, which corresponds to a reduced multiscale model to represent the class of solution u_h^{n+1} that we are interested in.

Numerical examples are presented in Subsection 6.5.3 to verify this claim, from which we actually observe that s can be taken as a fraction of the original number of multiscale basis m to give the approximation in (6.36).

6.5 Numerical Examples

In this section, we present numerical examples in support of the previous discussions on the reduced-order neural network. Specifically, Subsection 6.5.1 demonstrates that the $\mathcal{L}(\cdot)$ and $\hat{\mathcal{L}}(\cdot)$ functions similarly on a subspace by comparing the eigenvalues of the two operators; Subsection 6.5.2 later shows that a one-layer soft thresholding neural network can accurately recover a linear dynamics with a sparse coefficient vector; Subsection 6.5.3 then uses the learned coefficient W^2 from the one-layer neural network to conduct model reduction as described in Subsection 6.4.3; Subsection 6.5.4 later presents the predicting results for multi-layer reduced-order neural network which corresponds to Subsection 6.3.3; and Subsection 6.5.5 applies the clustering scheme to the nonlinear process modeling. All training are performed using the Python deep learning API TensorFlow [63].

6.5.1 $\hat{\mathcal{L}} \approx \mathcal{L}$ in a Subspace of $\mathbb{R}^m$

We recall that the one-layer neural network for a single-step linear dynamics in (6.13)

$$\mathcal{NN}(\mathbf{x}) := W^2 S_\gamma(W^1 \mathbf{x} + b),$$

and the definition of $\hat{\mathcal{L}}(\cdot)$ and $\mathcal{L}(\cdot)$ in (6.16) and (6.24) respectively:

$$\hat{\mathcal{L}}(\mathbf{x}) := \hat{W}\mathbf{x} + \hat{b}, \qquad \mathcal{L}(\mathbf{x}) = W^2 W^1 \mathbf{x} + W^2 b,$$

where $\hat{W}$ and$\hat{b}$ are defined as in (6.14), while W^2, W^1 are trained parameters of $\mathcal{NN}(\cdot)$. We also recall (6.31):

$$\mathcal{L}|_S \approx \hat{\mathcal{L}}|_S,$$

for $S \subset \mathbb{R}^m$.

To support this claim, we design a special subspace $S \subset \mathbb{R}^m$. For $r < m$, we then let

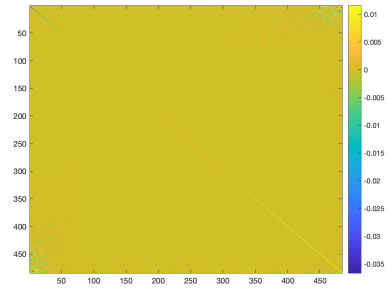
$$S = V_r := \text{span}\{v_i, 1 \leq i \leq r\} \quad \subset \quad \text{span}\{v_i, 1 \leq i \leq m\} = \mathbb{R}^m, \tag{6.37}$$

where $\{v_i\}_{i=1}^m$ are eigenvectors of $\hat{W}$ corresponding to eigenvalues λ_i in descending order. We also define matrix V as

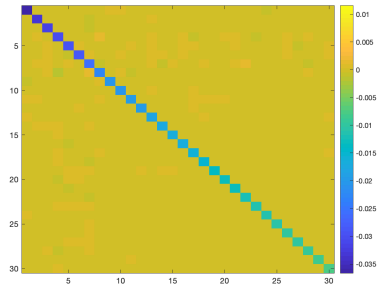
$$V := [v_1, v_2, \cdots, v_m]. \tag{6.38}$$

We then randomly pick training input $U \in V_r$ such that $U = \sum_{i=1}^r c_i v_i$. The $\mathcal{NN}(\cdot)$ is then trained with $(U, \hat{\mathcal{L}}(U))$-like training pairs, and we obtain a corresponding operator $\mathcal{L}(\cdot)$ with the trained coefficients. The linear operators of $\mathcal{L}(\cdot)$ and $\hat{\mathcal{L}}(\cdot)$ are compared by their eigenvalues, i.e. $V^T\hat{W}V$ and $V^TW^2W^1V$. By the definition of V, the former will exactly be a diagonal matrix with λ_i be its diagonal value. We expect W^2W^1 functions similarly to $\hat{W}$ on V_r, and further the r-by-r sub-matrix of $V^TW^2W^1V$ should be similar to that of $V^T\hat{W}V$.

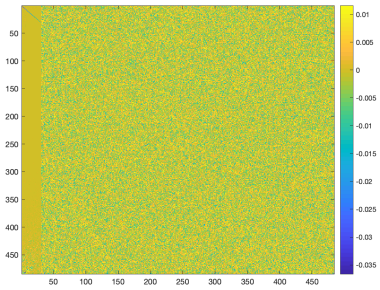
Figure 6.2 compares $V^TW^2W^1V$ and $V^T\hat{W}V$ for the case when V_r is constructed letting $r = 30$. We can tell that the first 30×30 submatrix are very much alike. That is to say, despite the fact that the operator $\mathcal{L}(\cdot)$ and $\hat{\mathcal{L}}(\cdot)$ are different on $\mathbb{R}^m$, their behavior on the subspace V_r are the same. Moreover, Figure 6.3, shows that such similarity only exist in V_r as for the i_{th} diagonal values of $V^TW^2W^1V$ and that of $V^T\hat{W}V$ distinct when $i > s$. This also makes sense as the operator $\mathcal{L}(\cdot)$ is defined from the trained parameters of $\mathcal{NN}(\cdot)$ where only subspace V_r is visible to the network.



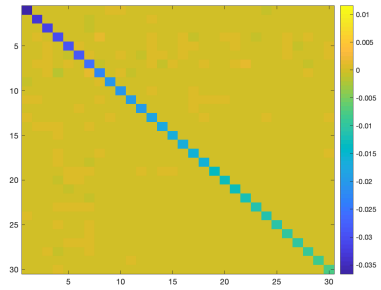
(a) $V^T \hat{W} V$



(b) 30×30 sub-matrix of $V^T \hat{W} V$



(c) $V^T W^2 W^1 V$



(d) 30×30 sub-matrix of $V^T W^2 W^1 V$

Figure 6.2: $\hat{W}$ and $W^2 W^1$ function similarly on S, where $r = 30$.

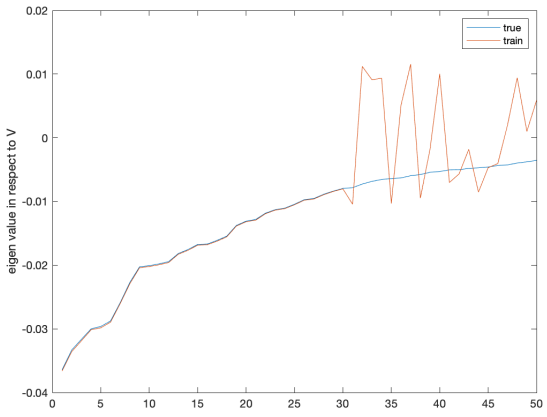


Figure 6.3: Comparison of the eigen-values of $\hat{W}$ and $W^2 W^1$ when $r = 30$.

6.5.2 One-layer Reduced Order Neural Network

In this example, we consider the one-layer reduced-order neural network as defined in (6.13). We use this neural network to predict one-step fluid dynamics, where the data are taken to be NLMC solution coefficients to (6.1) in the form of (U_i^0, U_i^1). We fix $\kappa(t,x)$ and $f(t,x)$ among samples, thus all data describes linear dynamics for different initial conditions.

We take 2% out of all data pairs as testing samples and the remaining 98% as training samples and use only the training sample to train $\mathcal{NN}(\cdot)$. We then evaluate the neural network by examining the accuracy of the following approximation for the unseen testing samples:

$$U^1 \approx \mathcal{NN}(U^0).$$

The ℓ_2 relative error of the prediction is computed by

$$\frac{||U^1 - \mathcal{NN}(U^0)||_2}{||U^1||_2}. \tag{6.39}$$

Table 6.1 is the error table for the case when we use 500 data pairs with 490 to be training samples and 10 to be testing samples. These data are generated with different choices of initial condition U_i^0. To match the realistic physical situation, we took all initial conditions to be the NLMC terminal pressure of a mobility driven nonlinear flow process.

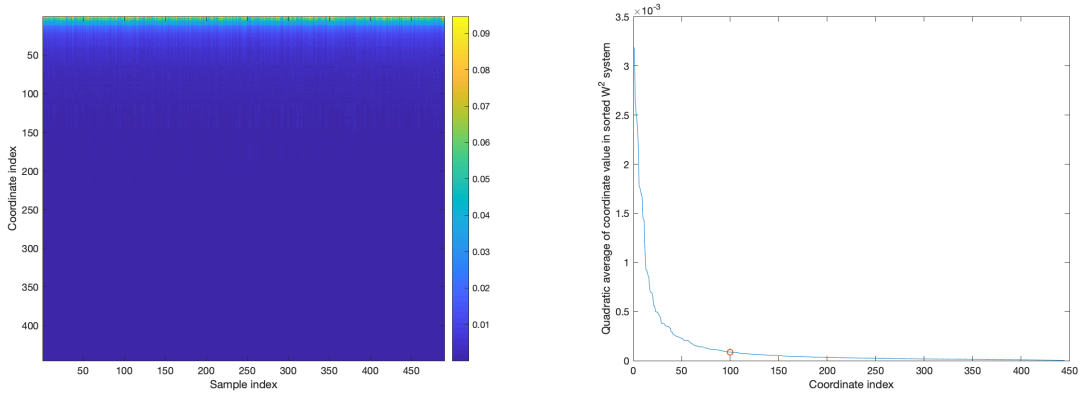
From Table 6.1, we can see that the prediction of our proposed network $\mathcal{NN}(U^0)$ is rather effective with an average ℓ_2 error of 5.34%.

We also verify that U^1 is sparse in the learned W^2-system for all data (training and testing). We first reorder the columns of W^2 by their dominance as discussed in Subsection 6.4.3, then compute the corresponding W^2-system coefficients U^{1,W^2}, which should be a roughly decreasing vector. From Figure 6.4, we can tell that the W^2-system coefficients U^{1,W^2} are sparse. This can be an reflection of successful learning of $\hat{W^2}$ in Assumption 1. Moreover, only a few dominant modes are needed to recover the solution as the quadratic averages of coordinates $|U_j^{1,W^2}|$ decays

Sample Index	Error(%)
#1	0.25
#2	0.43
#3	10.02
#4	9.91
#5	3.90
#6	8.18
#7	17.27
#8	1.57
#9	1.13
#10	0.76
Mean	5.34

Table 6.1: ℓ_2 relative error of $\mathcal{NN}(\cdot)$ prediction.

fast when $j > 100$.



(a) U^{1,W^2} for all data samples

(b) quadratic average of U^{1,W^2} among all data samples

Figure 6.4: Sparsity of the output of $\mathcal{NN}(U^0)$ in W^2 .

To summarize, the proposed neural network can indeed learn the dominant multiscale modes needed to represent U^1 from training data while properly reproducing the map between U^0 and U^1.

6.5.3 Model Reduction with W^2

As discussed in Subsection 6.4.3, we would like to use the reduced-order system $W^{2,s}$ to further conduct a model reduction. The reduced-order solution coefficient is defined with the reduced-order linear operator $\mathcal{L}_s(\cdot)$:

$$U_{sN}^{n+1} = \mathcal{L}_s(U^n). \tag{6.40}$$

We notice that U_{sN}^{n+1} is the coefficient of u_h^{n+1} in the original basis system $\{\psi_j\}_{j=1}^m$ and it is sparse in W^2-system. In fact, U_{sN}^{n+1,W^2} is sparse and has a maximum s nonzero elements.

The numerical experiments is conducted based on a one-layer neural network as defined in (6.13) for one-step linear dynamics. We would like to compare the following coefficient vectors:

$$U_{sN}^1 := \mathcal{L}_s(U^0) = W^{2,s}W^1 \cdot U^0 + W^{2,s}b,$$

$$U_{\text{true}}^1 = \hat{W}U^0 + \hat{b},$$

and

$$U_N^1 := \mathcal{NN}(U^0).$$

Here U_{true}^1 is the true solution to (6.15), while U_N^1 is the prediction of $\mathcal{NN}(\cdot)$.

Figure 6.5 shows the error decay of U_{sN}^1 compared to U_{true}^1. As s grows, the error gets smaller. This figure actually verified (6.36), i.e. the reduced operator $\mathcal{L}_s(\cdot)$ can approximate $\hat{\mathcal{L}}(\cdot)$. Moreover, this approximation gets more accurate as s gets larger. The error in Figure 6.5 at $s = m = 445$ is also an expected consequence of the training error. Besides, we observe that the error decays fast when $s > 40$ for our training samples.

Table 6.2 further facilitates such conclusion. The errors of U_{sN}^1 in testing samples are less than 12% when $s \geq 80$. We can thus represent the multiscale solution u_h^1 using $s = 80$ basis with little sacrifice in the solution accuracy. We notice that the order of the reduce operator $\mathcal{L}_s(\cdot)$ is only around 18% that of the original multiscale model.

We lastly present the comparison between U_{true}^1, U_N^1, and U_{sN}^1. From Table 6.3, we can tell that

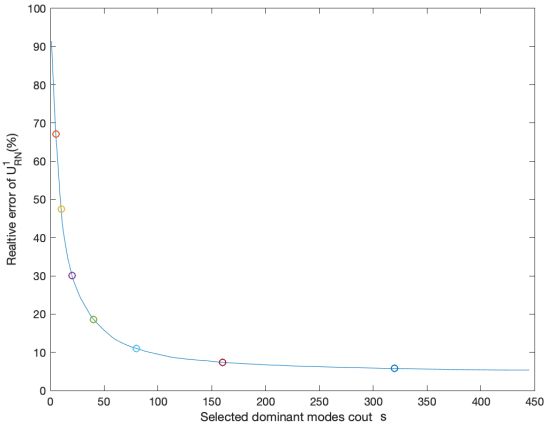


Figure 6.5: Decay of relative error $\frac{||U^1_{sN}-U^1_{\text{true}}||_2}{||U^1_{\text{true}}||_2}$ as s grows for training samples.

Sample Index \ s	5	10	20	40	80	160	320
#1	66.71	46.80	29.02	17.01	8.44	3.53	1.11
#2	66.55	46.58	29.03	17.41	9.48	4.62	1.59
#3	66.35	46.77	29.70	18.59	12.26	10.92	10.16
#4	67.95	48.58	31.23	19.87	12.59	10.03	9.90
#5	66.22	46.31	28.99	17.73	10.49	6.49	4.34
#6	66.47	46.76	29.40	17.89	10.86	9.12	8.33
#7	69.59	51.33	36.01	27.42	22.33	18.62	17.29
#8	66.60	46.61	28.81	16.39	7.47	3.86	1.96
#9	66.90	47.13	29.21	16.94	8.02	3.58	1.67
#10	67.14	47.39	29.44	17.23	8.23	3.19	1.32
Mean	67.05	47.43	30.08	18.65	11.02	7.40	5.77

Table 6.2: Decay of error $\frac{||U^1_{sN}-U^1_{\text{true}}||_2}{||U^1_{\text{true}}||_2}$ with respect to number of selected dominant modes in $W^{2,s}$ for testing samples.

for a single testing sample, we have

$$\frac{||U^1_{sN} - U^1_{\text{true}}||_2}{||U^1_{\text{true}}||_2} > \frac{||U^1_N - U^1_{\text{true}}||_2}{||U^1_{\text{true}}||_2}, \quad \frac{||U^1_{sN} - U^1_{\text{true}}||_2}{||U^1_{\text{true}}||_2} > \frac{||U^1_{sN} - U^1_N||_2}{||U^1_N||_2},$$

which are as expected since $\frac{||U^1_N - U^1_{\text{true}}||_2}{||U^1_{\text{true}}||_2}$and $\frac{||U^1_{sN} - U^1_N||_2}{||U^1_N||_2}$ are exactly the two components of error $\frac{||U^1_{sN} - U^1_{\text{true}}||_2}{||U^1_{\text{true}}||_2}$. These components stand for neural network prediction error and $\mathcal{L}_s$ truncation error, respectively. The latter can be reduced by increasing s, while the former one can only be improved with more effective training.

Sample Index	$\frac{\|U^1_N - U^1_{\text{true}}\|_2}{\|U^1_{\text{true}}\|_2}$	$\frac{\|U^1_{sN} - U^1_{\text{true}}\|_2}{\|U^1_{\text{true}}\|_2}$	$\frac{\|U^1_{sN} - U^1_N\|_2}{\|U^1_N\|_2}$
#1	0.25	6.48	6.41
#2	0.43	7.65	7.55
#3	10.02	11.59	5.81
#4	9.91	11.28	5.83
#5	3.90	8.93	8.38
#6	8.18	10.00	5.80
#7	17.27	20.98	10.25
#8	1.57	5.85	5.83
#9	1.13	6.13	6.03
#10	0.76	6.17	6.12
Mean	5.34	9.50	6.80

Table 6.3: Relative error percentage of solutions obtained in full W^2 system and reduced-order system $W^{2,s}$ for $s = 100$.

6.5.4 Multi-layer Reduced Order Neural Network

In this example, we use a multi-layer reduced-order neural network $\mathcal{NN}(\cdot)$ to predict multi-step fluid dynamics. Recall (6.10), it is defined as

$$\mathcal{NN}(\mathbf{x}^0) := \mathcal{N}^n(\cdots \mathcal{N}^1(\mathcal{N}^0(\mathbf{x}^0))).$$

The input of $\mathcal{NN}(\cdot)$ is taken to be U^0, the initial condition, while the outputs are the collection of outputs at n_{th}-layer sub-network $\mathcal{N}^n(\cdot)$ which correspond to the true values $[U^1, U^2, \cdots, U^9]$. Here, U^{n+1} are all taken to be the NLMC solutions of (6.1) at time step n for $n = 0, \cdots, 8$. Prediction accuracy is measured with the ℓ_2 relative error that is defined similar to (6.39).

Sample Index	U^1	U^2	U^3	U^4	U^5	U^6	U^7	U^8	U^9
#1	1.62	1.17	1.69	1.91	1.95	1.92	1.92	1.91	1.96
#2	3.33	1.86	2.10	1.98	2.15	1.53	1.04	0.94	0.77
#3	11.62	13.32	9.39	9.57	8.89	10.36	11.67	11.86	12.97
#4	9.74	9.00	4.21	3.45	3.46	3.17	2.93	2.87	2.81
#5	5.63	4.68	2.65	2.28	2.52	1.89	1.49	1.74	1.91
#6	9.54	11.50	9.30	9.70	9.14	10.50	11.73	12.00	13.09
#7	21.46	14.82	5.42	4.24	3.27	5.06	6.52	6.77	7.81
#8	5.72	1.40	0.67	0.77	1.11	1.49	1.95	2.54	3.20
#9	4.03	2.16	3.21	3.66	3.47	4.36	5.15	5.35	5.97
#10	4.62	1.01	3.14	3.84	3.81	4.47	5.05	5.24	5.68
Mean	7.73	6.09	4.18	4.14	3.98	4.47	4.95	5.12	5.62

Table 6.4: ℓ_2 relative error (%) of prediction of U^{n+1} using $\mathcal{NN}(\cdot)$.

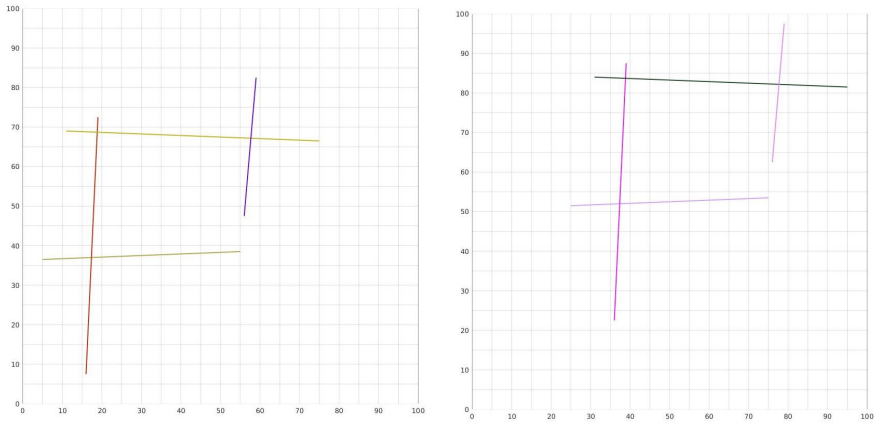
In Table 6.4, the columns show the prediction error for $U^{n+1}, n = 0, 1, \cdots 8$ where the average error is computed for all time steps among testing samples which are less than 10% in average. Therefore, we claim that the proposed multi-layer reduced order neural network $\mathcal{NN}(\cdot)$ is effective in the aspect of prediction. We also claim that the coefficients U^{n+1,W_n^2} for $n = 0, 1, \cdots, 8$ are sparse in the independent systems W_n^2. These systems are again learned simultaneously by training $\mathcal{NN}(\cdot)$.

6.5.5 Clustering

In this experiment, we aim to model the fluid dynamics correspond to two different fractured media as shown in Figure 6.6. More specifically, the permeability coefficient of matrix region is $\kappa_m = 1$ and the permeability of the fractures is $\kappa_f = 10^3$. The one-step NLMC solution pairs $(U_i^0, U_i^1), 1 \leq i \leq L$, generated following these two different configurations of fractures are then

referred as "Cluster 1" and "Cluster 2" (see Figure 6.7 for an illustration). We will then compare the one-step prediction of networks $\mathcal{NN}_1, \mathcal{NN}_2$ with that of $\mathcal{NN}_{\text{mixed}}$. The inputs are taken to be U_i^0, which are the terminal solutions of mobility driven 10-step nonlinear dynamics, while the output is an approximation of U_i^1.

$\mathcal{NN}_1, \mathcal{NN}_2$ and $\mathcal{NN}_{\text{mixed}}$ share the same one-layer soft thresholding neural network structure as in (6.13) while the first two network are trained with data for each cluster separately and the latter one is trained with the mixed data from two clusters.

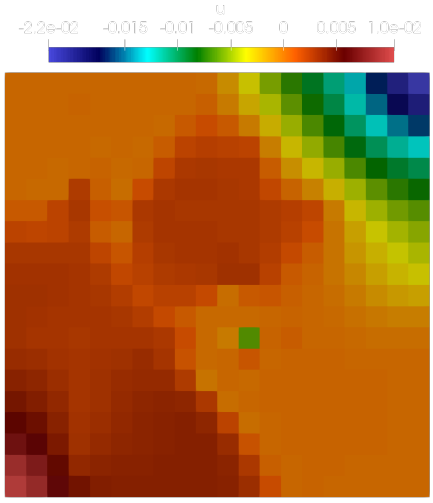


(a) Cluster 1 (b) Cluster 2

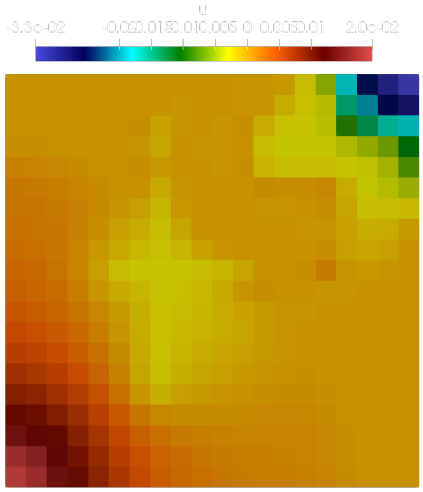
Figure 6.6: Fracture networks for two clusters.

Figure 6.6 shows the fracture networks we use to generate data for two clusters, while Figure 6.7 shows an example of solution U^0 and U^1 for each cluster (sample index i is omitted for simplicity). As observed from the profiles of U^0 and U^1 for both clusters, we can see that the solutions to Cluster 1 and Cluster 2 are very different due to the translation of the fractures. Moreover, since the data resulted from both clusters have non-uniform map between U^0 and U^1, the mixed data set can be considered as obtained from a nonlinear map.

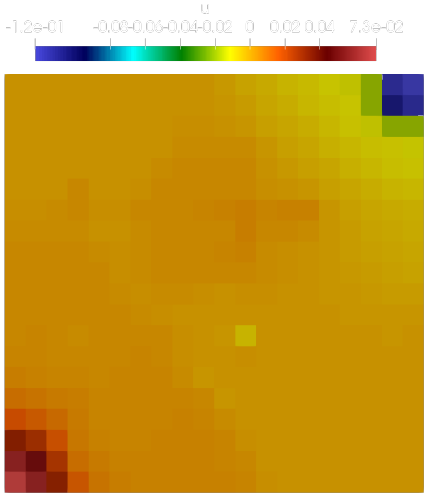
Table 6.5 demonstrates the comparison of the prediction accuracy when the network is fed with a single data cluster and when the network is fed with mixed data. This simple treatment can significantly improve the accuracy. For Cluster 1, the average prediction error of $\mathcal{NN}_1$ is around



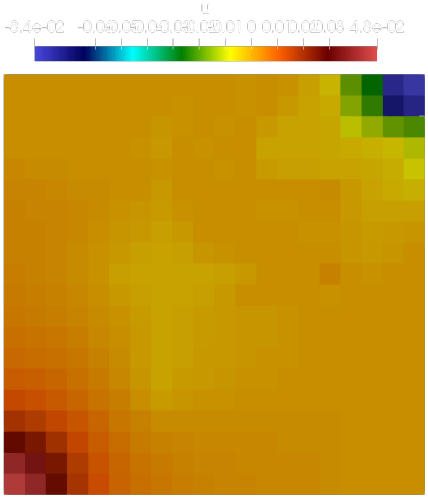
(a) Coarse-scale NLMC solution u^0 – Cluster 1



(b) Coarse-scale solution of pressure u^0 – Cluster 2



(c) Coarse-scale solution of pressure u^1 – Cluster 1



(d) Coarse-scale solution of pressure u^1 – Cluster 2

Figure 6.7: Example solution pairs for two different clusters.

Sample Index	Cluster 1 Error	Cluster 2 Error
#1	0.87	0.26
#2	3.19	0.45
#3	13.62	0.79
#4	12.36	0.49
#5	7.64	0.43
#6	10.69	1.00
#7	19.08	0.53
#8	2.57	0.92
#9	0.77	0.46
#10	3.70	0.38
Mean	7.45	0.57

(a) Relative ℓ_2 prediction error(%) of $\mathcal{N}_1$ and $\mathcal{N}_2$.

Sample Index	Cluster 1 Error	Cluster 2 Error
# 1	36.25	84.83
#2	35.08	84.20
#3	28.20	111.05
#4	44.60	69.35
#5	32.32	89.30
#6	30.03	106.41
#7	48.61	57.76
#8	35.63	88.46
#9	36.89	88.67
#10	38.78	83.85
Mean	36.64	86.39

(b) Relative ℓ_2 prediction error(%) of $\mathcal{N}_{\text{mixed}}$.

Table 6.5: Prediction error of $\mathcal{N}_1$, $\mathcal{N}_2$ and $\mathcal{N}_{\text{mixed}}$.

7.45% while that of $\mathcal{NN}_{\text{mixed}}$ is around 36.64%. Similar contrast can also be observed for Cluster 2.

6.6 Conclusion

In this chapter, we discussed a novel deep neural network approach of model reduction for multiscale problems. To numerically solve the multiscale problems, a fine mesh needs to be used but leads to large degrees of freedom. To this end, non-local multicontinuum (NLMC) upscaling [37] is used as a dimensionality reduction technique. In flow dynamics problems, multiscale solutions at consecutive time instants are regarded as an input-output mechanism and learnt with the deep neural networks techniques.

By exploiting a relation between a soft-thresholding neural network and a ℓ_1 minimization problem, multiscale features of the coarse-grid solutions are extracted using neural networks. This provides us with a new network-based construction of a reduced-order model, which involves extracting appropriate important modes at each time step. We also suggest an efficient strategy for a class of nonlinear flow problems. Finally, we present numerical examples to validate the effectiveness of our method.

7. SUMMARY AND CONCLUSIONS

In this thesis, our discussions mainly focused on two topics. The application of current multiscale methods and the development of deep learning techniques. In specific, in Chapter 3 and Chapter 4, we considered elliptic equations in heterogeneous domain. Chapter 3 targeted on designing a coarse solver that can hierarchically treat the fractured and vuggy domain while Chapter 4 concentrated on analyzing the impacts of the geometry to the multiscale model. In both chapters, we obtained accurate solutions with the reduced-order model following GMsFEM and reduced the computational cost significantly.

In Chapter 5 and Chapter 6, we investigated the neural networks applied to multiscale simulations and discussed designs of a novel deep neural network model reduction approach for multiscale problems. In Chapter 5, low-order models are used to construct sparsely connected neural networks. We formulated and learned input-output maps constructed with NLMC on a coarse grid. Further, observation data are included to fine-tone the learned model. Chapter 6, on the other hand, focused on deepening the order reduction of the multiscale model. By relating the input-output optimization to l_1 minimization of PDE solutions, we proposed a multi-layer networks with a soft thresholding activation function. Such neural network can learn the forward multiscale operators in a reduced way by selecting the important multiscale features for modeling the underlying flow. For a class of nonlinear problems, we also suggested clustering for effective modeling.

In all chapters, numerical examples are presented to confirm the success of our proposed method along with in-depth discussions and rigid analysis. We finally conclude that our proposed methods indeed improved the existing multiscale methods and some has been extended to more complicated engineering applications. The novel neural network architecture has the potential to be extended to broader areas.

REFERENCES

[1] R. Baca, R. Arnett, and D. Langford, "Modelling fluid flow in fractured-porous rock masses by finite-element techniques," *International Journal for Numerical Methods in Fluids*, vol. 4, no. 4, pp. 337–348, 1984.

[2] L. J. Durlofsky, "Numerical calculation of equivalent grid block permeability tensors for heterogeneous porous media," *Water resources research*, vol. 27, no. 5, pp. 699–708, 1991.

[3] B. Hassani and E. Hinton, "A review of homogenization and topology optimization i—homogenization theory for media with periodic structure," *Computers & Structures*, vol. 69, no. 6, pp. 707–717, 1998.

[4] I. Bilionis and N. Zabaras., "Solution of inverse problems with limited forward solver evaluations: a bayesian perspective," *Inverse Problems*, vol. 30, no. 015004, 2013.

[5] X. Ma, M. Al-Harbi, A. Datta-Gupta, and Y. Efendiev, "A multistage sampling approach to quantifying uncertainty during history matching geological models," *SPE Journal*, vol. 13, no. 10, pp. 77–87, 2008.

[6] I. Bilionis, N. Zabaras, B. A. Konomi, and G. Lin., "Multi-output separable gaussian process: Towards an efficient, fully bayesian paradigm for uncertainty quantification," *Journal of Computational Physics*, vol. 241, pp. 212–239, 2013.

[7] A. Mondal, Y. Efendiev, B. Mallick, and A. Datta-Gupta, "Bayesian uncertainty quantification for flows in heterogeneous porous media using reversible jump Markov Chain Monte-Carlo methods," *Adv. Water Resour.*, vol. 33, no. 3, pp. 241–256, 2010.

[8] Y. Zhu and N. Zabaras, "Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification," *Journal of Computational Physics*, vol. 366, pp. 415–447, 2018.

[9] A.-M. Matache and C. Schwab, "Two-scale fem for homogenization problems," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 36, no. 04, pp. 537–572, 2002.

[10] T. Arbogast, J. Douglas, Jr, and U. Hornung, "Derivation of the double porosity model of single phase flow via homogenization theory," *SIAM Journal on Mathematical Analysis*, vol. 21, no. 4, pp. 823–836, 1990.

[11] Y. Efendiev, J. Galvis, and T. Hou, "Generalized multiscale finite element methods (gmsfem)," *Journal of Computational Physics*, vol. 251, pp. 116–135, 2013.

[12] I. Lunati and P. Jenny, "The multiscale finite volume method: A flexible tool to model physically complex flow in porous media," in *10th European Conference on the Mathematics of Oil Recovery*, (Amsterdam, The Netherlands), 2006.

[13] G. Allaire and R. Brizzi, "A multiscale finite element method for numerical homogenization," *SIAM J. Multiscale Modeling and Simulation*, vol. 4, no. 3, pp. 790–812, 2005.

[14] Y. Efendiev, J. Galvis, and X. Wu, "Multiscale finite element methods for high-contrast problems using local spectral basis functions," *Journal of Computational Physics*, vol. 230, pp. 937–955, 2011.

[15] T. Arbogast, "Implementation of a locally conservative numerical subgrid upscaling scheme for two-phase Darcy flow," *Comput. Geosci*, vol. 6, pp. 453–481, 2002.

[16] E. T. Chung, Y. Efendiev, and G. Li, "An adaptive GMsFEM for high contrast flow problems," *J. Comput. Phys.*, vol. 273, pp. 54–76, 2014.

[17] D. L. Brown and D. Peterseim, "A multiscale method for porous microstructures," *arXiv preprint arXiv:1411.1944*, 2014.

[18] E. Chung, Y. Efendiev, and S. Fu, "Generalized multiscale finite element method for elasticity equations," *International Journal on Geomathematics*, vol. 5(2), pp. 225–254, 2014.

[19] A. Abdulle and Y. Bai, "Adaptive reduced basis finite element heterogeneous multiscale method," *Comput. Methods Appl. Mech. Engrg.*, vol. 257, pp. 203–220, 2013.

[20] M. Drohmann, B. Haasdonk, and M. Ohlberger, "Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation," *SIAM J. Sci. Comput.*, vol. 34, no. 2, pp. A937–A969, 2012.

[21] J. Fish and W. Chen, "Space–time multiscale model for wave propagation in heterogeneous media," *Computer Methods in applied mechanics and engineering*, vol. 193, no. 45, pp. 4837–4856, 2004.

[22] J. Fish and R. Fan, "Mathematical homogenization of nonperiodic heterogeneous media subjected to large deformation transient loading," *International Journal for numerical methods in engineering*, vol. 76, no. 7, pp. 1044–1064, 2008.

[23] H. Owhadi and L. Zhang, "Metric-based upscaling," *Comm. Pure. Appl. Math.*, vol. 60, pp. 675–723, 2007.

[24] P. Henning and M. Ohlberger, "The heterogeneous multiscale finite element method for elliptic homogenization problems in perforated domains," *Numerische Mathematik*, vol. 113, no. 4, pp. 601–629, 2009.

[25] E. T. Chung, Y. Efendiev, W. Leung, M. Vasilyeva, and Y. Wang, "Online adaptive local multiscale model reduction for heterogeneous problems in perforated domains," *Applicable Analysis*, vol. 96, no. 12, pp. 2002–2031, 2017.

[26] E. Chung, M. Vasilyeva, and Y. Wang, "A conservative local multiscale model reduction technique for stokes flows in heterogeneous perforated domains," *Journal of Computational and Applied Mathematics*, vol. 321, pp. 389–405, 2017.

[27] E. Chung, Y. Efendiev, and W. T. Leung, "Generalized multiscale finite element method for wave propagation in heterogeneous media," *SIAM Multicale Model. Simul.*, vol. 12, pp. 1691–1721, 2014.

[28] E. Chung and W. T. Leung, "A sub-grid structure enhanced discontinuous galerkin method for multiscale diffusion and convection-diffusion problems," *Communications in Computational Physics*, vol. 14, pp. 370–392, 2013.

[29] Y. Efendiev, J. Galvis, and X.-H. Wu, "Multiscale finite element methods for high-contrast problems using local spectral basis functions," *Journal of Computational Physics*, vol. 230, no. 4, pp. 937–955, 2011.

[30] Y. Efendiev and T. Hou, *Multiscale Finite Element Methods: Theory and Applications*, vol. 4 of *Surveys and Tutorials in the Applied Mathematical Sciences*. New York: Springer, 2009.

[31] T. Hou and X. Wu, "A multiscale finite element method for elliptic problems in composite materials and porous media," *J. Comput. Phys.*, vol. 134, pp. 169–189, 1997.

[32] P. Jenny, S. Lee, and H. Tchelepi, "Multi-scale finite volume method for elliptic problems in subsurface flow simulation," *J. Comput. Phys.*, vol. 187, pp. 47–67, 2003.

[33] Y. Efendiev, J. Galvis, and T. Y. Hou, "Generalized multiscale finite element methods (gmsfem)," *Journal of Computational Physics*, vol. 251, pp. 116–135, 2013.

[34] Y. Efendiev, S. Lee, G. Li, J. Yao, and N. Zhang, "Hierarchical multiscale modeling for flows in fractured media using generalized multiscale finite element method," *GEM-International Journal on Geomathematics*, vol. 6, no. 2, pp. 141–162, 2015.

[35] M. Wang, C. Wei, H. Song, Y. Efendiev, Y. Wang, *et al.*, "Generalized multiscale coupling of triple-continuum model and discrete fracture network for carbonate reservoir simulation," in *SPE Annual Technical Conference and Exhibition*, Society of Petroleum Engineers, 2017.

[36] E. T. Chung, Y. Efendiev, T. Leung, and M. Vasilyeva, "Coupling of multiscale and multicontinuum approaches," *GEM-International Journal on Geomathematics*, vol. 8, no. 1, pp. 9–41, 2017.

[37] E. T. Chung, Efendiev, W. T. Leung, M. Vasilyeva, and Y. Wang, "Non-local multi-continua upscaling for flows in heterogeneous fractured media," *arXiv preprint arXiv:1708.08379*, 2018.

[38] E. T. Chung, Y. Efendiev, and W. T. Leung, "Constraint energy minimizing generalized multiscale finite element method," *arXiv preprint arXiv:1704.03193*, 2017.

[39] Y. Efendiev, T. Hou, and V. Ginting, "Multiscale finite element methods for nonlinear problems and their applications," *Comm. Math. Sci.*, vol. 2, pp. 553–589, 2004.

[40] V. Calo, Y. Efendiev, J. Galvis, and M. Ghommem, "Multiscale empirical interpolation for solving nonlinear pdes using generalized multiscale finite element methods," Submitted.

[41] M. Alotaibi, V. M. Calo, Y. Efendiev, J. Galvis, and M. Ghommem, "Global–local nonlinear model reduction for flows in heterogeneous porous media," *Computer Methods in Applied Mechanics and Engineering*, vol. 292, pp. 122–137, 2015.

[42] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[44] C. D. Manning, C. D. Manning, and H. Schütze, *Foundations of statistical natural language processing*. MIT press, 1999.

[45] Z. Li and Z. Shi, "Deep residual learning and pdes on manifold," *arXiv:1708.05115.*, 2017.

[46] E. Weinan and B. Yu, "The deep ritz method: A deep learning-based numerical algorithm for solving variational problems," *Communications in Mathematics and Statistics*, vol. 6, no. 1, pp. 1–12, 2018.

[47] Y. Khoo, J. Lu, and L. Ying, "Solving parametric pde problems with artificial neural networks," *arXiv:1707.03351*, 2017.

[48] S. W. Cheung, E. T. Chung, Y. Efendiev, E. Gildin, and Y. Wang, "Deep global model reduction learning," *arXiv preprint arXiv:1807.09335*, 2018.

[49] V. Calo, Y. Efendiev, J. Galvis, and G. Li, "Randomized oversampling for generalized multiscale finite element methods," *arXiv preprint, arXiv: 1409.7114*, 2014.

[50] E. T. Chung, Y. Efendiev, W. T. Leung, M. Vasilyeva, and Y. Wang, "Non-local multi-continua upscaling for flows in heterogeneous fractured media," *Journal of Computational Physics*, 2018.

[51] M. Wang, S. W. Cheung, E. T. Chung, Y. Efendiev, W. T. Leung, and Y. Wang, "Prediction of discretization of gmsfem using deep learning," *arXiv preprint arXiv:1810.12245*, 2018.

[52] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2018.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[54] J. Warren, P. J. Root, *et al.*, "The behavior of naturally fractured reservoirs," 1963.

[55] L. K. Thomas, T. N. Dixon, R. G. Pierson, *et al.*, "Fractured reservoir simulation," *Society of Petroleum Engineers Journal*, vol. 23, no. 01, pp. 42–54, 1983.

[56] G. i. P. Panasenko, *Multi-scale modelling for structures and composites*, vol. 615. Springer, 2005.

[57] P.-A. Raviart and J.-M. Thomas, "A mixed finite element method for 2-nd order elliptic problems," in *Mathematical aspects of finite element methods*, pp. 292–315, Springer, 1977.

[58] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, PMLR, 2011.

[59] D. P. Kingma and J. Ba., "Adam: A method for stochastic optimization.," *arXiv preprint arXiv:1412.6980*, 2014.

[60] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[61] Y. Wang, S. W. Cheung, E. T. Chung, Y. Efendiev, and M. Wang, "Deep multiscale model learning," *arXiv preprint arXiv:1806.04830*, 2018.

[62] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[63] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[64] I. Farago, "Finite element method for solving nonlinear parabolic equations," *Computers & Mathematics with Applications*, vol. 21, no. 1, pp. 59–69, 1991.

APPENDIX

A.1 Proofs of error estimates

In this section, we present the proofs of the error estimates in Theorem 3.3.1 and Theorem 3.3.2.

A.1.1 Proof of Theorem 3.3.1

Proof. Using (3.14) and (3.17), we have

$$b(\frac{\partial(u-u_{\text{ms}})}{\partial t},v)+\sum_{1\le i<N}(a^i(u_i,v^i;u_i)-a^i(u^i_{\text{ms}},v^i;u^i_{\text{ms}}))+q(u-u_{\text{ms}},v)=0 \qquad \forall v\in V_{\text{ms}},\ t\in(0,T).$$

Let $w\in V_{\text{ms}}$ and take $v=w-u_{\text{ms}}$, we have

$$\begin{aligned}&b(\frac{\partial(w-u_{\text{ms}})}{\partial t},w-u_{\text{ms}})+q(w-u_{\text{ms}},w-u_{\text{ms}})-\sum_{1\le i<N}a^i(u^i_{\text{ms}},w^i-u^i_{\text{ms}};u^i_{\text{ms}})\\ =&b(\frac{\partial(w-u)}{\partial t},w-u_{\text{ms}})+q(w-u,w-u_{\text{ms}})-\sum_{1\le i<N}a^i(u^i,w^i-u^i_{\text{ms}};u^i)\end{aligned}$$

From this equation ,we can further get the following by the definition of a^i and the bounded condition (3.18) of $a(u)$,

$$\begin{aligned}&b(\frac{\partial(w-u_{\text{ms}})}{\partial t},w-u_{\text{ms}})+\alpha^-q(w-u_{\text{ms}},w-u_{\text{ms}})+\alpha^-|w-u^i_{\text{ms}}|^2_a\\ \le&|b(\frac{\partial(w-u)}{\partial t},w-u_{\text{ms}})|+\alpha^+|q(w-u,w-u_{\text{ms}})|+\alpha^+|w-u|_a|w-u_{\text{ms}}|_a\\ &+\sum_{1\le i<N}\int_\Omega|(\alpha(u^i_{\text{ms}})-a(u_i))\kappa^i\nabla u_i\cdot\nabla(w^i-u^i_{\text{ms}})|\,dx\end{aligned}$$

By Cauchy-Schwarz Inequality, this implies

$$\begin{aligned}
&\frac{1}{2}\frac{d}{dt}\|w-u_{\text{ms}}\|_b^2+\alpha^-\|w-u_{\text{ms}}\|_{aQ}^2\\
\leq&\|\frac{\partial(w-u)}{\partial t}\|_b\|w-u_{\text{ms}}\|_b+\alpha^+\|w-u\|_{aQ}\|w-u_{\text{ms}}\|_{aQ}\\
&+\sum_{1\leq i<N}\int_\Omega|(\alpha(u_{\text{ms}}^i)-\alpha(u^i))\kappa^i\nabla u^i\cdot\nabla(w^i-u_{\text{ms}}^i)|\,dx
\end{aligned}\tag{A.1}$$

The last term on the right-hand side of (A.1) can be written as

$$\begin{aligned}
&\int_\Omega|(\alpha(u_{\text{ms}}^i)-\alpha(u^i))\kappa^i\nabla u^i\cdot\nabla(w^i-u_{\text{ms}}^i)|\,dx\\
=&\int_{\Omega_M}|(\alpha(u_{\text{ms}}^i)-\alpha(u^i))\kappa^i\nabla u^i\cdot\nabla(w^i-u_{\text{ms}}^i)|\,dx\\
&+\sum_s\int_{\Omega_{F,s}}|(\alpha(u_{\text{ms}}^i)-\alpha(u^i))\kappa_{F,s}\nabla_F u^i\cdot\nabla_F(w^i-u_{\text{ms}}^i)|\,dx
\end{aligned}\tag{A.2}$$

Following [64], we employ generalized Holder's Inequality and the definition of $\alpha(\cdot)$ to obtain

$$\begin{aligned}
&\int_{\Omega_M}|(\alpha(u_{\text{ms}}^i)-\alpha(u^i))\kappa^i\nabla u^i\cdot\nabla(w^i-u_{\text{ms}}^i)|\,dx\\
\leq&\|\alpha(u_{\text{ms}}^i)-\alpha(u^i)\|_{L^4(\Omega_M)}\|(\kappa^i)^{1/2}\nabla u^i\|_{L^4(\Omega_M)}\|(\kappa^i)^{1/2}\nabla(w^i-u_{\text{ms}}^i)\|_{L^2(\Omega_M)}\\
=&\frac{c}{\mu}\|u_{\text{ms}}^i-u^i\|_{L^4(\Omega_M)}\|(\kappa^i)^{1/2}\nabla u^i\|_{L^4(\Omega_M)}\|(\kappa^i)^{1/2}\nabla(w^i-u_{\text{ms}}^i)\|_{L^2(\Omega_M)}
\end{aligned}\tag{A.3}$$

Further, with Ladyzhenskaya's Inequality, there exists some constant $C_1>0$ dependent only on Ω such that

$$\|u_{\text{ms}}^i-u^i\|_{L^4(\Omega_M)}\leq C_1\|u_{\text{ms}}^i-u^i\|_{L^2(\Omega_M)}^{1/2}\|\nabla(u_{\text{ms}}^i-u^i)\|_{L^2(\Omega_M)}^{1/2}\tag{A.4}$$

There also exist some constant K_1,K_2 such that

$$\begin{aligned}
\|\nabla(u_{\text{ms}}^i-u^i)\|_{L^2(\Omega_M)}^2=\int_{\Omega_M}(\nabla(u_{\text{ms}}^i-u^i))^2\,dx\leq K_1\int_{\Omega_M}\frac{\kappa^i}{\mu}(\nabla(u_{\text{ms}}^i-u^i))^2\,dx,\\
\|u_{\text{ms}}^i-u^i\|_{L^2(\Omega_M)}^2=\int_{\Omega_M}(u_{\text{ms}}^i-u^i)^2\,dx\leq K_2\int_{\Omega_M}b^i(u_{\text{ms}}^i-u^i)^2\,dx,
\end{aligned}$$

where $K_1=\max_{i,x\in\Omega}\{\frac{\mu}{\kappa^i}\}$ and $K_2=\max_i\{\frac{1}{b^i}\}$.

For the fracture part, again from generalized Holder's Inequality, we have

$$\begin{aligned}&\int_{\Omega_{F,s}} |(\alpha(u^i_{\text{ms}}) - \alpha(u^i))\kappa_{F,s}\nabla_F u^i \cdot \nabla_F(w^i - u^i_{\text{ms}})|\, dx \\ &\leq C_2 \|u^i_{\text{ms}} - u^i\|_{L^2(\Omega_{F,s})} \|(\kappa_{F,s})^{1/2}\nabla(w^i - u_{\text{ms}})\|_{L^2(\Omega_{F,s})},\end{aligned} \tag{A.5}$$

for $C_2 = \frac{c}{\mu}\max_{i,s}\|(\kappa_{F,s})^{1/2}\nabla u^i\|_{L^\infty}$.

To sum up, we have for any $\zeta > 0$,

$$\int_\Omega |(\alpha(u^i_{\text{ms}}) - \alpha(u^i))\kappa^i\nabla u^i \cdot \nabla(w^i - u^i_{\text{ms}})|\, dx \leq\ C_3(\frac{1}{2\zeta}\|u^i_{\text{ms}} - u^i\|_b + \frac{\zeta}{2}|u^i_{\text{ms}} - u^i|_a) \cdot |w^i - u^i_{\text{ms}}|_a \tag{A.6}$$

for some constant C_3. Plug back to (A.1), and notice that $|\cdot|_a \leq \|\cdot\|_{a_Q}$ we can use Young's Inequality to derive

$$\begin{aligned}&\frac{1}{2}\frac{d}{dt}\|w - u_{\text{ms}}\|_b^2 + \alpha^-\|w - u_{\text{ms}}\|_{a_Q}^2 \\ &\leq \frac{1}{2\eta}\|\frac{\partial(w-u)}{\partial t}\|_b^2 + \frac{\eta}{2}\|w - u_{\text{ms}}\|_b^2 + \frac{\alpha^+}{2\xi}\|w-u\|_{a_Q}^2 + \frac{\alpha^+\xi}{2}\|w - u_{\text{ms}}\|_{a_Q}^2 \\ &\quad + \frac{C_3}{2\epsilon\zeta}\sum_{1\leq i<N} b^i(w^i - u^i, w^i - u^i) + \frac{C_3}{2\epsilon\zeta}\|w - u_{\text{ms}}\|_b^2 + \frac{C_3\zeta}{2\epsilon}\|w-u\|_{a_Q}^2 \\ &\quad + \frac{C_3\zeta}{2\epsilon}\|w - u_{\text{ms}}\|_{a_Q}^2 + \frac{C_3\epsilon\zeta}{4}\|w - u_{\text{ms}}\|_{a_Q}^2.\end{aligned}$$

Rearrange the inequality we obtain that

$$\begin{aligned}&\frac{1}{2}\frac{d}{dt}\|w - u_{\text{ms}}\|_b^2 - (\frac{C_3}{2\epsilon\zeta} + \frac{\eta}{2})\|w - u_{\text{ms}}\|_b^2 + (\alpha^- - \frac{\alpha^+\xi}{2} - \frac{C_3\zeta}{2\epsilon} - \frac{C_3\epsilon\zeta}{4})\|w - u_{\text{ms}}\|_{a_Q}^2 \\ &\leq\ \frac{1}{2\eta}\|\frac{\partial(w-u)}{\partial t}\|_b^2 + \frac{C_3}{2\epsilon\zeta}\sum_{1\leq i<N} b^i(w^i - u^i, w^i - u^i) + (\frac{\alpha^+}{2\xi} + \frac{C_3\zeta}{2\epsilon})\|w-u\|_{a_Q}^2\end{aligned} \tag{A.7}$$

We carefully choose $\epsilon = 1$, $\zeta = \frac{2\alpha^-}{3C_3} =$, $\xi = \frac{\alpha^-}{2\alpha^+}$, $\eta = 1$ and let

$$K = \frac{C_3}{\epsilon\zeta} + \eta = \frac{3C_3^2}{2\alpha^-} + 1,$$

then the previous equation can be simplified as

$$\begin{aligned}
&\frac{1}{2}\frac{d}{dt}\|w-u_{\text{ms}}\|_b^2-\frac{1}{2}K\|w-u_{\text{ms}}\|_b^2+(\frac{\alpha^-}{4})\|w-u_{\text{ms}}\|_{a_Q}^2\\
\leq\ &\frac{1}{2}\|\frac{\partial(w-u)}{\partial t}\|_b^2+\frac{3C_3^2}{4\alpha^-}\sum_{1\leq i<N}b^i(w^i-u^i,w^i-u^i)+(\frac{(\alpha^+)^2}{\alpha^-}+\frac{\alpha^-}{2})\|w-u\|_{a_Q}^2
\end{aligned}\tag{A.8}$$

To get rid of term $\|u_{\text{ms}}-w\|_b^2$, we multiply a $e^{-Kt}\leq 1$ to the above inequality and integrate over t from 0 to T for both sides, then we have

$$\begin{aligned}
&\frac{e^{-KT}}{2}\|w(T,\cdot)-u_{\text{ms}}(T,\cdot)\|_b^2+\frac{\alpha^-\cdot e^{-KT}}{4}\int_0^T\|w-u_{\text{ms}}\|_{a_Q}^2\,dt\\
\leq\ &\frac{1}{2}\int_0^T\|\frac{\partial(w-u)}{\partial t}\|_b^2\,dt+(\frac{(\alpha^+)^2}{\alpha^-}+\frac{\alpha^-}{2})\int_0^T\|w-u\|_{a_Q}^2\,dt\\
&+\frac{3C_3^2}{4\alpha^-}\int_0^T\sum_{1\leq i<N}b^i(w^i-u^i,w^i-u^i)\,dt+\frac{1}{2}\|w(0,\cdot)-u_{\text{ms}}(0,\cdot)\|_b^2.
\end{aligned}\tag{A.9}$$

We further define initial value $u_{\text{ms}}(0,\cdot)\in V_{\text{ms}}$, s.t.

$$b(u_{\text{ms}}(0,\cdot),v)=b(u(0,\cdot),v)\quad\forall v\in V_{\text{ms}}.$$

Thus,

$$\|w(0,\cdot)-u_{\text{ms}}(0,\cdot)\|_b\leq\|w(0,\cdot)-u(0,\cdot)\|_b.\tag{A.10}$$

Making use of the Poincare Inequality, we also have for some constant $K_3>0$

$$\sum_{1\leq i<N}b^i(w^i-u^i,w^i-u^i)\leq K_3\|w-u\|_{a_Q}^2.\tag{A.11}$$

Combining (A.9), (A.10) and (A.11), we conclude that there exist a constant $C_4>0$, such that

$$\begin{aligned}
&\|w(T,\cdot)-u_{\text{ms}}(T,\cdot)\|_b^2+\int_0^T\|w-u_{\text{ms}}\|_{a_Q}^2\,dt\\
\leq\ &C_4(\int_0^T\|\frac{\partial(w-u)}{\partial t}\|_b^2\,dt+\int_0^T\|w-u\|_{a_Q}^2\,dt+\|w(0,\cdot)-u(0,\cdot)\|_b^2).
\end{aligned}\tag{A.12}$$

With (A.12), we can start derive the inequality for Theorem 3.3.1,

$$\begin{aligned}&\|u(T,\cdot)-u_{\text{ms}}(T,\cdot)\|_b^2+\int_0^T\|u-u_{\text{ms}}\|_{a_Q}^2\,dt\\ \leq&\|w(T,\cdot)-u(T,\cdot)\|_b^2+\|w(T,\cdot)-u_{\text{ms}}(T,\cdot)\|_b^2+\int_0^T\|w-u\|_{a_Q}^2\,dt+\int_0^T\|w-u_{\text{ms}}\|_{a_Q}^2\,dt.\end{aligned}\tag{A.13}$$

For the first term on the right hand side of Inequality (A.13), we have

$$\|w(T,\cdot)-u(T,\cdot)\|_b^2\leq\ 2\int_0^T\|\frac{\partial(w-u)}{\partial t}\|_b^2\,dt+2\|w^i(0,\cdot)-u^i(0,\cdot)\|_b^2.$$

Combining the last estimate with (A.13) and (A.12), we conclude that for any $w\in V_{\text{ms}}$, the inequality holds for a constant $C>0$, such that

$$\begin{aligned}&\|u(T,\cdot)-u_{\text{ms}}(T,\cdot)\|_b^2+\int_0^T\|u-u_{\text{ms}}\|_{a_Q}^2\,dt\\ \leq C(&\int_0^T\|\frac{\partial(w-u)}{\partial t}\|_b^2\,dt+\int_0^T\|w-u\|_{a_Q}^2\,dt+\|w(0,\cdot)-u(0,\cdot)\|_b^2).\end{aligned}\tag{A.14}$$

This completes our proof.

□

A.1.2 Proof of Theorem 3.3.2

Proof. Since $u_{\text{snap}}\in V_{\text{snap}}$, we can write

$$u_{\text{snap}}(t,x)=\sum_j\sum_k c_k^{(j)}(t)\chi^{\omega_j}(x)\psi_k^{\omega_j}(x),\tag{A.15}$$

and we define the local component of u_{snap} by

$$u_{\text{snap}}^{(j)}(t,x)=\sum_k c_k^{(j)}(t)\psi_k^{\omega_j}(x).\tag{A.16}$$

We define $w \in V_{\text{ms}}$ as the projection of u_{snap} onto V_{ms} by

$$w = \sum_j \sum_{k=1}^{L_{\omega_j}} c_k^{(j)}(t)\psi_{k,ms}^{\omega_j}(x) = \sum_j \sum_{k=1}^{L_{\omega_j}} c_k^{(j)}(t)\chi^{\omega_j}(x)\psi_k^{\omega_j}(x). \tag{A.17}$$

From the definitions (A.15) and (A.17) ,we have

$$u_{\text{snap}} - w = \sum_j \sum_{k>L_{\omega_j}} c_k^{(j)}(t)\chi^{\omega_j}(x)\psi_k^{\omega_j}(x), \tag{A.18}$$

The desired result follows from the estimates in Lemma A.1.1, Lemma A.1.3 and Lemma A.1.4.

□

Lemma A.1.1. *Let $u_{snap} \in V_{snap}$ be defined in* (3.21) *and $w \in V_{ms}$ be defined in* (A.17)*. Then there exists a constant $C > 0$ such that*

$$\left\| \frac{\partial(u_{snap} - w)}{\partial t} \right\|_b^2 \leq \frac{C}{\Lambda} \left\| \frac{\partial u}{\partial t} \right\|_{a_Q}^2. \tag{A.19}$$

Proof.

$$\frac{\partial(u_{\text{snap}} - w)}{\partial t} = \sum_j \sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\chi^{\omega_j}(x)\psi_k^{\omega_j}(x)$$

Thus, for some constant $D_1 > 0$, we have

$$\|\frac{\partial(u_{\text{snap}} - w)}{\partial t}\|_b^2 \leq \ D_1 \sum_j \| \sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x)\|_b^2, \tag{A.20}$$

and the right-hand side can be estimated as

$$
\begin{aligned}
&\| \sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x)\|_b^2 \\
=& \sum_i \int_{\Omega_M} b^i(\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{i,\omega_j}(x))^2 \, dx + \sum_i \sum_s \int_{\Omega_{F,s}} b_{F,s}(\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{i,\omega_j}(x))^2 \, dx \\
\leq& D_2[\sum_{1\leq i<N} (\int_{\Omega_M} \frac{\kappa^i}{\mu}(\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{i,\omega_j}(x))^2 \, dx + \sum_s \int_{\Omega_{F,s}} \frac{\kappa_{F,s}}{\mu}(\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{i,\omega_j}(x))^2 \, dx) \\
&+ (\int_{\Omega_M} (\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{N,\omega_j}(x))^2 \, dx + \sum_s \int_{\Omega_{F,s}} (\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{N,\omega_j}(x))^2 \, dx)] \\
=& D_2[\sum_{1\leq i<N} (\int_{\omega_{j,M}} \frac{\kappa^i}{\mu}(\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{i,\omega_j}(x))^2 \, dx + \sum_s \int_{\omega_{j,F,s}} \frac{\kappa_{F,s}}{\mu}(\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{i,\omega_j}(x))^2 \, dx) \\
&+ (\int_{\omega_M} (\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{N,\omega_j}(x))^2 \, dx + \sum_s \int_{\omega_{F,s}} (\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{N,\omega_j}(x))^2 \, dx)] \\
=& D_2 s^{(j)}(\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x), \sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x))
\end{aligned}
$$

for some constant $D_2 > 0$.

By spectral problem (2.9) and the orthogonality of eigenfunctions $\{\psi_k^{\omega_j}\}_k$, we have

$$
\begin{aligned}
& s^{(j)}(\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x), \sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x)) \\
\leq& \frac{1}{\lambda_{L_{\omega_j}+1}^{\omega_j}} a_Q^{(j)}(\sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x), \sum_{k>L_{\omega_j}} (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x)) \\
\leq& \frac{1}{\lambda_{L_{\omega_j}+1}^{\omega_j}} a_Q^{(j)}(\sum_k (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x), \sum_k (\frac{d}{dt}c_k^{(j)}(t))\psi_k^{\omega_j}(x)) \\
=& \frac{1}{\lambda_{L_{\omega_j}+1}^{\omega_j}} a_Q^{(j)}(\frac{\partial u_{\text{snap}}^{(j)}}{\partial t}, \frac{\partial u_{\text{snap}}^{(j)}}{\partial t}).
\end{aligned}
\tag{A.21}
$$

Substituting this equation back to (A.20), we obtain

$$
\| \frac{\partial (u_{\text{snap}} - w)}{\partial t} \|_b^2 \leq D_1 D_2 \sum_j \frac{1}{\lambda_{L_{\omega_j}+1}^{\omega_j}} a_Q^{(j)}(\frac{\partial u_{\text{snap}}^{(j)}}{\partial t}, \frac{\partial u_{\text{snap}}^{(j)}}{\partial t}). \tag{A.22}
$$

Since u_{snap} is the projection of u in each ω_j by definition (A.16), so we have

$$a_Q^j(u^{(j)}, v) = a_Q^j(u_{\text{snap}}^{(j)}, v) \qquad \forall v \in V_{\text{snap}}^{(j)}.$$

More specifically, let $v = u_{\text{snap}}^{(j)}$ we have

$$\begin{aligned} a_Q^j(u_{\text{snap}}^{(j)}, u_{\text{snap}}^{(j)}) &= a_Q^j(u^{(j)}, u_{\text{snap}}^{(j)}), \\ \|u_{\text{snap}}^{(j)}\|_{a_Q}^2 &\leq \|u_{\text{snap}}^{(j)}\|_{a_Q} \|u^{(j)}\|_{a_Q}. \end{aligned}$$

Therefore,

$$a_Q^j(u_{\text{snap}}^{(j)}, u_{\text{snap}}^{(j)}) \leq a_Q^j(u^{(j)}, u^{(j)}).$$

Similarly,

$$a_Q^j(\frac{\partial u_{\text{snap}}^{(j)}}{\partial t}, \frac{\partial u_{\text{snap}}^{(j)}}{\partial t}) \leq a_Q^j(\frac{\partial u^{(j)}}{\partial t}, \frac{\partial u^{(j)}}{\partial t}). \tag{A.23}$$

Thus, from (A.22), we have

$$\|\frac{\partial(u_{\text{snap}} - w)}{\partial t}\|_b^2 \leq D_1 D_2 \sum_j \frac{1}{\lambda_{L_{\omega_j}+1}^{\omega_j}} a_Q^j(\frac{\partial u^{(j)}}{\partial t}, \frac{\partial u^{(j)}}{\partial t}) \leq \frac{D_1 D_2}{\min_j\{\lambda_{L_{\omega_j}+1}^{\omega_j}\}} \|\frac{\partial u}{\partial t}\|_{a_Q}^2. \tag{A.24}$$

This completes the proof. □

Lemma A.1.2. *For coupled multiscale basis function, if u satisfies the following*

$$\sum_{1\leq i<N} \int_{\omega_{j,M}} \kappa^i \nabla u^i \nabla v^i \, dx + \sum_{1\leq i<N} \sum_s \frac{\kappa_{F,s}}{\mu} \int_{\omega_{j,F,s}} \kappa_{F,s} \nabla_F u^i \nabla_F v^i \, dx + q(u, v) = \int_\omega fv \, dx, \; \forall v \in V_{snap}^{(j)}, \tag{A.25}$$

there exists some constant C, such that

$$\begin{aligned}
&\sum_{1\le i<N}\int_{\omega_{j,M}}\kappa^i(\chi^{\omega_j})^2(\nabla u^i)^2\,dx+\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\kappa_{f,s}(\chi^{\omega_j})^2(\nabla_F u^i)^2\,dx+q(\chi^{\omega_j}u,\chi^{\omega_j}u)\\
\le C\{&\sum_{1\le i<N}[\int_{\omega_j}(f^i)^2\frac{(\chi^{\omega_j})^2}{|\nabla\chi^{\omega_j}|^2\kappa^i}\,dx+\int_{\omega_{j,M}}\kappa^i(u^i\nabla\chi^{\omega_j})^2\,dx+\sum_s\int_{\omega_{j,F,s}}\kappa_{F,s}(u^i\nabla_F\chi^{\omega_j})^2\,dx]\\
&+\int_{\omega_j}(f^N)^2\frac{(\chi^{\omega_j})^2}{|\nabla\chi^{\omega_j}|^2}\,dx\}.
\end{aligned}\tag{A.26}$$

Proof. Let $v=(\chi^{\omega_j})^2u$ and obtain

$$\begin{aligned}
&\sum_{1\le i<N}\int_{\omega_{j,M}}\kappa^i\nabla u^i\nabla((\chi^{\omega_j})^2u^i)\,dx+\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\frac{\kappa_{F,s}}{\mu}\nabla_F u^i\nabla_F((\chi^{\omega_j})^2u^i)\,dx+q(u,(\chi^{\omega_j})^2u)\\
=&\int_\omega f(\chi^{\omega_j})^2u\,dx.
\end{aligned}$$

This can be further rewrite as

$$\begin{aligned}
&\sum_{1\le i<N}\int_{\omega_{j,M}}\kappa^i(\chi^{\omega_j})^2(\nabla u^i)^2\,dx+\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\kappa_{F,s}(\chi^{\omega_j})^2(\nabla_F u^i)^2\,dx+q(\chi^{\omega_j}u,\chi^{\omega_j}u)\\
=&\sum_{1\le i<N}\int_\omega f^i\frac{(\chi^{\omega_j})^2}{\nabla\chi^{\omega_j}\sqrt{\kappa^i}}\sqrt{\kappa^i}u^i\nabla\chi^{\omega_j}\,dx+\int_\omega f^N\frac{(\chi^{\omega_j})^2}{\nabla\chi^{\omega_j}}u^N\nabla\chi^{\omega_j}\,dx\\
&-2\sum_{1\le i<N}\int_{\omega_{j,M}}\kappa^i\nabla u^i\nabla\chi^{\omega_j}u^i\chi^{\omega_j}\,dx-2\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\kappa_{F,s}\nabla_F u^i\nabla_F\chi^{\omega_j}u^i\chi^{\omega_j}\,dx\\
\le&\frac{\epsilon}{2}\sum_{1\le i<N}\int_{\omega_j}(f^i)^2\frac{(\chi^{\omega_j})^4}{|\nabla\chi^{\omega_j}|^2\kappa^i}\,dx+\frac{1}{2\epsilon}\sum_{1\le i<N}\int_{\omega_{j,M}}\kappa^i(u^i\nabla\chi^{\omega_j})^2\,dx\\
&+\frac{\epsilon}{2}\int_{\omega_j}(f^N)^2\frac{(\chi^{\omega_j})^4}{|\nabla\chi^{\omega_j}|^2}\,dx+\frac{1}{2\epsilon}\int_{\omega_{j,M}}(u^N\nabla)^2\,dx\\
&+\sum_{1\le i<N}\sum_s\frac{1}{2\epsilon}\int_{\omega_{j,F,s}}\kappa_{F,s}(u^i\nabla_F)^2\,dx+\sum_s\frac{1}{2\epsilon}\int_{\omega_{j,F,s}}(u^N\nabla_F)^2\,dx\\
&+\epsilon\sum_{1\le i<N}\int_{\omega_{j,M}}\kappa^i(\chi^{\omega_j}\nabla u^i)^2\,dx+\frac{1}{\epsilon}\sum_{1\le i<N}\int_{\omega_{j,M}}\kappa^i(u^i\nabla\chi^{\omega_j})^2\,dx\\
&+\epsilon\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\kappa_{F,s}(\chi^{\omega_j}\nabla_F u^i)^2\,dx+\frac{1}{\epsilon}\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\kappa_{F,s}(u^i\nabla_F\chi^{\omega_j})^2\,dx.
\end{aligned}$$

Let $\epsilon = 1/2$ and rearrange the inequality. Then, for some constant $C > 0$, we obtain the conclusion of (A.26).

□

Lemma A.1.3. *Let $u_{snap} \in V_{snap}$ be defined in* (3.21) *and $w \in V_{ms}$ be defined in* (A.17)*. Then there exists a constant $C > 0$ such that*

$$\int_0^T \|w - u_{snap}\|_{a_Q}^2 \, dt \leq \frac{C}{\Lambda} \int_0^T \|u\|_{a_Q}^2. \tag{A.27}$$

Proof. By (A.18), we have

$$\|w - u_{\text{snap}}\|_{a_Q}^2 = \| \sum_j \sum_{k>L_{\omega_j}} c_k^{(j)}(t) \chi^{\omega_j}(x) \psi_k^{\omega_j}(x) \|_{a_Q}^2 \leq N_v \sum_j \|\chi^{\omega_j}(x) \sum_{k>L_{\omega_j}} c_k^{(j)}(t) \psi_k^{\omega_j}(x)\|_{a_Q}^2. \tag{A.28}$$

Let

$$e^{(j)} = \sum_{k>L_{\omega_j}} c_k^{(j)}(t) \psi_k^{\omega_j}(x),$$

then

$$\begin{aligned}
&\|\chi^{\omega_j}(x) e^{(j)}\|_{a_Q}^2 \\
&= \sum_{1\leq i<N} \int_{\omega_{j,M}} \frac{\kappa^i}{\mu} (\chi^{\omega_j})^2 [\nabla e^{(j),i}]^2 \, dx + \sum_{1\leq i<N} \int_{\omega_{j,M}} \frac{\kappa^i}{\mu} (\nabla \chi^{\omega_j})^2 [e^{(j),i}]^2 \, dx \\
&\quad + \sum_{1\leq i<N} \sum_s \int_{\omega_{j,F,s}} \frac{\kappa_{F,s}}{\mu} [\nabla_F (\chi^{\omega_j})]^2 [e^{(j),i}]^2 \, dx + \sum_{1\leq i<N} \sum_s \int_{\omega_{j,F,s}} \frac{\kappa_{F,s}}{\mu} (\chi^{\omega_j})^2 [e^{(j),i}]^2 \, dx \\
&\quad + q(\chi^{\omega_j}(x) e^{(j)}, \chi^{\omega_j}(x) e^{(j)}),
\end{aligned}$$

where

$$\begin{aligned}
&\sum_{1\le i<N}\int_{\omega_{j,M}}\frac{\kappa^i}{\mu}(\nabla\chi^{\omega_j})^2[e^{(j),i}]^2\,dx+\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\frac{\kappa_{F,s}}{\mu}[\nabla_F(\chi^{\omega_j})]^2[e^{(j),i}]^2\,dx\\
&\le D_3\sum_{1\le i<N}\int_{\omega_{j,M}}\frac{\kappa^i}{\mu}[e^{(j),i}]^2\,dx+D_3\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\frac{\kappa_{F,s}}{\mu}[e^{(j),i}]^2\,dx\\
&\le D_3 s^{(j)}(e^{(j)},e^{(j)})
\end{aligned}$$

for some constant D_3. From Lemma A.1.2, there exists some constant D_4 such that

$$\begin{aligned}
&\sum_{1\le i<N}\int_{\omega_{j,M}}\frac{\kappa^i}{\mu}(\chi^{\omega_j})^2[\nabla e^{(j),i}]^2\,dx\\
&+\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\frac{\kappa_{F,s}}{\mu}(\chi^{\omega_j})^2[\nabla_F e^{(j),i}]^2\,dx+q(\chi^{\omega_j}(x)e^{(j)},\chi^{\omega_j}(x)e^{(j)})\\
&\le D_4[\sum_{1\le i<N}\int_{\omega_{j,M}}\frac{\kappa^i}{\mu}|\nabla\chi^{\omega_j}|^2(e^{(j)})^2\,dx+\sum_{1\le i<N}\sum_s\int_{\omega_{j,F,s}}\frac{\kappa_{F,s}}{\mu}|\nabla_F\chi^{\omega_j}|^2(e^{(j)})^2\,dx]\\
&\le D_3D_4\ s^{(j)}(e^{(j)},e^{(j)}).
\end{aligned}$$

By bilinearity of $a^{(j)}$ and $s^{(j)}$ as well as the orthogonality of $\{\psi_k^{\omega_j}\}_k$,we finally have

$$\begin{aligned}
&\|w-u_{\text{snap}}\|_{a_Q}^2\le N_v\sum_j\|\chi^{\omega_j}(x)e^{(j)}\|_{a_Q}^2\le D_5\sum_j s^{(j)}(e^{(j)},e^{(j)})\\
&\le D_5\sum_j\frac{1}{\lambda^{\omega_j}_{L_{\omega_j}+1}}a_Q^{(j)}(e^{(j)},e^{(j)})\le\frac{D_5}{\Lambda}a_Q(u_{\text{snap}},u_{\text{snap}})=\frac{D_5}{\Lambda}\|u_{\text{snap}}\|_{a_Q}^2,
\end{aligned}\tag{A.29}$$

for a properly selected constant D_5.

□

Lemma A.1.4. *Let $u_{snap}\in V_{snap}$ be defined in* (3.21) *and $w\in V_{ms}$ be defined in* (A.17)*. Then there exists a constant $C>0$ such that*

$$\|w(0,\cdot)-u_{snap}(0,\cdot)\|_b^2\le\frac{C}{\Lambda}\|u(0,\cdot)\|_{a_Q}^2.\tag{A.30}$$

Proof. Using a similar idea as in Lemma A.1.3, we let

$$e_0^{(j)} = \sum_{k>L_{\omega_j}} c_k^{(j)}(0)\psi_k^{\omega_j}(x).$$

Then we have

$$\begin{aligned}
&\|u_{\text{snap}}(0,\cdot) - w(0,\cdot)\|_b^2 = \|\sum_j \chi^{\omega_j}(x) \sum_{k>L_{\omega_j}} c_k^{(j)}(0)\psi_k^{\omega_j}(x)\|_b^2 = \|\sum_j \chi^{\omega_j}(x) e_0^{(j)}\|_b^2 \\
\leq\ & D_1 \sum_j \|e_0^{(j)}\|_b^2 \leq D_1 D_2 \sum_j s^{(j)}(e_0^{(j)}, e_0^{(j)}) \leq D_1 D_2 \frac{1}{\Lambda} \sum_j a_Q^{(j)}(e_0^{(j)}, e_0^{(j)}) \\
\leq\ & D_1 D_2 \frac{1}{\Lambda} \sum_j a_Q^{(j)}(u_{\text{snap}}^{(j)}(0,x), u_{\text{snap}}^{(j)}(0,x)) = D_1 D_2 \frac{1}{\Lambda} \sum_j a_Q^{(j)}(u^{(j)}(0,x), u^{(j)}(0,x)) \\
=\ & D_1 D_2 \frac{1}{\Lambda} \|u^{(j)}(0,\cdot)\|_{a_Q}^2.
\end{aligned} \tag{A.31}$$

This completes the proof.

□