MODEL REDUCTION AND DEEP LEARNING APPROACHES IN RESERVOIR SIMULATION

A Dissertation

by

JINGYAN ZHANG

Submitted to the Graduate and Professional School 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,	Yuhe Wang
Committee Members,	Raytcho Lazarov
	Jianxin Zhou
Head of Department,	Sarah Witherspoon

December 2021

Major Subject: Mathematics

ABSTRACT

Numerical simulation of problems involving media with multiple spatial scales has important applications in many engineering and scientific areas, including material science, chemistry and unconventional reservoir simulation. While performing high-fidelity simulations is a primitive method in obtaining accurate approximation results, explicitness for complex models naturally gives rise to the need of large system of equations, due to the multiple scales and high contrast properties intrinsic to the reservoir. This will result in large degrees of freedom and will require enormous computational costs.

From the perspective of simulation, performing optimal reservoir management has always been challenging and extensive research efforts have been devoted to developing numerical methods with both accuracy and efficiency. One category of typical approaches include Heterogeneous Multiscale Methods, Variational Multiscale Methods and Multiscale Finite Element Methods. In such methods, effective properties are computed on a coarse-grid scale, which is much coarser than the fine-grid scale while multiscale basis functions are constructed to capture local oscillatory effects and to recover fine-scale information as needed. However, with more complex high-contrast heterogeneous media, such methods are insufficient in representing medium properties. In this dissertation, we discuss and analyze a novel multiscale model reduction approach for dual-continuum model, which serves as a powerful tool in subsurface formation applications from reservoir simulation.

Another category of approaches over the decades falls into surrogate modeling and physics-based model reductions to mitigate the difficulties induced by discretization of the nonlinear partial differential equations. Such techniques, such as POD-based methods, have been applied successfully in multi-phase flow problems and can efficiently maintain accuracy while establishing models with reduced complexity. However, such methods has no guarantee on solution stability and may lead to unphysical solutions, especially in the case of multi-phase flow simulation. Consequently,

many ad-hoc remedies have been implemented in recent years, including techniques based on deep learning, which has been proved with capability in approximating a wide variety of functions. In this dissertation, we also investigate methodologies of performing numerical simulations in combination with deep learning approaches for predicting nonlinear multi-phase dynamics in reservoir simulation.

ACKNOWLEDGMENTS

Firstly, I want to express my uttermost gratitude to my advisor, Dr. Yalchin Efendiev, for his continuous guidance and support throughout my graduate studies at Texas A&M University. Dr. Efendiev has always been supportive and provided me with precious opportunities to attend workshops, conferences and professional internship, which allow me to exchange ideas with researchers from different areas to broaden my horizons and to build up my own professional network. He has always been patient and inspiring to my research and career development. He is also caring when I'm faced with any difficulties. I sincerely appreciate all his contributions of time and ideas in helping me finish this dissertation.

Secondly, I am very grateful to my co-advisor, Dr. Yuhe Wang, for his professional suggestions and help on my research. Dr. Wang was very generous for supporting me to visit China University of Petroleum for a semester. During the visit, he also provided me with great opportunities to attend conferences and communicate with people having different areas of expertise.

Besides, I would also like to thank Dr. Raytcho Lazarov and Dr. Jianxin Zhou for taking their precious time serving as my committee. Their insightful comments and encouragement are very beneficial to me in the completion of this dissertation. I am also very grateful to Dr. Bicheng Yan and Dr. Siu Wun Cheung for patiently explaining research ideas and providing useful advice whenever I encountered difficulties. Moreover, sincere appreciation goes to Dr. Yanfang Yang, Dr. Yating Wang and Dr. Min Wang for sharing their precious experience and suggestions. Their selfless help has made my graduate life more smooth.

I appreciate all the help from the Department of Mathematics of Texas A&M University. It would be difficult for me to pursue my doctorate degree without the support of the department. The Department of Mathematics gives me many opportunities to attend different seminars, conferences and workshops, which helped me to explore future development of my research. The opportunities to engage in outreach programs also enriched my PhD life with warmth and joy. Special thanks also go to all my dear friends for their support and company throughout the ups and downs during

my life at Texas A&M University.

Last but not least, I want to express my profound gratitude to my parents for their support and love. Without their understanding and encouragement, I would not have been able to focus on my study.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professors Yalchin Efendiev of the Department of Mathematics and Professor Yuhe Wang of the Department of Petroleum Engineering.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University.

NOMENCLATURE

GMsFEM	Generalized Multiscale Finite Element Method
Ω	Spatial domain
$\mathcal{T}^h$	Fine-scale partition
$\mathcal{T}^H$	Coarse-scale partition
h	The fine mesh size
H	The coarse mesh size
K_j	A coarse-grid element
Ω_i	A fine grid cell
$\mathcal{E}^H$	Coarse-grid edge
γ_{ij}	edge of fine cells Ω_i and Ω_j
$K_{i,m}$	Coarse oversampled region
ϕ_l	An auxillary basis function
ψ_l	A multiscale basis function
κ	Permeability
p	Pressure
s	Saturation

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Literature

Reservoir management is a complex process to be performed given the multidisciplinary nature of all of the steps involved in combining proper reservoir modeling, data processing and decision support directives. From a simulation stand-point, performing optimal reservoir management has always been challenging due to the complexities and uncertainties intrinsic to reservoir dynamics [1]. Accurate simulation results can be obtained by exploring the full spectrum of uncertainties and optimal control settings using high fidelity models. In the case of large fractures, Discrete Fracture Model (DFM) and Embedded Fracture Model (EFM) are used to explicitly define fracture networks with accuracy [2, 3, 4]. However, this leads to solving expensive large-scale system of equations, which can be unmanageable given the computational infrastructure at one's disposal. In many cases, it is a critical obstacle to the fast changing decision-making process related to real-time data optimization and assimilation [5, 6].

To overcome this difficulty, research efforts had been devoted to constructing numerical solvers on a coarse grid, which is typically much coarser than the fine grid which captures all the heterogeneities in the medium properties. Typical approaches involve computing upscaled effective properties in each local coarse-grid block or representative volume [7, 8]. Such approaches are known to be insufficient when more than one important modes exist in the same coarse block or representative volume. In these cases, more efficient upscaling methods, typically the multicontinuum models, are employed [9, 10, 11, 12, 13, 14]. In such approaches, several effective properties are formulated in each coarse block and interaction terms are defined to characterize the transfer between different continua. Another class of methods is the multiscale methods, including Heterogeneous Multiscale Methods (HMM) [15, 16, 17], Variational Multiscale Methods (VMS) [18, 19, 20, 21] and Multiscale Finite Element Method (MsFEM) [22, 23]. Similar to upscaling approaches, multiscale scale is to construct numerical solvers on the coarse grid, which is typically

much coarser than the fine grid which captures all the heterogeneities in the medium properties. Instead of computing the effective medium properties, multiscale basis functions which are responsible for capturing the local oscillatory effects of the solution are constructed and coarse-scale macroscopic equations are formulated. The solution of the coarse-scale system can then be used to recover fine-scale information with the multiscale basis functions.

However, for more complex high-contrast heterogeneous media, each local coarse region contains several high-conductivity regions and multiple multiscale basis functions are required to represent the local solution space. The aforementioned multiscale methods typically use one basis function per coarse region, which is insufficient and may give rise to large error. To this end, it is crucial to systemically enrich the multiscale space with suitable fine-scale information for low-dimensional solution representation. One such approach is the Generalized Multiscale Finite Element Method (GMsFEM) [24, 25, 26, 27], which involves the construction appropriate snapshots space which consists of fine-scale data for solution representation by local snapshot problems and the construction of multiscale basis functions by performing feature extraction through local spectral decompositions to the snapshot space. Since multiscale basis functions are identified for multiscale solution representation in high-contrast heterogeneous media, multiple basis functions from the spectral problem are required to attain a small error. By introducing adaptivity [28, 27], one can add multiscale basis functions in selected regions. The connection of GMsFEM to multicontinuum models is discussed in [29], and GMsFEM are successfully applied to multicontinuum models that originate from fracture models and contain nonlinearities [30, 31, 32, 33].

More recently, a combination of GMsFEM and localization has been discussed in [34] as the approach of Constraint Energy Minimizing GMsFEM (CEM-GMsFEM). The method uses oversampling computational regions for the construction of multiscale basis functions. The first step is to find the auxiliary multiscale basis functions by GMsFEM. The second step is to construct multiscale basis functions by minimizing energy functionals subject to certain constraints, the purpose of which is to localize the multiscale basis functions. The method has been applied to various discretization and model problems [35, 36, 31, 37], and it has been theoretically and numerically

verified that the multiscale solutions spanned by the multiscale basis functions in CEM-GMsFEM have both spectral convergence and mesh dependent convergence.

On the other hand, to mitigate the computational effort induced by the discretization of the underlying coupled nonlinear partial differential equations, numerous techniques have been adapted over the decades. Surrogate modeling and physics-based model reduction, e.g., POD-based methods, have been applied successfully in multi-phase flow simulation [38, 39, 40, 41, 42, 43]. In many cases, it is shown that reduced-order modeling techniques are efficient in maintaining high level accuracy while reducing the complexity of models. However, there is not a general theory or methodology that can guarantee stable solutions for all test cases, especially when testing inputs are very different from the one used to generate the reduced models. Many ad-hoc remedies have been implemented in recent years [44, 45, 41], and some new ideas relying on machine learning techniques have been devised to compute robust projection basis [42, 43] . In particular [43] uses Recurrent Neural Nets to alleviate the computational cost associated with the training step (offline step), but no physical relation of the POD basis and the degrees of freedom (e.g. solution values at selected locations) is provided. This can lead to unphysical solutions, especially in the case of multi-phase flow systems.

In recent years, deep learning has attracted lots of attention. The concept of deep learning has been applied to a wide range of applications and has gained remarkable success in many fields including image and speech recognition. The core of deep learning is about training different artificial neural networks, inspired by the biological neural system, to achieve different targets. A typical branch of artificial neural network is the Deep Neural Network (DNN), which normally consists of an input layer, an output layer and several hidden layers in between. Each layer contains several neurons and each neuron can process and transfer signals to other neurons through connections.

There have been a lot of studies in the literature on the ability of neural networks in approximating a wide class of functions. Many researchers have also provided theory on the expressivity of deep neural networks [46, 47, 48, 49, 50, 51]. The multi-layer structure utilize DNN the ability

to solve complex problems and approximate complicated functions. Some activation functions, used as nonlinear signal transformation, can determine whether a neuron is activated or not. This inspires many works in exploiting deep learning for solving nonlinear differential equations and model reductions. Examples can be seen in [52], where deep neural network is used to represent the trial function in the Ritz method. This utilize deep neural network the ability in solving Poisson problems and eigenvalue problems. Another work is given in [53]. The author builds a connection between residual networks (ResNet) and transport equation, which propose a continuous flow model for ResNet and shows an alternative perspective to understand deep neural networks.

Efforts have also been made in developing physics-constrained surrogate modeling for partial differential equations. In the development of physics-informed neural network (PINN), loss functions are regularized by physics governing equations through automatic differentiation so that the consistency between the neural networks and the physics of fluid flow in porous media is guaranteed [54]. While PINNs can predict fluid dynamics by physics with medium complexity, they still suffer from the curse of dimensionality and high computational costs induced by high nonlinearity.

Recently, image-based approaches have also been developed to predict flow dynamics in porous media. The typical approaches leverage convolutional neural networks (CNN), which have existed for decades [55]. CNNs have a wide range of successful applications in areas including video and image recognition [56], image segmentation [57] and natural language processing [58]. They are specialized to capture spatial and temporal patterns in images. With locally connected layers, they can perform more efficiently in training while maintaining the capability of extracting rich hidden features. This inspires works in approximating nonlinear relationships between geological maps and flow maps in fluid flow in porous media [59, 60, 61]. Examples can be seen in [62], where Bayesian deep convolutional encoder-decoder networks show good results as surrogate models for solving stochastic PDEs. In another work [63], a residual convolutional neural network is developed to predict subsurface flow dynamics in channelized permeability field.

1.2 Organization of this dissertation

In Chapter 2, we will develop and analyze a rigorous multiscale upscaling method for dual continuum model, which serves as a powerful tool in subsurface formation applications. Our proposed method is capable of identifying different continua and capturing non-local transfer and effective properties in the computational domain via constructing localized multiscale basis functions. The construction of the basis functions consists of solving local problems defined on oversampling computational region, subject to the energy minimizing constraints that the mean values of the local solution are zero in all continua except for the one targeted. The basis functions constructed are shown to have good approximation properties. We provide rigorous mathematical analysis on that the method has a coarse mesh dependent convergence. We also present some numerical examples to illustrate the performance of the proposed method.

In Chapter 3, we will study the application of model reduction techniques, combining with multi-layer neural network approaches for simulations of nonlinear multi-phase flow, taking into account both the observed data and physical modeling concepts. The output of the multi-layer network is regarded as the solution at current time step given initial time and input parameters, which come from various sources including permeability fields and source terms. We select basis functions of the global reduced order model such that the degrees of freedom can represent the solutions at observation points. In such manner, learning of basis functions, which can also be accomplished by neural networks, can be avoided. We will investigate how the neural network architecture, including the number of layers and neurons, affects the approximation of the forward map in the governing nonlinear equations of multi-phase flow.

In Chapter 4, we will further investigate efficient image-based deep learning techniques for approximating the dynamics of multi-phase flow. A hybrid workflow using both deep neural network and numerical simulator is designed and tested. By breaking down the coupled multi-phase flow system into pressure and saturation equations, relying on the capability of image-based operations, we are able to construct efficient surrogate models using image-based neural networks to predict pressure, which can be later fed into explicit fine-scale saturation solvers for the simulating of sat-

uration. Physics-based loss functions are introduced into the training process to further guarantee spatial continuity and temporal stability in the proposed scheme. Some numerical examples are included to illustrate the efficiency and accuracy of the proposed workflow.

2. ANALYSIS OF NON-LOCAL MULTICONTINUUM UPSCALING FOR DUAL CONTINUUM MODEL

In this chapter, we provide rigorous mathematical analysis on the nonlocal multicontinuum (NLMC) upscaling method for a dual continuum model. The auxiliary basis functions are simply defined in each coarse block for each continuum, which represents fractures and matrix. To be more precise, in each oversampling domain, the auxiliary basis functions are constant in a continuum, and each has mean value one for the chosen continuum and zero otherwise. Out of the oversampling domain, the value of the basis functions are zero. The degrees of freedom is the same as the number of the continua, which is the minimal number needed to represent the heterogeneous property of the reservoir. To obtain the multiscale basis functions, we solve local minimization problems in oversampling computational domain. We show that the minimizer has a good decay property. With a proper number of oversampling layers, the basis functions derived can well capture the fine-grid information. Moreover, we show that the method has a convergence dependent on coarse mesh size. We also present some numerical examples to depict the performance of the method.

The chapter is organized as follows. In Section 2.1 we present the dual continuum model. The proposed method is introduced in Section 2.2 and analyzed in Section 2.3. In Section 2.4 some numerical experiments are demonstrated to confirm the the theory.

2.1 Dual continuum model

We consider the following dual continuum model [10, 64, 13]

$$\begin{aligned} c_1\frac{\partial p_1}{\partial t} - \text{div}(\kappa_1\nabla p_1) + \sigma(p_1 - p_2) &= f_1, \\ c_2\frac{\partial p_2}{\partial t} - \text{div}(\kappa_2\nabla p_2) - \sigma(p_1 - p_2) &= f_2, \end{aligned} \tag{2.1}$$

in a computational domain $\Omega \subset \mathbb{R}^2$. Here, for $i = 1, 2$, c_i is the compressibility, p_i is the pressure, κ_i is the permeability, and f_i is the source function for the i-th continuum. In addition, the continua

are coupled through the mass exchange, and σ is a parameter which accounts for the strength of mass transfer between the continua. One particular application of the dual continuum model (2.1) is to represent the global interactive effects of the unresolved fractures and the matrix. In this work, we consider high-contrast channelized media. We prescribe the initial condition $p_i(0,\cdot) = p_i^0$ in Ω and the boundary condition $p_i(t,\cdot) = 0$ on $\partial\Omega$ for $t > 0$.

Let $V = [H_0^1(\Omega)]^2$. Also, for a subdomain $D \subset \Omega$, we denote the restriction of V on D by $V(D)$. The weak formulation of (2.1) then reads: find $p = (p_1, p_2)$ such that $p(t,\cdot) \in V$ and

$$c\left(\frac{\partial p}{\partial t}, v\right) + a_Q(p, v) = (f, v), \tag{2.2}$$

for all $v = (v_1, v_2)$ with $v(t,\cdot) \in V$. Here, $(\cdot,\cdot)$ denotes the standard $L^2(\Omega)$ inner product. Moreover, the bilinear forms are defined as:

$$\begin{aligned}
c_i(p_i, v_i) &= \int_\Omega c_i(x) p_i v_i \,\mathrm{d}x,\\
c(p, v) &= \sum_i c_i(p_i, v_i),\\
a_i(p_i, v_i) &= \int_\Omega \kappa_i(x) \nabla p_i \cdot \nabla v_i \,\mathrm{d}x,\\
a(p, v) &= \sum_i a_i(p_i, v_i),\\
q(p, v) &= \sum_i \sum_{i'} \int_\Omega \sigma(p_i - p_{i'}) v_i \,\mathrm{d}x,\\
a_Q(p, v) &= a(p, v) + q(p, v).
\end{aligned} \tag{2.3}$$

2.2 Method description

In this section, we will describe our proposed method in detail. To start with, we introduce the concepts of coarse and fine meshes. We start with a plain partition of calculation domain Ω, $\mathcal{T}^H$. This partition is called a coarse mesh, which does not necessarily resolve any multiscale features. We denote one element in $\mathcal{T}^H$ as K and name it as a coarse element. Here, $H > 0$ is

the coarse mesh size. We denote the number of coarse elements and coarse grid nodes as N and N_c respectively. The collection of all coarse element edges is called $\mathcal{E}^H$. To sufficiently resolve the solution, we refine the coarse mesh $\mathcal{T}^H$ into a fine mesh $\mathcal{T}^h$, where $h > 0$ is called the fine mesh size. We remark that the fine grid system is only used in locally solving process, where all local problems are solved continuously. Therefore, we don't consider fine grid in our analysis hereinafter.

Next, we clarify more notions concerning every coarse element. Letting $K_j \in \mathcal{T}^H$ be the j-th coarse element, an oversampling domain $K_{j,m}$ is defined by expanding K_j with m layers of coarse elements in Ω. An illustration of the fine-coarse-oversampling mesh system is given in Figure 2.1. Moreover, similar to the partition of computational domain Ω, for $i = 1, 2$, we denote $K_j = \cup_{l=1}^{L_i^{(j)}} K_l^{(i,j)}$, where $L_i^{(j)}$ denotes the number of high conductive channels plus matrix within K_j for continua i.

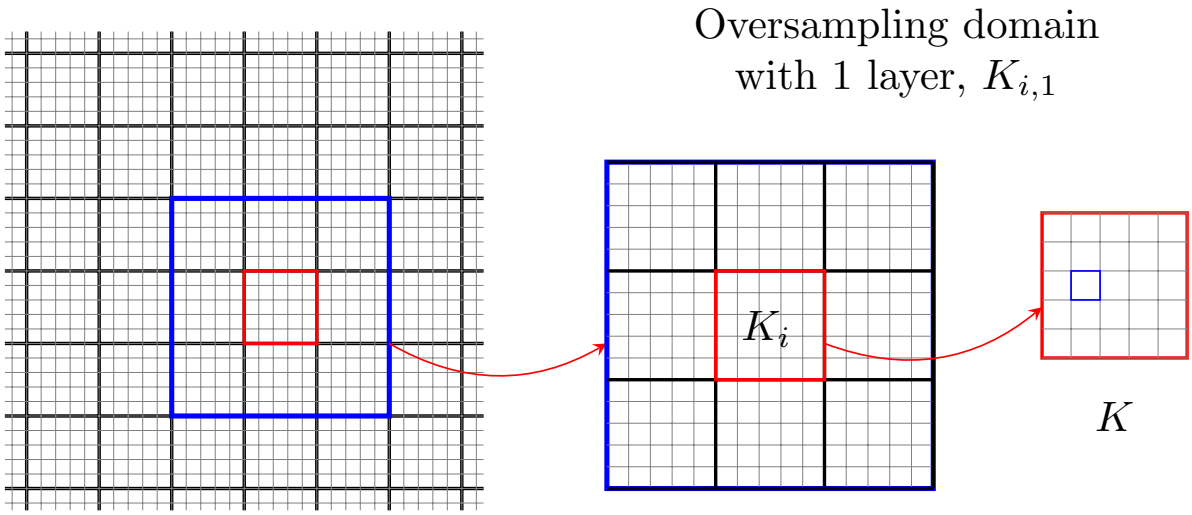


Figure 2.1: An illustration of fine mesh, coarse mesh and oversampling domain.

We now proceed to describing the proposed method step by step.

Step 1. Definition of auxiliary basis functions. Within $K_j (j = 1, \ldots, N)$, for each $l =$

$1, \ldots, L_i^{(j)}$, we directly define our auxiliary basis function $\phi_l^{(i,j)} \in P^0(\mathcal{T}^h) \cap L^2(K_j)$ as

$$\phi_l^{(i,j)} = \frac{1}{|K_l^{(i,j)}|} \mathcal{I}_{K_l^{(i,j)}}. \tag{2.4}$$

Here, $|K_l^{(i,j)}|$ denotes the area of $K_l^{(i,j)}$ and $\mathcal{I}_{K_l^{(i,j)}}$ is the characteristic function. The local auxiliary space for continua i is constructed as

$$V_{\text{aux}}^{(i,j)} = \text{span}\{\phi_l^{(i,j)} | 1 \leq l \leq L_i^{(j)}\}. \tag{2.5}$$

Furthermore, we define the local auxiliary space as

$$V_{\text{aux}}^{j} = V_{\text{aux}}^{(1,j)} \times V_{\text{aux}}^{(2,j)}. \tag{2.6}$$

The global auxiliary space is defined as the direct sum of all local auxiliary spaces as

$$V_{\text{aux}} = \oplus_{j=1}^{N} V_{\text{aux}}^{j}. \tag{2.7}$$

Step 2. Construction of multiscale basis functions. For the convenience of describing the method and the subsequent convergence analysis, for all $v = (v_1, v_2) \in [L^2(K_j)]^2$, we introduce a local projection operator $\pi_j : [L^2(K_j)]^2 \to V_{\text{aux}}$ as

$$\pi_j(v) = \left(\sum_{l=1}^{L_1^{(j)}} (v_1, \phi_l^{(1,j)}) \phi_l^{(1,j)}, \sum_{l=1}^{L_2^{(j)}} (v_2, \phi_l^{(2,j)}) \phi_l^{(2,j)} \right) \tag{2.8}$$

and a global projection operator $\pi : [L^2(\Omega)]^2 \to V_{\text{aux}}$ is defined as

$$\pi(v) = \sum_{j=1}^{N} \pi_j(v), \quad \forall v \in [L^2(\Omega)]^2. \tag{2.9}$$

With these tools, we can define a set of global multiscale basis functions $\psi_l^{(i,j)}$ by solving a

saddle point problem: for every coarse element K_j in Ω, find $\psi_l^{(i,j)} \in V$ and $T^{i,j,l}_{i',j',l'} \in \mathbb{R}$ such that

$$\begin{aligned} a_Q(\psi_l^{(i,j)}, w) + \sum_{i'=1}^{2} \sum_{K_{l'}^{(i',j')} \subset \Omega} T^{i,j,l}_{i',j',l'}(w \cdot \mathbf{e}_{i'}, \phi_{l'}^{(i',j')}) = 0, \quad \forall w \in V, \\ (\psi_l^{(i,j)} \cdot \mathbf{e}_{i'}, \phi_{l'}^{(i',j')}) = \delta_{i,i'}\delta_{l,l'}\delta_{j,j'}, \quad \forall K_{l'}^{(i',j')} \subset \Omega. \end{aligned} \tag{2.10}$$

The global multiscale finite element space is thus defined as

$$V_{\text{glo}} = \text{span}\{\psi_l^{(i,j)} | 1 \leq l \leq L_i^{(j)}, 1 \leq j \leq N, i = 1, 2\}. \tag{2.11}$$

Here, $\mathbf{e}_i$ is the canonical basis for $\mathbb{R}^2$. $\delta_{i,i'}$, $\delta_{l,l'}$ and $\delta_{j,j'}$ are the delta Dirac function. We remark that if we denote $\tilde{V}$ as the null space of the global projection operator π, for any $\psi_l^{(i,j)} \in V_{\text{glo}}$, we have

$$a_Q(\psi_l^{(i,j)}, v) = 0, \quad \forall v \in \tilde{V}. \tag{2.12}$$

This implies that with respect to the inner product of a_Q, $\tilde{V} \subset V_{\text{glo}}^{\perp}$. As a matter of fact, $\tilde{V} = V_{\text{glo}}^{\perp}$.

As our analysis suggests, the global basis functions exhibit exponential decay properties and have small values outside a sufficiently large oversampling region. The fact suggests that we can save computational costs without introducing huge error by truncating the domain. This inspires the definition the localized multiscale basis functions which are constructed in the similar way but on localized oversampled domains $K_{j,m}$ of every $K_j \subset \Omega$: find $\psi_{l,\text{ms}}^{(i,j)} \in V(K_{j,m})$ and $\overline{T}^{i,j,l}_{i',j',l'} \in \mathbb{R}$ such that for $i = 1, 2$, we have

$$\begin{aligned} a_Q(\psi_{l,\text{ms}}^{(i,j)}, w) + \sum_{i'=1}^{2} \sum_{K_{l'}^{(i',j')} \subset K_{j,m}} \overline{T}^{i,j,l}_{i',j',l'}(w \cdot \mathbf{e}_{i'}, \phi_{l'}^{(i',j')}) = 0, \quad \forall w \in V(K_{j,m}), \\ (\psi_{l,\text{ms}}^{(i,j)} \cdot \mathbf{e}_{i'}, \phi_{l'}^{(i',j')}) = \delta_{i,i'}\delta_{l,l'}\delta_{j,j'}, \quad \forall K_{l'}^{(i',j')} \subset K_{j,m}. \end{aligned} \tag{2.13}$$

We use the local multiscale basis functions to obtain the multiscale finite element space, which

will be used for deriving the multiscale solution, as

$$V_{\text{ms}} = \text{span}\{\psi_{l,\text{ms}}^{(i,j)} | 1 \leq l \leq L_i^{(j)}, 1 \leq j \leq N, i = 1, 2\}. \tag{2.14}$$

Step 3. Multiscale solution. The process of finding the multiscale solution can be described as follows. Find $p_{\text{ms}} = (p_{\text{ms},1}, p_{\text{ms},2})$ with $p_{\text{ms}}(t, \cdot) \in V_{\text{ms}}$ s.t. for all $v = (v_1, v_2)$ with $v(t, \cdot) \in V_{\text{ms}}$,

$$c\left(\frac{\partial p_{\text{ms}}}{\partial t}, v\right) + a_Q(p_{\text{ms}}, v) = (f, v). \tag{2.15}$$

2.3 Convergence analysis

In this section, we will analyze the proposed method. First, we define the following norms and semi-norms on V:

$$\begin{aligned}
\|p\|_c^2 &= c(p, p), \\
\|p\|_a^2 &= a(p, p), \\
|p|_q^2 &= q(p, p), \\
\|p\|_{a_Q}^2 &= a_Q(p, p), \\
\|p\|_{\mathbb{L}^2(\Omega;\kappa)}^2 &= \sum_i (\kappa_i^{\frac{1}{2}} p_i, \kappa_i^{\frac{1}{2}} p_i), \\
\|p\|_{\mathbb{L}^2(\Omega;\kappa^{-1})}^2 &= \sum_i (\kappa_i^{-\frac{1}{2}} p_i, \kappa_i^{-\frac{1}{2}} p_i).
\end{aligned} \tag{2.16}$$

For a subdomain $D = \bigcup_{j\in J} K_j$ as a union of coarse grid blocks, we also define the following local norms and semi-norms on V:

$$\begin{aligned}
\|p\|^2_{a(D)} &= \sum_{j\in J} a^{(j)}(p,p),\\
|p|^2_{q(D)} &= \sum_{j\in J} q^{(j)}(p,p),\\
\|p\|^2_{a_Q(D)} &= \sum_{j\in J} a^{(j)}_Q(p,p),\\
\|p\|^2_{\mathbb{L}^2(D;\kappa)} &= \sum_{j\in J} (\kappa_i^{\frac{1}{2}} p_i, \kappa_i^{\frac{1}{2}} p_i)_{L^2(K_j)},\\
\|p\|^2_{\mathbb{L}^2(D;\kappa^{-1})} &= \sum_{j\in J} (\kappa_i^{-\frac{1}{2}} p_i, \kappa_i^{-\frac{1}{2}} p_i)_{L^2(K_j)}.
\end{aligned} \tag{2.17}$$

We remark that

$$\begin{aligned}
\|p\|_{\mathbb{L}^2(D;\kappa)} &\leq \overline{\kappa}^{\frac{1}{2}} \|p\|_{[L^2(D)]^2},\\
\|p\|_{\mathbb{L}^2(D;\kappa^{-1})} &\leq \underline{\kappa}^{-\frac{1}{2}} \|p\|_{[L^2(D)]^2}.
\end{aligned} \tag{2.18}$$

In addition, we introduce some operators which will be used in our analysis, namely $R_{\text{glo}} : V \to V_{\text{glo}}$ given by: for any $u \in V$, the image $R_{\text{glo}}u \in V_{\text{glo}}$ is defined by

$$a_Q(R_{\text{glo}}u, v) = a_Q(u,v), \quad \forall v \in V_{glo}, \tag{2.19}$$

and similarly, $R_{\text{ms}} : V \to V_{\text{ms}}$ given by: for any $u \in V$, the image $R_{\text{ms}}u \in V_{\text{ms}}$ is defined by

$$a_Q(R_{\text{ms}}u, v) = a_Q(u,v), \quad \forall v \in V_{\text{ms}}. \tag{2.20}$$

We also define $\mathcal{C} : V \to V$ given by: for any $u \in V$, the image $\mathcal{C}u \in V$ is defined by

$$(\mathcal{C}u, v) = c(u,v), \quad \forall v \in V. \tag{2.21}$$

Moreover, the operator $\mathcal{A} : D(\mathcal{A}) \to [L^2(\Omega)]^2$ is defined on a subspace $D(\mathcal{A}) \subset V$ by: for any $u \in D(\mathcal{A})$, the image $\mathcal{A}u \in [L^2(\Omega)]^2$ is defined by

$$(\mathcal{A}u, v) = a_Q(u, v), \quad \forall v \in V. \tag{2.22}$$

The following lemma shows that the projection operator R_{glo} has a good approximation property with respect to the a_Q-norm and L^2-norm.

Lemma 1. *Let $u \in D(\mathcal{A})$, then we have $u - R_{glo}u \in \tilde{V}$ and*

$$\|u - R_{glo}u\|_{a_Q} \leq CH \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})}. \tag{2.23}$$

and

$$\|u - R_{glo}u\|_{[L^2(\Omega)]^2} \leq CH^2 \underline{\kappa}^{-\frac{1}{2}} \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})}. \tag{2.24}$$

Proof. By (2.19), we directly get $u - R_{\text{glo}}u \in V_{\text{glo}}$. This yields

$$a_Q(u - R_{\text{glo}}u, R_{\text{glo}}u) = 0. \tag{2.25}$$

Thus, we have

$$\begin{aligned} a_Q(u - R_{\text{glo}}u, u - R_{\text{glo}}u) &= a_Q(u - R_{\text{glo}}u, u) - a_Q(u - R_{\text{glo}}u, R_{\text{glo}}u) \\ &= a_Q(u - R_{\text{glo}}u, u) \\ &= a_Q(u, u - R_{\text{glo}}u) \\ &= (\mathcal{A}u, u - R_{\text{glo}}u) \\ &\leq \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})} \|(u - R_{\text{glo}}u)\|_{\mathbb{L}^2(\Omega;\kappa)}. \end{aligned} \tag{2.26}$$

Since $u - R_{\text{glo}}u \in \tilde{V}$, we have $\pi_j(u - R_{\text{glo}}u) = 0$ for all $l = 1, 2, \ldots, L_j$ and $j = 1, 2, \ldots, N$.

Moreover, the Poincaré inequality gives

$$\int_{K_l^{(i,j)}} [(u - R_{\text{glo}}u) \cdot \mathbf{e}_i]^2 \leq CH^2 \int_{K_l^{(i,j)}} |\nabla[(u - R_{\text{glo}}u) \cdot \mathbf{e}_i]|^2. \tag{2.27}$$

This yields that

$$\begin{aligned} \|(u - R_{\text{glo}}u)\|^2_{\mathbb{L}^2(\Omega;\kappa)} &= \sum_i \left\| \kappa_i^{\frac{1}{2}} (u - R_{\text{glo}}u) \cdot \mathbf{e}_i \right\|^2_{L^2(\Omega)} \\ &= \sum_i \sum_{K_l^{(i,j)} \subset \Omega} \left\| \kappa_i^{\frac{1}{2}} (u - R_{\text{glo}}u) \cdot \mathbf{e}_i \right\|^2_{L^2(K_l^{(i,j)})} \\ &\leq CH^2 \|u - R_{\text{glo}}u\|^2_{a_Q}. \end{aligned} \tag{2.28}$$

Thus, we have

$$\|u - R_{\text{glo}}u\|^2_{a_Q} \leq CH \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})} \|u - R_{\text{glo}}u\|_{a_Q}, \tag{2.29}$$

which gives the estimate in the (2.23).

The proof of (2.24) follows a duality argument. Define $w \in V$ such that

$$a_Q(w, v) = (u - R_{\text{glo}}u, v) \quad \forall v \in V. \tag{2.30}$$

Then we have

$$\|u - R_{\text{glo}}u\|^2_{[L^2(\Omega)]^2} = (u - R_{\text{glo}}u, u - R_{\text{glo}}u) = a_Q(w, u - R_{\text{glo}}u). \tag{2.31}$$

Taking $v = R_{\text{glo}}w \in V_{\text{glo}}$ in (2.19), we obtain

$$a_Q(u - R_{\text{glo}}u, R_{\text{glo}}w) = 0. \tag{2.32}$$

Since $w \in D(\mathcal{A})$ and $\mathcal{A}w = u - R_{\text{glo}}u$, we have

$$\begin{aligned}
\|u - R_{\text{glo}}u\|^2_{[L^2(\Omega)]^2} &= a_Q(w - R_{\text{glo}}w, u - R_{\text{glo}}u) \\
&\leq \|w - R_{\text{glo}}w\|_{a_Q} \|u - R_{\text{glo}}u\|_{a_Q} \\
&\leq \left(CH \|\mathcal{A}w\|_{\mathbb{L}^2(\Omega;\kappa^{-1})}\right)\left(CH \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})}\right) \\
&\leq \left(CH\underline{\kappa}^{-\frac{1}{2}} \|\mathcal{A}w\|_{[L^2(\Omega)]^2}\right)\left(CH \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})}\right) \\
&\leq CH^2\underline{\kappa}^{-\frac{1}{2}} \|u - R_{\text{glo}}u\|_{[L^2(\Omega)]^2} \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})}.
\end{aligned} \tag{2.33}$$

□

Next, we show that the global basis functions are localizable. For the purpose of this, for each coarse block K, we define a bubble function B such that $B(x) > 0, \forall x \in \text{int}(K)$ and $B(x) = 0, \forall x \in \partial K$. We will take $B = \prod_{x_k} \chi^H_k$, where χ^H_k is coarse scale partition of unity on K. Based on the bubble function, we define a constant as follows.

$$C_{\text{equiv}} = \sup_{K_j \in \mathcal{T}^H, v \in V_{\text{aux}}} \frac{\|v\|^2_{[L^2(K_j)]^2}}{\left\|B^{\frac{1}{2}}v\right\|^2_{[L^2(K_j)]^2}}. \tag{2.34}$$

Lemma 2. *For all $v_{aux} \in V_{aux}$, there exists a function $v \in V$ such that*

$$\pi(v) = v_{aux}, \quad \|v\|^2_{a_Q} \leq D \|v_{aux}\|^2_{\mathbb{L}^2(\Omega;\kappa)}, \quad \text{supp}(v) \subset \text{supp}(v_{aux}), \tag{2.35}$$

where $D = \frac{C_{\mathcal{T}}^2}{H^2} + 2\max_i \left\|\frac{\sigma_i}{\kappa_i}\right\|_{L^\infty(\Omega)}$ and $C_{\mathcal{T}}$ is the maximum of vertices over all coarse elements.

Proof. Without loss of generality we assume $v_{\text{aux}} \in V^j_{\text{aux}}$ with $\|v_{\text{aux}}\|_{[L^2(K_j)]^2} = 1$. We consider the following saddle point problem: find $v \in V_0(K_j)$ and $T^{i'}_{l'} \in \mathbb{R}$ such that

$$\begin{aligned}
a_Q(v, w) + \sum_{i'=1}^{2} \sum_{K^{(i',j)}_{l'} \subset K_j} T^{i'}_{l'}(w \cdot \mathbf{e}_{i'}, \phi^{(i',j)}_{l'}) = 0, \quad \forall w \in V_0(K_j), \\
((v - v_{\text{aux}}) \cdot \mathbf{e}_{i'}, \phi^{(i',j)}_{l'}) = 0, \quad \forall K^{(i',j)}_{l'} \subset K_j.
\end{aligned} \tag{2.36}$$

The well-posedness of the above saddle point problem is equivalent to the existence of $\widetilde{v} \in V_0(K_j)$ such that

$$(\widetilde{v}, v_{\text{aux}}) \geq C_1 \left\| v_{\text{aux}} \right\|^2_{\mathbb{L}^2(K_j;\kappa)}, \quad \left\| \widetilde{v} \right\|_{a_Q(K_j)} \leq C_2 \left\| v_{\text{aux}} \right\|_{\mathbb{L}^2(K_j;\kappa)}, \tag{2.37}$$

where C_1, C_2 are constants to be determined. Taking $\widetilde{v} = Bv_{\text{aux}}$, we have

$$(\widetilde{v}, v_{\text{aux}}) = \left\| B^{\frac{1}{2}} v_{\text{aux}} \right\|^2_{[L^2(K_j)]^2} \geq C^{-1}_{\text{equiv}} \left\| v_{\text{aux}} \right\|^2_{[L^2(K_j)]^2}. \tag{2.38}$$

On the other hand, on every $K_l^{(i,j)}$, we have

$$\nabla(\widetilde{v} \cdot \mathbf{e}_i) = B\nabla(\widetilde{v} \cdot \mathbf{e}_i) + \nabla B(\widetilde{v} \cdot \mathbf{e}_i). \tag{2.39}$$

By definition of V_{aux}, $\left\| B\nabla(\widetilde{v} \cdot \mathbf{e}_i) \right\|_{L^2(K_j)} = 0$. At the same time, $|B| \leq 1$, $|\nabla B| \leq C_{\mathcal{T}} H^{-1}$. Thus,

$$\left\| \kappa_i^{\frac{1}{2}} \nabla(\widetilde{v} \cdot \mathbf{e}_i) \right\|^2_{L^2(K_j)} \leq \frac{C_{\mathcal{T}}^2}{H^2} \left\| \kappa_i (\widetilde{v} \cdot \mathbf{e}_i) \right\|^2_{L^2(K_j)}. \tag{2.40}$$

This yields

$$\left\| \widetilde{v} \right\|^2_{a(K_j)} \leq \frac{C_{\mathcal{T}}^2}{H^2} \sum_i \left\| \kappa_i^{\frac{1}{2}} (v_{\text{aux}} \cdot \mathbf{e}_i) \right\|^2_{L^2(K_j)} \tag{2.41}$$

and

$$|\widetilde{v}|^2_{q(K_j)} \leq 2 \max_i \left\| \frac{\sigma_i}{\kappa_i} \right\|_{L^\infty(K_j)} \sum_i \left\| \kappa_i^{\frac{1}{2}} (v_{\text{aux}} \cdot \mathbf{e}_i) \right\|^2_{L^2(K_j)}. \tag{2.42}$$

Thus, we have

$$\left\| \widetilde{v} \right\|^2_{a_Q(K_j)} \leq \left(\frac{C_{\mathcal{T}}^2}{H^2} + 2 \max_i \left\| \frac{\sigma_i}{\kappa_i} \right\|_{L^\infty(\Omega)} \right) \left\| v_{\text{aux}} \right\|^2_{\mathbb{L}^2(K_j;\kappa)}, \tag{2.43}$$

This guarantees the existence and uniqueness of $v \in V_0(K_j)$ and $T_{l'}^{i'} \in \mathbb{R}$ satisfying (2.36), in which v satisfies our desired properties. □

Here, we make a remark that we can assume $D \geq 1$ without loss of generality.

In order to estimate the difference between the global basis functions and localized basis functions, we need the notion of a cutoff function with respect to the oversampling regions. For each

coarse grid K_j and $M > m$, we define $\chi_j^{M,m} \in \operatorname{span}\{\chi_k^H\}$ such that $0 \leq \chi_j^{M,m} \leq 1$ and $\chi_j^{M,m} = 1$ on the inner region $K_{j,m}$ and $\chi_j^{M,m} = 0$ outside the region $K_{j,M}$.

The following lemma shows that our multiscale basis functions have a decay property. In particular, the global basis functions are small outside an oversampling region specified in the lemma, which is important in localizing the multiscale basis functions. By truncating the support of the multiscale basis function $\psi_{l,\text{ms}}^{(i,j)}$ into the oversampled region $K_{j,m}$, it introduces an discrepancy to the global basis function $\psi_l^{(i,j)}$ which scales with an exponent $-m$, where m is the number of oversampled layers. It is formally stated and proved by Caccioppoli-like arguments in the following Lemma.

Lemma 3. *Given $\phi_l^{(i,j)} \in V_{aux}^j$ and an oversampling region $K_{j,m}$ with number of layers $m \geq 2$. Let $\psi_{l,ms}^{(i,j)}$ be a localized multiscale basis function defined on $K_{j,m}$ given by* (2.13)*, and $\psi_l^{(i,j)}$ be the corresponding global basis function given by* (2.10)*. Then we have*

$$\left\| \psi_l^{(i,j)} - \psi_{l,ms}^{(i,j)} \right\|_{a_Q}^2 \leq E \left\| \phi_l^{(i,j)} \right\|_{\mathbb{L}^2(K_j;\kappa)}^2, \tag{2.44}$$

where $E = 8D(2+D)(1+CH^2)\left(1 + (C^{\frac{1}{2}}D^{\frac{1}{2}}H + CH^2)^{-1}\right)^{1-m}$.

Proof. By Lemma 2, there exists $v \in V$ such that

$$\pi(v) = \phi_l^{(i,j)}, \quad \|v\|_{a_Q}^2 \leq D \left\| \phi_l^{(i,j)} \right\|_{\mathbb{L}^2(\Omega;\kappa)}^2, \quad \operatorname{supp}(v) \subset K_j. \tag{2.45}$$

We take $\eta = \psi_l^{(i,j)} - v \in V$ and $\zeta = v - \psi_{l,\text{ms}}^{(i,j)} \in V(K_{j,m})$. Then $\pi(\eta) = \pi(\zeta) = 0$ and hence $\eta, \zeta \in \widetilde{V}$. We first see that for $K_{j'} \subset K_{j,m-1}$,

$$\pi_{j'}(\chi_j^{m,m-1}\eta) = \pi_{j'}(\eta) = 0, \tag{2.46}$$

since $\chi_j^{m,m-1} = 1$ on $K_{j,m-1}$ and $\eta \in \widetilde{V}$. On the other hand, for $K_{j'} \subset \Omega \setminus K_{j,m}$,

$$\pi_{j'}(\chi_j^{m,m-1}\eta) = \pi_{j'}(0) = 0, \tag{2.47}$$

since $\chi_j^{m,m-1} = 0$ on $\Omega \setminus K_{j,m}$. Therefore, we have supp $\left(\pi(\chi_j^{m,m-1}\eta)\right) \subset K_{j,m} \setminus K_{j,m-1}$. Again, by Lemma 2, there exists $\beta \in V$ such that

$$\pi(\beta) = \pi(\chi_j^{m,m-1}\eta), \quad \|\beta\|_{a_Q}^2 \leq D \left\|\pi(\chi_j^{m,m-1}\eta)\right\|^2_{\mathbb{L}^2(K_{j,m}\setminus K_{j,m-1};\kappa)}, \quad \text{supp}(\beta) \subset K_{j,m} \setminus K_{j,m-1}. \tag{2.48}$$

Take $\tau = \beta - \chi_j^{m,m-1}\eta \in V(K_{j,m})$. Again, $\pi(\tau) = 0$ and hence $\tau \in \widetilde{V}$. Now, by the variational problems (2.10) and (2.13), we have

$$\begin{aligned} a_Q(\psi_l^{(i,j)}, w) + \sum_{i'=1}^{2} \sum_{K_{l'}^{(i',j')} \subset \Omega} T^{i,j,l}_{i',j',l'}(w \cdot \mathbf{e}_{i'}, \phi_{l'}^{(i',j')}) = 0, \quad \forall w \in V, \\ a_Q(\psi_{l,\text{ms}}^{(i,j)}, w) + \sum_{i'=1}^{2} \sum_{K_{l'}^{(i',j')} \subset K_{j,m}} \overline{T}^{i,j,l}_{i',j',l'}(w \cdot \mathbf{e}_{i'}, \phi_{l'}^{(i',j')}) = 0, \quad \forall w \in V(K_{j,m}) \end{aligned} \tag{2.49}$$

Taking $w = \tau - \zeta \in V(K_{j,m})$ and using the fact that $\tau - \zeta \in \widetilde{V}$, we have

$$a_Q(\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}, \tau - \zeta) = 0, \tag{2.50}$$

which implies

$$\begin{aligned} \left\|\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}\right\|_{a_Q}^2 &= a_Q(\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}, \psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}) \\ &= a_Q(\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}, \eta + \zeta) \\ &= a_Q(\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}, \eta + \tau) \\ &\leq \left\|\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}\right\|_{a_Q} \|\eta + \tau\|_{a_Q}. \end{aligned} \tag{2.51}$$

By (2.48), we have

$$\begin{aligned}\left\|\psi_l^{(i,j)}-\psi_{l,\mathrm{ms}}^{(i,j)}\right\|_{a_Q}^2 &\leq \|\eta+\tau\|_{a_Q}^2\\ &= \left\|(1-\chi_j^{m,m-1})\eta+\beta\right\|_{a_Q}^2\\ &\leq 2\left(\left\|(1-\chi_j^{m,m-1})\eta\right\|_{a_Q}^2+\|\beta\|_{a_Q}^2\right)\\ &\leq 2\left(\left\|(1-\chi_j^{m,m-1})\eta\right\|_{a_Q}^2+D\left\|\chi_j^{m,m-1}\eta\right\|_{\mathbb{L}^2(K_{j,m}\setminus K_{j,m-1};\kappa)}^2\right).\end{aligned} \tag{2.52}$$

For the first term on the right hand side of (2.52), since

$$\nabla\left((1-\chi_j^{m,m-1})(\eta\cdot\mathbf{e}_i)\right)=(1-\chi_j^{m,m-1})\nabla(\eta\cdot\mathbf{e}_i)-(\eta\cdot\mathbf{e}_i)\nabla\chi_j^{m,m-1}, \tag{2.53}$$

and $|1-\chi_j^{m,m-1}|\leq 1$, we have

$$\left\|(1-\chi_j^{m,m-1})\eta\right\|_a^2\leq 2\left(\|\eta\|_{a(\Omega\setminus K_{j,m-1})}^2+\|\eta\|_{\mathbb{L}^2(\Omega\setminus K_{j,m-1};\kappa)}^2\right). \tag{2.54}$$

On the other hand, we have

$$|(1-\chi_j^{m,m-1})\eta|_q^2\leq|\eta|_{q(\Omega\setminus K_{j,m-1})}^2. \tag{2.55}$$

Therefore, we arrive at

$$\left\|(1-\chi_j^{m,m-1})\eta\right\|_{a_Q}^2\leq 2\left(\|\eta\|_{a_Q(\Omega\setminus K_{j,m-1})}^2+\|\eta\|_{\mathbb{L}^2(\Omega\setminus K_{j,m-1};\kappa)}^2\right). \tag{2.56}$$

For the second term on the right hand side of (2.52), using the fact that $|\chi_j^{m,m-1}|\leq 1$, we have

$$\begin{aligned}\left\|\pi(\chi_j^{m,m-1}\eta)\right\|_{\mathbb{L}^2(K_{j,m}\setminus K_{j,m-1};\kappa)}^2 &\leq \left\|\chi_j^{m,m-1}\eta\right\|_{\mathbb{L}^2(K_{j,m}\setminus K_{j,m-1};\kappa)}^2\\ &\leq \|\eta\|_{\mathbb{L}^2(K_{j,m}\setminus K_{j,m-1};\kappa)}^2.\end{aligned} \tag{2.57}$$

To sum up, we have

$$\left\|\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}\right\|_{a_Q}^2 \leq 4\left\|\eta\right\|_{a_Q(\Omega\setminus K_{j,m-1})}^2 + (4+2D)\left\|\eta\right\|_{\mathbb{L}^2(\Omega\setminus K_{j,m-1};\kappa)}^2. \tag{2.58}$$

Since $\eta \in \widetilde{V}$, using Poincaré inequality, we obtain

$$\left\|\eta\right\|_{\mathbb{L}^2(\Omega\setminus K_{j,m-1};\kappa)}^2 \leq CH^2\left\|\eta\right\|_{a_Q(\Omega\setminus K_{j,m-1})}^2. \tag{2.59}$$

Combining all the estimates, we have

$$\left\|\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}\right\|_{a_Q}^2 \leq (4+2D)(1+CH^2)\left\|\eta\right\|_{a_Q(\Omega\setminus K_{j,m-1})}^2. \tag{2.60}$$

Next, we will prove a recursive estimate for $\left\|\eta\right\|_{a_Q(\Omega\setminus K_{j,m-1})}^2$. We take $\xi = 1-\chi_j^{m-1,m-2}$. Then $\xi = 1$ in $\Omega\setminus K_{j,m-1}$ and $0\leq\xi\leq 1$. Hence, using

$$\nabla(\xi^2(\eta\cdot\mathbf{e}_i)) = \xi^2\nabla(\eta\cdot\mathbf{e}_i) + 2\xi(\eta\cdot\mathbf{e}_i)\nabla\xi, \tag{2.61}$$

we have

$$|\xi\eta|_a^2 = a(\eta,\xi^2\eta) + \left\|\eta\right\|_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)}^2, \tag{2.62}$$

which results in

$$\left\|\eta\right\|_{a_Q(\Omega\setminus K_{j,m-1})}^2 \leq \left\|\xi\eta\right\|_{a_Q}^2 \leq a_Q(\eta,\xi^2\eta) + \left\|\eta\right\|_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)}^2. \tag{2.63}$$

We will estimate the first term on the right hand side of (2.63). Following the preceding argument, we see that $\text{supp}(\pi(\xi^2\eta)) \subset K_{j,m-1}\setminus K_{j,m-2}$. By Lemma 2, there exists $\gamma\in V$ such that

$$\pi(\gamma) = \pi(\xi^2\eta), \quad \left\|\gamma\right\|_{a_Q}^2 \leq D\left\|\pi(\xi^2\eta)\right\|_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)}^2, \quad \text{supp}(\gamma)\subset K_{j,m-1}\setminus K_{j,m-2}. \tag{2.64}$$

Take $\theta = \xi^2\eta - \gamma$. Again, $\pi(\theta) = 0$ and hence $\theta \in \widetilde{V}$. Therefore, we have

$$a_Q(\psi_l^{(i,j)}, \theta) = 0. \tag{2.65}$$

Additionally, $\mathrm{supp}(\theta) \subset \Omega \setminus K_{j,m-2}$. Recall that, in (2.45), we have $\mathrm{supp}(v) \subset K_j$. Hence θ and v have disjoint supports, and

$$a_Q(v, \theta) = 0. \tag{2.66}$$

Therefore, we obtain

$$a_Q(\eta, \theta) = a_Q(\psi_l^{(i,j)}, \theta) - a_Q(v, \theta) = 0. \tag{2.67}$$

Note that $\xi^2\eta = \theta + \gamma$. Using (2.64), we have

$$\begin{aligned} a_Q(\eta, \xi^2\eta) &= a_Q(\eta, \gamma) \\ &\leq \|\eta\|_{a_Q(K_{j,m-1}\setminus K_{j,m-2})} \|\gamma\|_{a_Q(K_{j,m-1}\setminus K_{j,m-2})} \\ &\leq D^{\frac{1}{2}} \|\eta\|_{a_Q(K_{j,m-1}\setminus K_{j,m-2})} \left\|\pi(\xi^2\eta)\right\|_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)}. \end{aligned} \tag{2.68}$$

Since $|\xi| \leq 1$, we have

$$\left\|\pi(\xi^2\eta)\right\|_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)} \leq \left\|\xi^2\eta\right\|_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)} \leq \|\eta\|_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)}. \tag{2.69}$$

Hence, the right hand side of (2.63) can be estimated by

$$\|\eta\|^2_{a_Q(\Omega\setminus K_{j,m-1})} \leq D^{\frac{1}{2}} \|\eta\|_{a_Q(K_{j,m-1}\setminus K_{j,m-2})} \|\eta\|_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)} + \|\eta\|^2_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)}. \tag{2.70}$$

Since $\pi(\eta) = 0$, using Poincaré inequality, we have

$$\|\eta\|^2_{\mathbb{L}^2(K_{j,m-1}\setminus K_{j,m-2};\kappa)} \cdot \leq CH^2 \|\eta\|^2_{a_Q(K_{j,m-1}\setminus K_{j,m-2})}, \tag{2.71}$$

which implies

$$\|\eta\|^2_{a_Q(\Omega\setminus K_{j,m-1})} \leq (C^{\frac{1}{2}} D^{\frac{1}{2}} H + CH^2) \|\eta\|^2_{a_Q(K_{j,m-1}\setminus K_{j,m-2})}. \tag{2.72}$$

Therefore,

$$\begin{aligned} \|\eta\|^2_{a_Q(\Omega\setminus K_{j,m-2})} &= \|\eta\|^2_{a_Q(\Omega\setminus K_{j,m-1})} + \|\eta\|^2_{a_Q(K_{j,m-1}\setminus K_{j,m-2})} \\ &\geq \left(1 + (C^{\frac{1}{2}} D^{\frac{1}{2}} H + CH^2)^{-1}\right) \|\eta\|^2_{a_Q(\Omega\setminus K_{j,m-1})}. \end{aligned} \tag{2.73}$$

Inductively, we have

$$\begin{aligned} \|\eta\|^2_{a_Q(\Omega\setminus K_{j,m-1})} &\leq \left(1 + (C^{\frac{1}{2}} D^{\frac{1}{2}} H + CH^2)^{-1}\right)^{1-m} \|\eta\|^2_{a_Q(\Omega\setminus K_j)} \\ &\leq \left(1 + (C^{\frac{1}{2}} D^{\frac{1}{2}} H + CH^2)^{-1}\right)^{1-m} \|\eta\|^2_{a_Q}. \end{aligned} \tag{2.74}$$

Finally, we estimate the term on the right hand side of (2.74). Recall from the first property of v in (2.45), we have $\pi(v) = \phi_l^{(i,j)}$, which implies

$$(v \cdot \mathbf{e}_{i'}, \phi_{l'}^{(i',j')}) = \delta_{i,i'}\delta_{l,l'}\delta_{j,j'}, \quad \forall K_{l'}^{(i',j')} \subset \Omega. \tag{2.75}$$

Taking $w = \eta$ in (2.10), we have

$$a_Q(\psi_l^{(i,j)}, \eta) = 0, \tag{2.76}$$

which implies

$$\|\psi_l^{(i,j)}\|_{a_Q} \leq \|v\|_{a_Q}. \tag{2.77}$$

Using a triangle inequality and the second property of v in (2.45), we have

$$\|\eta\|_{a_Q} = \left\|\psi_l^{(i,j)} - v\right\|_{a_Q} \leq 2\|v\|_{a_Q} \leq 2D^{\frac{1}{2}} \left\|\phi_l^{(i,j)}\right\|_{\mathbb{L}^2(K_j;\kappa)}. \tag{2.78}$$

Combining (2.60), (2.74) and (2.78), we obtain our desired result. □

The following lemma shows that, similar to the global projection operator R_{glo}, our localized multiscale finite element projection operator R_{ms} can also provide a good approximation with respect to the a_Q-norm and L^2-norm.

Lemma 4. *Let $u \in D(\mathcal{A})$. Let $m \geq 2$ be the number of coarse grid layers in the oversampling regions in* (2.13)*. If $m = O\left(\log\left(\frac{\overline{\kappa}}{H}\right)\right)$, we have*

$$\|u - R_{ms}u\|_{a_Q} \leq CH \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})}. \tag{2.79}$$

and

$$\|u - R_{ms}u\|_{[L^2(\Omega)]^2} \leq CH^2 \underline{\kappa}^{-\frac{1}{2}} \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})}. \tag{2.80}$$

Proof. We write

$$R_{\text{glo}}u = \sum_{i=1}^{2}\sum_{j=1}^{N}\sum_{l=1}^{L_i^{(j)}} \alpha_l^{(i,j)}\psi_l^{(i,j)} \in V_{\text{glo}} \tag{2.81}$$

and define

$$w = \sum_{i=1}^{2}\sum_{j=1}^{N}\sum_{l=1}^{L_i^{(j)}} \alpha_l^{(i,j)}\psi_{l,\text{ms}}^{(i,j)} \in V_{\text{ms}}. \tag{2.82}$$

By (2.20), we have

$$\|u - R_{\text{ms}}u\|_{a_Q} \leq \|u - w\|_{a_Q} \leq \|u - R_{\text{glo}}u\|_{a_Q} + \|R_{\text{glo}}u - w\|_{a_Q} \tag{2.83}$$

By Lemma 3, we have that

$$\begin{aligned}
\|R_{\text{glo}}u - w\|_{a_Q}^2 &= \left\| \sum_{i=1}^{2}\sum_{j=1}^{N}\sum_{l=1}^{L_i^{(j)}} \alpha_l^{(i,j)}(\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}) \right\|_{a_Q}^2 \\
&\leq C(m+1)^2 \sum_{j=1}^{N} \left\| \sum_{i=1}^{2}\sum_{l=1}^{L_i^{(j)}} \alpha_l^{(i,j)}(\psi_l^{(i,j)} - \psi_{l,\text{ms}}^{(i,j)}) \right\|_{a_Q}^2 \\
&\leq CE(m+1)^2 \sum_{j=1}^{N} \left\| \sum_{i=1}^{2}\sum_{l=1}^{L_i^{(j)}} \alpha_l^{(i,j)}\phi_l^{(i,j)} \right\|_{\mathbb{L}^2(K_j;\kappa)}^2 \\
&\leq CE(m+1)^2 \|R_{\text{glo}}u\|_{\mathbb{L}^2(\Omega;\kappa)}^2 .
\end{aligned} \tag{2.84}$$

Combining (2.83), (2.84) and Lemma 1, we have

$$\|u - R_{\text{ms}}u\|_{a_Q} \leq CH \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})} + CE^{\frac{1}{2}}(m+1) \|R_{\text{glo}}u\|_{\mathbb{L}^2(\Omega;\kappa)} . \tag{2.85}$$

Now, we estimate $\|R_{\text{glo}}u\|_{\mathbb{L}^2(\Omega;\kappa)}^2$. By Ponicaré inequality, we have

$$\|R_{\text{glo}}u\|_{\mathbb{L}^2(\Omega;\kappa)}^2 \leq \overline{\kappa} \|R_{\text{glo}}u\|_{[L^2(\Omega)]^2}^2 \leq C_p \overline{\kappa}\underline{\kappa}^{-1} \|R_{\text{glo}}u\|_{a_Q}^2 . \tag{2.86}$$

Taking $v = R_{\text{glo}}u$ in (2.19) and by Cauchy-Schwarz inequality, we obtain

$$\|R_{\text{glo}}u\|_{a_Q}^2 = a_Q(u, R_{\text{glo}}u) = (\mathcal{A}u, R_{\text{glo}}u) \leq \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})} \|R_{\text{glo}}u\|_{\mathbb{L}^2(\Omega;\kappa)} . \tag{2.87}$$

Combining (2.86) and (2.87), we obtain

$$\|R_{\text{glo}}u\|_{\mathbb{L}^2(\Omega;\kappa)} \leq C\overline{\kappa}\underline{\kappa}^{-1} \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})} . \tag{2.88}$$

This yields

$$\|u - R_{\text{ms}}u\|_{a_Q} \leq C(H + \overline{\kappa}\underline{\kappa}^{-1}E^{\frac{1}{2}}(m+1)) \|\mathcal{A}u\|_{\mathbb{L}^2(\Omega;\kappa^{-1})} \tag{2.89}$$

To obtain desired result, we will need

$$H^{-1}\overline{\kappa}\underline{\kappa}^{-1}E^{\frac{1}{2}}(m+1) = O(1). \tag{2.90}$$

Taking logarithm, we have

$$\log(H^{-1}) + \log(\overline{\kappa}) - \log(\underline{\kappa}) + \frac{1-m}{2}\log\left(1 + (C^{\frac{1}{2}}D^{\frac{1}{2}}H + CH^2)^{-1}\right) = O(1). \tag{2.91}$$

Therefore, if we take $m = O\left(\log\left(\frac{\overline{\kappa}}{H}\right)\right)$, we have (2.79). The proof of (2.80) follows a similar duality argument as in Lemma 1. □

Now we are ready to obtain our main theorem on estimating error between p and p_{ms}.

Theorem 5. *Suppose $f \in [L^2(\Omega)]^2$. Let $m \geq 2$ be the number of coarse grid layers of the over-sampling domain in* (2.13)*. Let p be the solution of* (2.2) *and p_{ms} of* (2.15)*. If $m = O\left(\log\left(\frac{\overline{\kappa}}{H}\right)\right)$, we have*

$$\|p(T,\cdot) - p_{ms}(T,\cdot)\|_c^2 + \int_0^T \|p - p_{ms}\|_{a_Q}^2 \,\mathrm{d}t \leq CH^2\underline{\kappa}^{-1}\left(\left\|p^0\right\|_{a_Q}^2 + \int_0^T \|f\|_{[L^2(\Omega)]^2}^2 \,\mathrm{d}t\right). \tag{2.92}$$

Proof. Taking $v = \frac{\partial p}{\partial t}$ in (2.2), we have

$$\left\|\frac{\partial p}{\partial t}\right\|_c^2 + \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|p\|_{a_Q}^2 = \left(f, \frac{\partial p}{\partial t}\right) \leq C\|f\|_{[L^2(\Omega)]^2}^2 + \frac{1}{2}\left\|\frac{\partial p}{\partial t}\right\|_c^2. \tag{2.93}$$

Integrating over $(0,T)$, we have

$$\frac{1}{2}\int_0^T \left\|\frac{\partial p}{\partial t}\right\|_c^2 \mathrm{d}t + \frac{1}{2}\|p(T,\cdot)\|_{a_Q}^2 \leq C\left(\left\|p^0\right\|_{a_Q}^2 + \int_0^T \|f\|_{[L^2(\Omega)]^2}^2 \,\mathrm{d}t\right) \tag{2.94}$$

Similarly, by taking $v = \frac{\partial p_{\text{ms}}}{\partial t}$ in (2.15) and integrating over $(0,T)$, we obtain

$$\frac{1}{2}\int_0^T \left\|\frac{\partial p_{\text{ms}}}{\partial t}\right\|_c^2 \mathrm{d}t + \frac{1}{2}\|p_{\text{ms}}(T,\cdot)\|_{a_Q}^2 \leq C\left(\left\|p^0\right\|_{a_Q}^2 + \int_0^T \|f\|_{[L^2(\Omega)]^2}^2 \,\mathrm{d}t\right) \tag{2.95}$$

At the same time, by (2.2), we can see that

$$\mathcal{A}p = f - \mathcal{C}\frac{\partial p}{\partial t}. \tag{2.96}$$

Thus, we have

$$\|\mathcal{A}p\|^2_{[L^2(\Omega)]^2} \leq C\left(\|f\|^2_{[L^2(\Omega)]^2} + \left\|\frac{\partial p}{\partial t}\right\|_c\right) \tag{2.97}$$

By the definition of p and p_{ms} in (2.2) and (2.15), respectively, we can get that $\forall v \in V_{\text{ms}}$ and $t \in (0, T)$, we have

$$c\left(\frac{\partial(p - p_{\text{ms}})}{\partial t}, v\right) + a_Q(p - p_{\text{ms}}, v) = 0. \tag{2.98}$$

Thus, we have

$$\begin{aligned}
&\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|p - p_{\text{ms}}\|_c^2 + \|p - p_{\text{ms}}\|_{a_Q}^2 \\
&= c\left(\frac{\partial(p - p_{\text{ms}})}{\partial t}, p - p_{\text{ms}}\right) + a_Q(p - p_{\text{ms}}, p - p_{\text{ms}}) \\
&= c\left(\frac{\partial(p - p_{\text{ms}})}{\partial t}, p - R_{\text{ms}}p\right) + a_Q(p - p_{\text{ms}}, p - R_{\text{ms}}p) \\
&\leq \left\|\frac{\partial(p - p_{\text{ms}})}{\partial t}\right\|_c \|p - R_{\text{ms}}p\|_c + \|p - p_{\text{ms}}\|_{a_Q} \|p - R_{\text{ms}}p\|_{a_Q} \\
&\leq \left(\left\|\frac{\partial p}{\partial t}\right\|_c + \left\|\frac{\partial p_{\text{ms}}}{\partial t}\right\|_c\right) \|p - R_{\text{ms}}p\|_c + \frac{1}{2}\|p - p_{\text{ms}}\|_{a_Q}^2 + \frac{1}{2}\|p - R_{\text{ms}}p\|_{a_Q}^2 .
\end{aligned} \tag{2.99}$$

Integrating over $(0,T)$ and using (2.96) by Lemma 4 with (2.18), we obtain

$$\begin{aligned}
&\frac{1}{2}\left\|p(T,\cdot)-p_{\mathrm{ms}}(T,\cdot)\right\|_c^2+\frac{1}{2}\int_0^T\left\|p-p_{\mathrm{ms}}\right\|_{a_Q}^2\,\mathrm{d}t\\
&\leq\int_0^T\left(\left\|\frac{\partial p}{\partial t}\right\|_c+\left\|\frac{\partial p_{\mathrm{ms}}}{\partial t}\right\|_c\right)\left\|p-R_{\mathrm{ms}}p\right\|_c\,\mathrm{d}t+\frac{1}{2}\int_0^T\left\|p-R_{\mathrm{ms}}p\right\|_{a_Q}^2\,\mathrm{d}t\\
&\leq\left(\int_0^T\left(\left\|\frac{\partial p}{\partial t}\right\|_c+\left\|\frac{\partial p_{\mathrm{ms}}}{\partial t}\right\|_c\right)^2\mathrm{d}t\right)^{\frac{1}{2}}\left(\int_0^T\left\|p-R_{\mathrm{ms}}p\right\|_c^2\,\mathrm{d}t\right)^{\frac{1}{2}}+\frac{1}{2}\int_0^T\left\|p-R_{\mathrm{ms}}p\right\|_{a_Q}^2\,\mathrm{d}t\\
&\leq\left(\int_0^T\left(\left\|\frac{\partial p}{\partial t}\right\|_c+\left\|\frac{\partial p_{\mathrm{ms}}}{\partial t}\right\|_c\right)^2\mathrm{d}t\right)^{\frac{1}{2}}\left(\int_0^T CH^4\underline{\kappa}^{-2}\left(\left\|f\right\|_{[L^2(\Omega)]^2}+\left\|\frac{\partial p}{\partial t}\right\|_c\right)^2\mathrm{d}t\right)^{\frac{1}{2}}\\
&\quad+\int_0^T CH^2\underline{\kappa}^{-1}\left(\left\|f\right\|_{[L^2(\Omega)]^2}+\left\|\frac{\partial p}{\partial t}\right\|_c\right)^2\mathrm{d}t\\
&\leq CH^2\underline{\kappa}^{-1}\int_0^T\left(\left\|\frac{\partial p}{\partial t}\right\|_c^2+\left\|\frac{\partial p}{\partial t}\right\|_c^2+\left\|f\right\|_{[L^2(\Omega)]^2}^2\right)\mathrm{d}t.
\end{aligned}\tag{2.100}$$

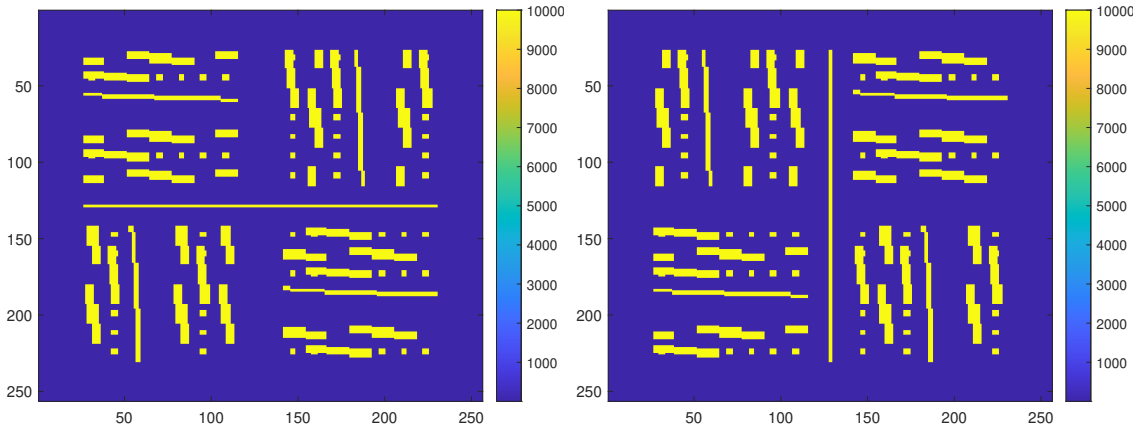
Combining (2.94), (2.95) and (2.100), we get the result in the theorem. □

2.4 Numerical results

In this section, we present two numerical examples with high-contrast media to verify the convergence of our proposed method, using a fine-scale approximation p_{f} as a reference solution. We will compute the coarse cell average $\bar{p}_{\mathrm{f}}$ of the fine-scale solution p_{f} and $\bar{p}_{\mathrm{ms}}$ of the multiscale solution p_{ms}, and compare the relative L^2 error of coarse cell average, i.e.

$$e_{L^2}^{(i)}=\left\|\bar{p}_{\mathrm{f},i}-\bar{p}_{\mathrm{ms},i}\right\|_{L^2},\qquad\left\|\bar{p}_{\mathrm{f},i}-\bar{p}_{\mathrm{ms},i}\right\|_{L^2}^2=\frac{\sum_K(\bar{p}_{\mathrm{f},i}^K-\bar{p}_{\mathrm{ms},i}^K)^2}{\sum_K(\bar{p}_f^K)^2},\qquad\bar{p}_{\mathrm{f},i}^K=\frac{1}{|K|}\int_K p_{\mathrm{f},i}\mathrm{d}x.\tag{2.101}$$

In all the experiments, we take the spatial domain to be $\Omega=(0,1)^2$ and the fine mesh size to be $h=1/256$. An example of the media κ_1 and κ_2 used in the experiments is illustrated in Figure 2.2. In the figure, the contrast values, i.e. the ratio of the maximum and the minimum in Ω, of the media are $\overline{\kappa}_1=\overline{\kappa}_2=10^4$. Unless otherwise specified, we set $\sigma=1$.



(a) κ_1 (b) κ_2

Figure 2.2: High contrast permeability field for the experiments.

2.4.1 Experiment 1

In this experiment, we consider the dual continuum model under steady state

$$\begin{aligned} -\text{div}(\kappa_1 \nabla p_1) + \sigma(p_1 - p_2) &= f_1, \\ -\text{div}(\kappa_2 \nabla p_2) - \sigma(p_1 - p_2) &= f_2. \end{aligned} \tag{2.102}$$

The source terms are given as $f_1(x, y) = 1$ and $f_2(x, y)$ as shown in Figure 2.3. In Figure 2.4, we plot the fine-scale solution, the coarse-scale average and the NLMC coarse-scale solution with coarse mesh size $H = 1/64$ and number of oversampling layers $m = 8$, from which we observe very good agreement between the coarse-scale average and the NLMC solution. In Table 2.1, we present the relative L^2 error with varying coarse grid size. With the number of oversampling layers satisfying the sufficient condition, we can see that the error converges. The order of convergence is numerically calculated from the slope of the best fit line of the log-log plot of the error against the mesh size. As reflected in Figure 2.5, the order of convergence in this numerical experiment is higher than the theoretical one.

We also compare the performance of different numbers of oversampling layers under fixed coarse mesh size H. The results are summarized in Table 2.2 for $H = 1/32$ and Table 2.3 for

$H = 1/64$. It can be seen that the error decays quickly with respect to the number of oversampling layers m for both cases, which verifies the fact that the oversampling region has to be sufficiently large to obtain quality numerical approximations.

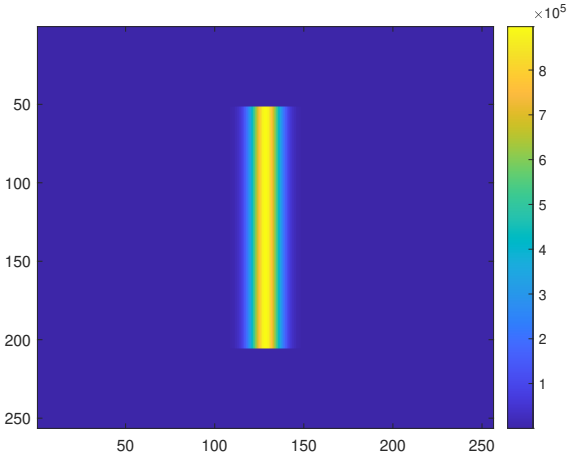


Figure 2.3: Source term f_2 in Experiment 1.

H	m	Area Ratio	$e_{L^2}^{(1)}$	$e_{L^2}^{(2)}$
1/8	3	87.5%	96.7318%	88.9818%
1/16	5	47.27%	32.3229%	19.3473%
1/32	6	16.50%	0.6045%	0.3680%
1/64	8	7.06%	0.0550%	0.0296%

Table 2.1: Convergence of e_{L^2} with respect to coarse mesh size H in Experiment 1.

2.4.2 Experiment 2

In this experiment, we consider the dual continuum model with time dependency. This case faces the similar issue with the error. $f_1(x, y) = 1$ and $f_2(x, y)$ is depicted in Figure 2.6, which represents a simplified five-spot well rate. The temporal domain is $[0, T]$ with final time $T = 5$. In Figure 2.7, we plot the fine-scale solution, the coarse-scale average and the NLMC coarse-scale

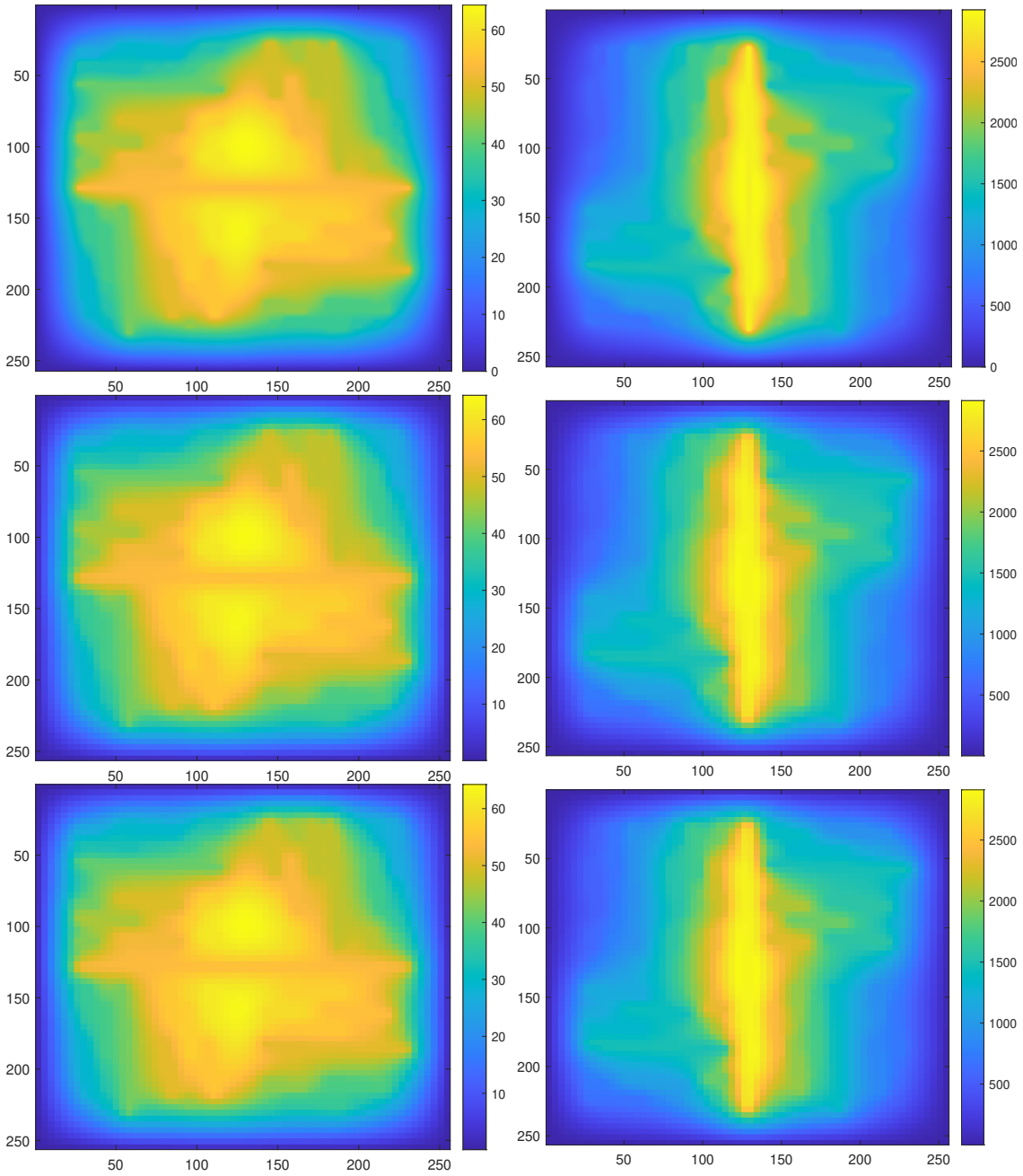


Figure 2.4: Plots of the numerical approximations of pressure with coarse mesh size $H = 1/64$ and $m = 8$ oversampling layers in Experiment 1. Left: first continuum. Right: second continuum. First row: fine-scale solution. Second row: coarse-scale average of fine-scale solution. Third row: NLMC solution.

solution with coarse mesh size $H = 1/64$ and number of oversampling layers $m = 8$. Again, the NLMC solution is a good approximation for the coarse-scale average. In Figure 2.9, we depict

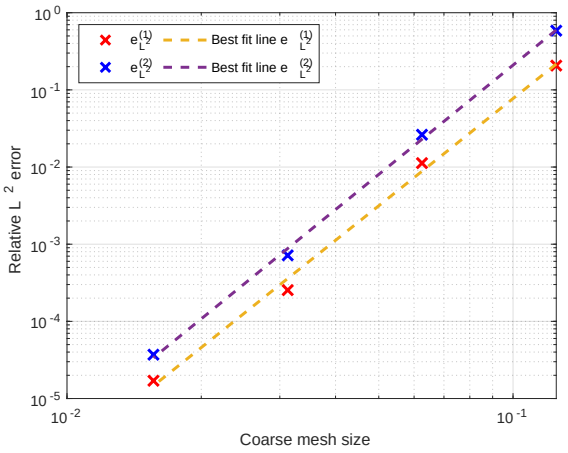


Figure 2.5: Log-log scale plot of relative error and coarse mesh size in Experiment 1. Slope of the best fit line for $e_{L^2}^{(1)}$ is 3.8082. Slope of the best fit line for $e_{L^2}^{(2)}$ is 4.0377.

m	Area Ratio	$e_{L^2}^{(1)}$	$e_{L^2}^{(2)}$
3	4.79%	99.3677%	93.9357%
4	7.91%	76.6083%	55.1631%
5	11.81%	8.6605%	5.3115%
6	16.50%	0.6045%	0.3680%

Table 2.2: Comparison of e_{L^2} error with different number of oversampling layers m for $H = 1/32$ in Experiment 1.

m	Area Ratio	$e_{L^2}^{(1)}$	$e_{L^2}^{(2)}$
2	0.61%	99.9102%	97.9631%
4	1.98%	99.1268%	91.9240%
6	4.13%	11.8898%	6.3181%
7	5.49%	0.7959%	0.4219%
8	7.06%	0.0550%	0.0296%

Table 2.3: Comparison of e_{L^2} error with different number of oversampling layers m for $H = 1/64$ in Experiment 1.

the change of pressure at different time steps. In Table 2.4, we present the relative L^2 error with varying coarse grid size. Again, the error converges with the the coarse mesh size while the number of oversampling layers satisfying the sufficient condition. As reflected in Figure 2.8, the order of

convergence in this numerical experiment is higher than the theoretical one.

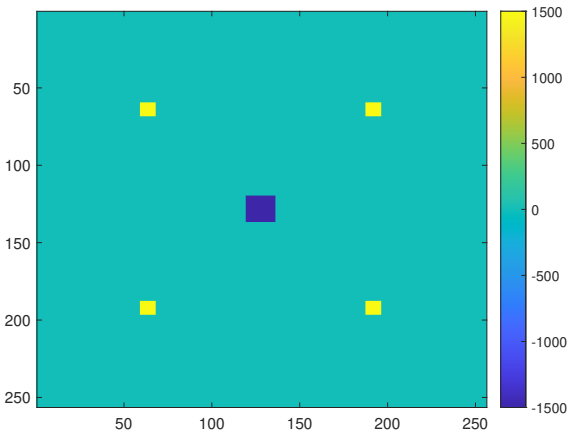


Figure 2.6: Source term f_2 in Experiment 2.

H	m	Area Ratio	$e_{L^2}^{(1)}$	$e_{L^2}^{(2)}$
1/8	3	87.5%	20.6422%	58.4975%
1/16	5	47.27%	1.1245%	2.6226%
1/32	6	16.50%	0.0254%	0.0717%
1/64	8	7.06%	0.0017%	0.0037%

Table 2.4: Convergence of e_{L^2} with respect to coarse mesh size H in Experiment 2.

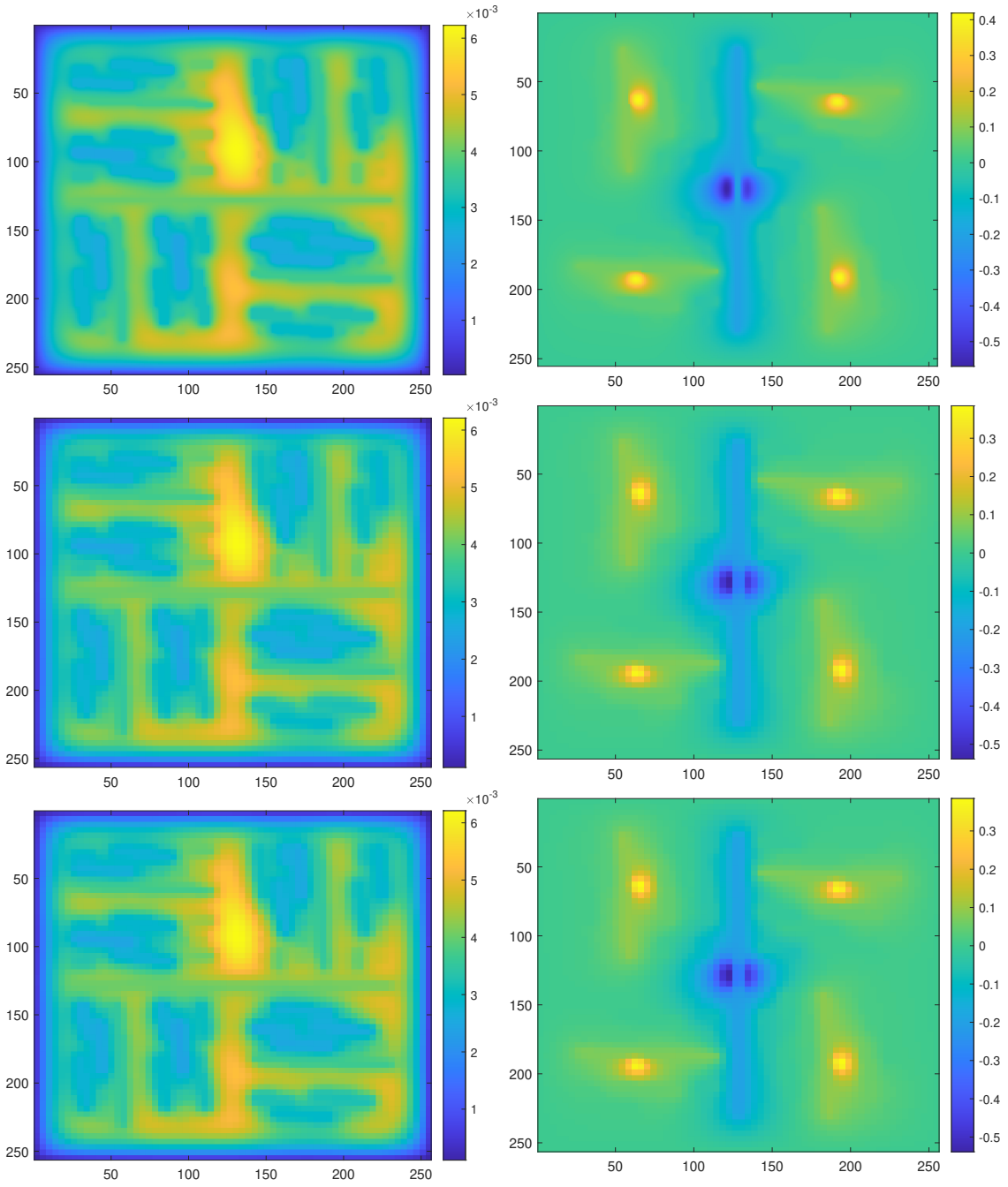


Figure 2.7: Plots of the numerical approximations of final-time pressure with coarse mesh size $H = 1/64$ and $m = 8$ oversampling layers in Experiment 2. Left: first continuum. Right: second continuum. First row: fine-scale solution. Second row: coarse-scale average of fine-scale solution. Third row: NLMC solution.

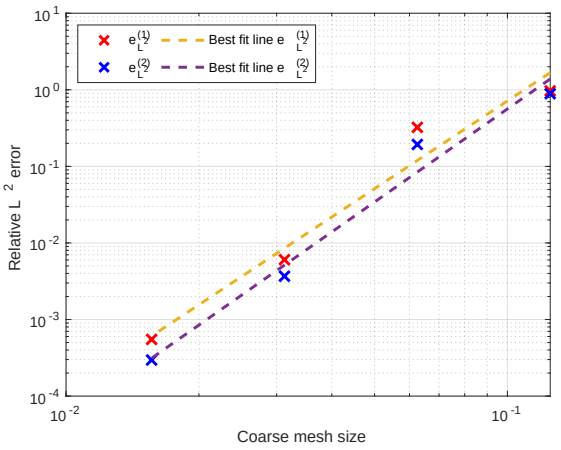


Figure 2.8: Log-log scale plot of relative error and coarse mesh size in Experiment 2. Slope of the best fit line for $e_{L^2}^{(1)}$ is 4.6172. Slope of the best fit line for $e_{L^2}^{(2)}$ is 4.7039.

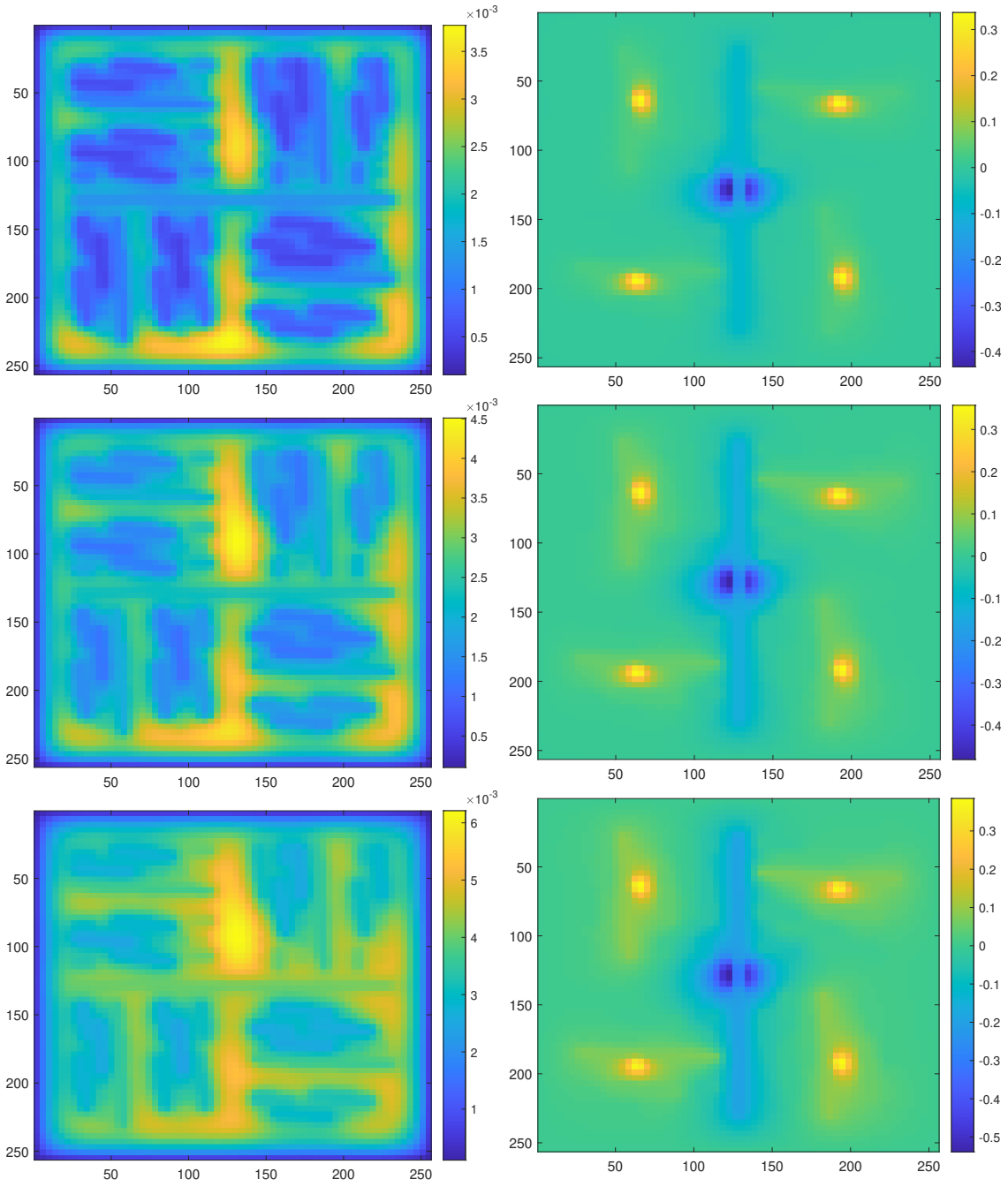


Figure 2.9: Plots of the NLMC numerical approximations of pressure at various time instants with coarse mesh size $H = 1/64$ and $m = 8$ oversampling layers in Experiment 2. Left: first continuum. Right: second continuum. First row: $t = 1.25$. Second row: $t = 2.5$. Third row: $t = 5$.

3. DEEP MODEL REDUCTION-MODEL LEARNING FOR RESERVOIR SIMULATION *

In this chapter, we design novel multi-layer neural network architectures for simulations of multi-phase flow taking into account the observed data (e.g., production data) and physical modeling concepts. Our approaches use deep learning concepts combined with model reduction methodologies to predict multi-phase flow dynamics. The use of reduced-order model concepts is important for constructing robust deep learning architectures. The reduced-order models provide fewer degrees of freedom and allow handling the cases relevant to reservoir engineering that is limited to production and near-well data.

Multi-phase flow dynamics can be thought as multi-layer networks. More precisely, the solution, pressures and saturation, at the time instant n+1 depends on the solution at the time instant n and input parameters, such as permeability, well rates, and so on. Thus, one can regard the solution as a multi-layer network, where each layer is a nonlinear forward map. The number of time steps is user-defined quantity, which will be treated as an unknown within our deep learning algorithms. We will rely on rigorous model reduction concepts to define unknowns and connections for each layer. Novel proper orthogonal basis functions will be constructed such that the degrees of freedom have physical meanings (e.g., represent the solution values at selected locations) and basis functions have limited support, which will allow localizing the forward dynamics. This will allow writing the forward map for the solution values at selected locations with pre-computed neighborhood structure that will be used in deep learning algorithms.

In each layer, our reduced-order models will provide a forward map, which will be modified ("trained") using available data. It is critical to use reduced-order models for this purpose, which will identify the regions of influence and the appropriate number of variables. Because of the lack of available data, the training will be supplemented with computational data as needed and

*Republished with permission of Society of Petroleum Engineers (SPE), from "Deep Model Reduction-Model Learning for Reservoir Simulation", by Jingyan Zhang, Siu Wun Cheung, Yalchin Efendiev, Eduardo Gildin, and Eric T. Chung, 2019. SPE Reservoir Simulation Conference, Copyright [2019] by Society of Petroleum Engineers; permission conveyed through Copyright Clearance Center, Inc.

the interpolation between data-rich and data-deficient models. We will also use deep learning algorithms to train the elements of the reduced model discrete system. In this case, deep learning architectures will be employed to approximate the elements of the discrete system and reduced-order model basis functions.

The numerical results will use deep learning architectures to predict the solution and reduced-order model variables. Trained basis functions will allow interpolating the solution between the observation points. We show how network architecture, which includes the neighborhood connection, number of layers, and neurons, affect the approximation. Our results show that with a fewer number of layers, the multi-phase flow dynamics can be approximated. The proposed approach uses physical model concepts and deep learning methods to design a novel forward map, which combines the available data and physical models. This will benefit to develop a fast and data-based algorithms for reservoir simulations.

The chapter is organized as follows. In Section 3.1, we introduce the governing equations for the two-phase flow model and summarize the background methodology for POD-DEIM based model reduction. We also include a subsection on Nodal Basis Functions which will be used to connect Deep Learning and POD. In Section 3.2, the main idea of deep learning is introduced and its connection to model reduction is investigated. A few test cases will be explored in Section 3.3. Some discussion is also included in this section.

3.1 Preliminaries

In this section, we discuss the governing equations of the two-phase oil-water flow in porous media, and a numerical solver for the problem with Proper Orthogonal Decomposition (POD) for model order reduction and Discrete Empirical Interpolation Method (DEIM) for approximations of nonlinear source functions.

Let Ω be the reservoir domain, in which the two phases, namely water (denoted by subscript w) and oil (denoted by subscript o), of the flow are immiscible. The flow equation of the phase

velocity u_a is described by the Darcy's law:

$$u_a = -\mu_a^{-1} k_{ra}(s)\kappa \cdot \nabla p, \tag{3.1}$$

where κ is the permeability tensor, k_{ra} is the relative permeability to phase a, s is the (water) saturation, and p is the pressure, for each phase $a = w, o$. Here, we assume the displacement is dominated by viscous effects, so that the gravitational acceleration, capillary pressure effects and compressibility effects can be neglected. The Darcy's law is coupled with the conservation of mass and this yields a initial-boundary value problem, known as the pressure-saturation equations:

$$\begin{aligned} -\nabla \cdot (\lambda(s)\kappa \nabla p) &= q_w + q_o \quad \text{in } \Omega, \\ \phi \frac{\partial s}{\partial t} + \nabla \cdot (f_w(s)u) &= \frac{q_w}{\rho_w} \quad \text{in } \Omega, \\ u \cdot n &= 0 \quad \text{on } \partial\Omega, \\ s &= 0 \quad \text{at } t = 0. \end{aligned} \tag{3.2}$$

Here, q_a is the volumetric source term for phase a, $u = u_w + u_o$ is the total velocity, ϕ is the porosity, λ is the total mobility and $f(s)$ is the flux function. To be precise,

$$\begin{aligned} \lambda(s) &= \lambda_w(s) + \lambda_o(s) = \frac{k_{rw}(s)}{\mu_w} + \frac{k_{ro}(s)}{\mu_o}, \\ f_w(s) &= \frac{\lambda_w(s)}{\lambda(s)} = \frac{k_{rw}(s)}{k_{rw}(s) + \frac{\mu_w}{\mu_o} k_{ro}(s)}. \end{aligned} \tag{3.3}$$

For each time step, we first solve for the pressure and velocity with a mixed finite element method and then solve for the saturation solutions with a mass conservative finite volume in a backward Euler discretization method. In a cell Ω_i, the nonlinear equation for the evolution of saturation, s_i is given by

$$s_i^{n+1} = s_i^n + \frac{\Delta t}{|\Omega_i|}(q^+ - \sum_j F_{ij}(s^{n+1})u_{ij} + f_w(s_i^{n+1})q^-). \tag{3.4}$$

where the superscript $n+1$ denotes the time t_{n+1} and the superscripts $+$ and $-$ stand for the positive and negative part respectively. F_{ij} is the numerical approximation of flux over the edge of cells Ω_i and Ω_j, γ_{ij}. To be precise,

$$F_j \approx \int_{\gamma_{ij}} (f_{ij}(s)u_{ij}) \cdot n_{ij} \mathrm{d}v \tag{3.5}$$

where n_{ij} is the normal vector pointing out of Ω_i associated to γ_{ij}.

In what follows, we will apply POD to the pressure, velocity and saturation equations to reduce the computational cost associate with their evaluations, and DEIM to the flux function, to reduce the complexity of evaluating the non-linear function.

3.1.1 POD-DEIM

In this section, we give a brief summary on the global model reduction methods of Proper Orthogonal Decomposition (POD) and Discrete Empirical Interpolation Method (DEIM).

3.1.1.1 Proper Orthogonal Decomposition

POD starts from a snapshot matrix of state solutions,

$$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{N_s}] \in \mathbb{R}^{N \times N_s} \tag{3.6}$$

where N is the number of grid blocks and N_s is the number of snapshots. The goal is to find an orthonormal basis $\{\phi_i\}_{i=1}^r \subset \mathbb{R}^N$ such that

$$\underset{\{\phi_i\}_{i=1}^r}{\operatorname{argmin}} \sum_{j=1}^{N_s} \|\mathbf{y}_j - \sum_{i=1}^{n} (\mathbf{y}_j^T \phi_i)\phi_i\|_2^2. \tag{3.7}$$

The above minimization problem can be solve by applying singular value decomposition (SVD) to the snapshot matrix,

$$\mathbf{y} = \mathbf{V}\mathbf{\Lambda}\mathbf{W}^T. \tag{3.8}$$

Here, $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$ and $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ are the left and right singular matrix consisting

of singular vectors ($\mathbf{v}_i$ and $\mathbf{w}_i$) of $\mathbf{y}$ respectively, where $m = \dim(\mathbf{S})$. $\mathbf{\Lambda} = \mathrm{diag}(\sigma_1, \ldots, \sigma_m)$ is a diagonal matrix containing the singular values of $\mathbf{y}$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m > 0$. The number of basis, r is determined by the captured fractional energy,

$$E = \frac{\sum_{i=1}^{r} \sigma_i}{\sum_{i=1}^{N_s} \sigma_i}. \tag{3.9}$$

Normally, r is selected such that $0.9 < E < 1$. The basis $\{\phi_i\}_{i=1}^{r}$ is then determined by $\mathbf{v}_1, \ldots, \mathbf{v}_r$. If the singular values decay fast, only a small number of POD basis will be needed. DEIM further improves the efficiency of model reduction by avoiding the extensive computations of the nonlinear functions as briefly explained below.

3.1.1.2 Discrete Empirical Interpolation Method

Given a nonlinear function $f(\tau) \in \mathbb{R}^N$, where τ is time t or other control parameter μ. We first collect the snapshots of f. Then, by singular value decomposition, a projection matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m] \in \mathbb{R}^{N \times M}$ is obtained. f can then be approximated by

$$f(\tau) \approx \mathbf{U}c(\tau). \tag{3.10}$$

To determine the coefficient vector $c(\tau)$, we first apply greedy strategy to select M distinct interpolation points $\{p_1, \ldots, p_M\}$. Next, we assemble the selection matrix

$$\mathbf{P} = [\mathbf{e}_{p_1}, \ldots, \mathbf{e}_{p_M}] \in \mathbb{R}^{N \times M} \tag{3.11}$$

where $\mathbf{e}_i$ is the i-th canonical unit vector. Assuming $\mathbf{P}^T\mathbf{U}$ is nonsingular, we have

$$f(\tau) \approx \mathbf{U}c(\tau) = \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T f(\tau). \tag{3.12}$$

3.1.2 Nodal basis functions

In this section, we briefly present the idea of constructing nodal basis functions. This enables us to use coefficients that represent the solution values at selected locations in the mesh such that the coefficients will have physical meaning.

Given a set of nodes $\{x_k\}_{k=1}^m$ corresponds to particular physical spot in the spatial domain, we construct nodal basis functions as a linear combination of our obtained POD basis functions $\{\mathbf{v}_k\}_{k=1}^m$ by seeking coefficients α_{ij} such that

$$\sum_{k=1}^{m} \alpha_{ij} v_{jk} = \delta_{ik}. \tag{3.13}$$

Here, v_{jk} represents the value of $\mathbf{v}_j$ at x_k. The nodal basis is then constructed as follows.

$$\psi_i = \sum_{k=1}^{m} \alpha_{ik} \mathbf{v}_k. \tag{3.14}$$

Some examples of the nodal basis functions are illustrated in Figure 3.1.

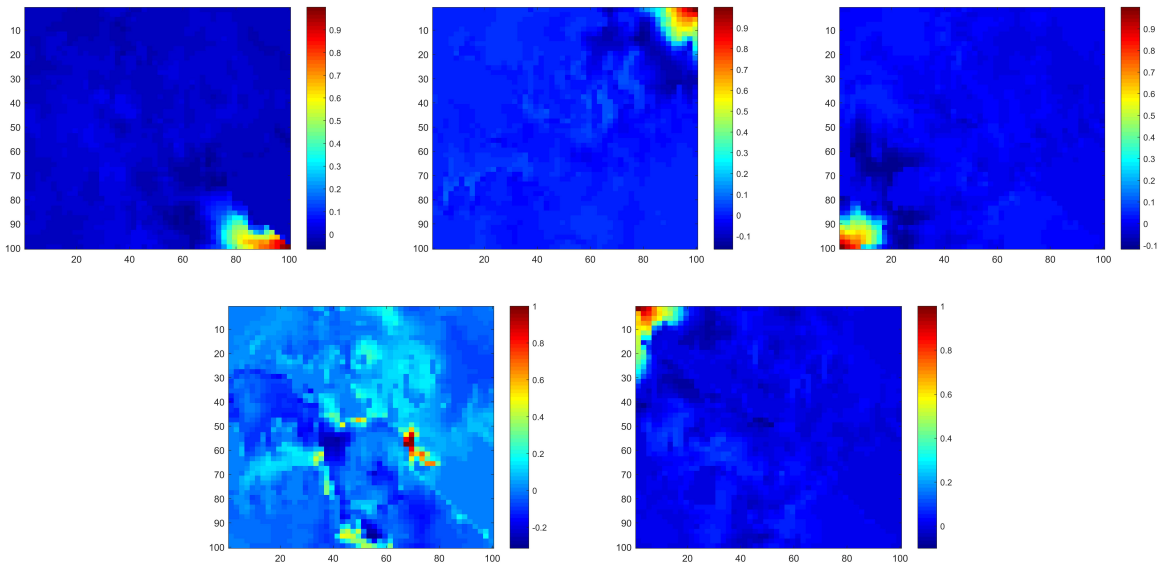


Figure 3.1: Illustrations of nodal basis functions.

3.2 Deep global model reduction learning

3.2.1 Main idea

The idea is to combine deep learning concepts with our reduced-order model to provide a numerical model efficient in modeling flow profile. We apply the POD-DEIM method to a two-phase flow (oil-water) reservoir model under the water flooding recovery process. Given m samples of training set, we use POD-DEIM to obtain coarse grid solution coefficients of one unique set of basis at all time steps. Since we are using nodal basis functions, the coefficients we obtain represent values of solutions on selected locations. We try, by training these coefficients, to establish a forward map between saturation solutions of time step n and time step $n+1$. That is

$$s^{n+1} = \mathcal{N}(s^n, I^{n+1}) \tag{3.15}$$

Here, $\mathcal{N}$ is the multi-layer neural network to be trained and I^n can be permeability fields or injection and production rates and so on. An illustration of a deep neural network in our work is depicted in Figure 3.2.

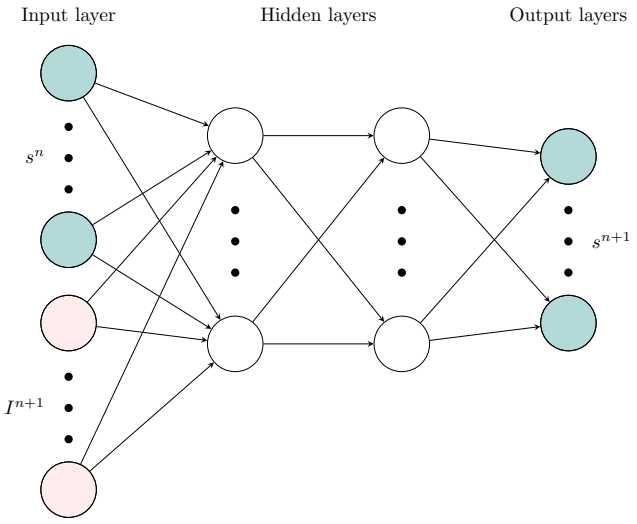


Figure 3.2: An illustration of deep neural network.

We also consider available observed data, s_o^n, apart from simulation data, s_c^n. With observed data, there are three different types of networks to be considered.

1. Pure simulation network: Use only simulation data as output,

$$s_c^{n+1} = \mathcal{N}_s(s_c^n, I^{n+1}) \tag{3.16}$$

2. Pure observation network: Use only observation data as output,

$$s_o^{n+1} = \mathcal{N}_o(s_o^n, I^{n+1}) \tag{3.17}$$

3. Mixed network: Use a mixture of simulation and observation data as output,

$$s_m^{n+1} = \mathcal{N}_m(s_m^n, I^{n+1}) \tag{3.18}$$

The pure simulation network describes only simulation model and can be used as a fast simulator. The pure observation network, on the other hand, corresponds to the situation when observation data is sufficient. The mixed network take both simulation and observation data into training process and leads to a data-driven model.

3.2.2 Network structure

In this section, we briefly summarize the architecture of the networks $\mathcal{N}_\alpha$, where $\alpha = s, o, m$ as defined in (3.16), (3.17) and (3.18).

For each neural network, we take $\mathbf{x} = (s^n, \theta, q^n)$ as the input vector, which contains both coefficients of coarse scale solution, permeability parameter and source term of a particular time step. The output data will be $\mathbf{y} = s^{n+1}$, containing coefficients of coarse scale solutions in the next time step. Mathematically, the neural network can be written as

$$\mathcal{N}_\alpha(\mathbf{x}; \eta) = \sigma(W_L \sigma(\cdots \sigma(W_1 \mathbf{x} + b_1) + \cdots) + b_L). \tag{3.19}$$

Here, L is the number of layers of the neural network, of which layers 1 to $L-1$ are hidden layers and the Lst layer is the output layer. In our work, L varies from 2 to 11 and in each hidden layer, the number of neurons varies from 12 to 300 in different experiments. For the convenience of our work, we choose to use Leaky ReLU [65] instead of ReLU [66] as the activation function σ, to avoid slow convergence.

In (3.19), W_i and b_i are the weight matrices and bias vectors of layer i respectively. We define $\eta = \{W_1, \ldots, W_L, b_1, \ldots, b_L\}$ as a set of parameters to be optimized. To be precise, given data pairs $(\mathbf{x}_i, \mathbf{y}_i)$, the deep neural network aims at finding the parameter η^* such that

$$\eta^* = \underset{\eta}{\operatorname{argmin}} \frac{1}{m} \sum_{j=1}^{m} \|\mathbf{y}_i - \mathcal{N}(\mathbf{x}_i; \eta)\|_2^2, \tag{3.20}$$

where m is the number of training samples and $L(\eta) = \frac{1}{m} \sum_{j=1}^{m} \|\mathbf{y}_i - \mathcal{N}(\mathbf{x}_i; \eta)\|_2^2$ is the cost function or loss function. We use AdaMax [67] as optimizer to minimize the loss function of our high-dimensional parameters. Python deep learning API Keras [68, 69] is used for the training and testing of all the neural networks.

3.3 Numerical results

In our first two numerical examples, we use the basic model from the SPE10 benchmark data set. We have an injector in the center of the reservoir and four producers in the corners. The permeability models are based on first few layers of SPE10 data. To this end, we set the same permeability fields for all examples. To be precise, as our permeability setting, we first take two initial permeability fields, κ_1 and κ_2. The uncertainties in the permeability fields are parametrized by a convex combination

$$\kappa(\theta) = (1-\theta)\kappa_1 + \theta\kappa_2, \tag{3.21}$$

for $0 \leq \theta \leq 0.1$. Examples of $\kappa(\theta)$ are demonstrated in Figure 3.3.

For each source term and permeability parameter given, we run POD-DEIM simulation on the reservoir for 300 days and collected the coefficients of saturation solutions every 30 days. This

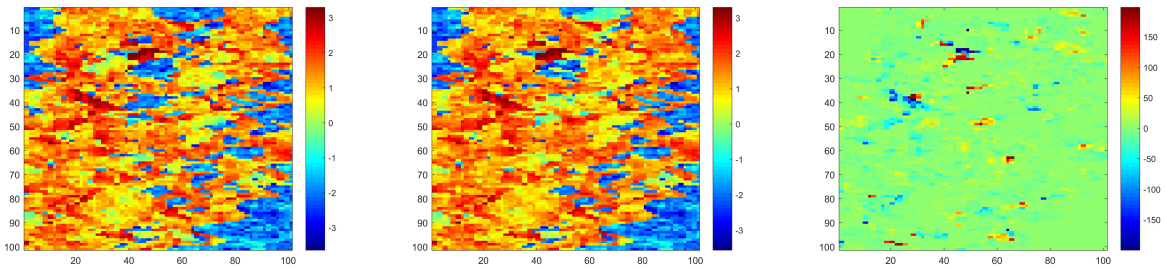


Figure 3.3: Figures of permeability fields. Left: plot of logarithm of $\kappa(0)$. Middle: plot of logarithm of $\kappa(0.1)$. Right: difference of $\kappa(0)$ and $\kappa(0.1)$. Republished with permission of Society of Petroleum Engineers (SPE), from "Deep Model Reduction-Model Learning for Reservoir Simulation", by Jingyan Zhang, Siu Wun Cheung, Yalchin Efendiev, Eduardo Gildin, and Eric T. Chung, 2019. SPE Reservoir Simulation Conference, Copyright [2019] by Society of Petroleum Engineers; permission conveyed through Copyright Clearance Center, Inc.

serves as simulation data in our work. The observation data are generated from fine grid solver. Data-driven models will be investigated and, therefore, observation data will be supplemented as needed, by randomly mixing different realizations from simulation and observation data. We split the the output response in two equal data sets in which half of the data is simulation data and half is observation data.

The neural network models are trained to map particular time step solutions onto the next time step. All data are divided into training and testing data accordingly so that only unseen samples will be used to perform prediction. To evaluate the efficiency of our models, we will feed them with testing samples to perform two types of data prediction as follows.

1. One-step prediction

$$s^{n+1} = \mathcal{N}(s^n, \theta, q^{n+1}; \eta^*) \tag{3.22}$$

2. Multi-step (final time) prediction

$$s^{10} = \mathcal{N}(\ldots \mathcal{N}(s^0, \theta, q^1; \eta^*), \ldots, \theta, q^{10}; \eta^*) \tag{3.23}$$

Given time step n, observation percentage errors will be used for comparing reference solution

s^n_{ref} and our predicted solution s^n_{pred}. To be precise, we compute

$$e^n_{\text{pred}} = \frac{\left\| s^n_{\text{ref}} - s^n_{\text{pred}} \right\|_{L_2(\Omega)}}{\left\| s^n_{\text{ref}} \right\|_{L_2(\Omega)}}. \tag{3.24}$$

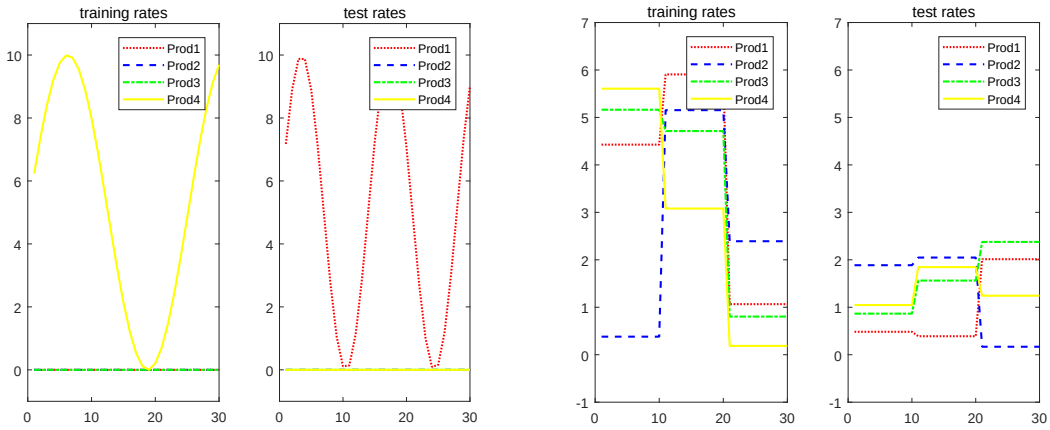


Figure 3.4: Illustrations of production rates for training and testing. Left: Experiment 1. Right: Experiment 2. Republished with permission of Society of Petroleum Engineers (SPE), from "Deep Model Reduction-Model Learning for Reservoir Simulation", by Jingyan Zhang, Siu Wun Cheung, Yalchin Efendiev, Eduardo Gildin, and Eric T. Chung, 2019. SPE Reservoir Simulation Conference, Copyright [2019] by Society of Petroleum Engineers; permission conveyed through Copyright Clearance Center, Inc.

3.3.1 Experiment 1

In this example, for each permeability parameter, we use $q = 5(1+\sin(\beta t))$ as production rate and $q = -5(1+\sin(\beta t))$ as injection rate, where $0.15 < \beta < 0.5$. Only one producer and one injector is used. An example of training and testing production rate schedules is given in Figure 3.4. Every simulation realization is unique in terms of well rates combined with permeability field. In all, we have 3520 samples. 3170 of these samples are used for training and 350 for testing. We perform model training and predictions based on two sets of basis. The number of basis is 25 or 5. The observation points corresponds to the basis are demonstrated in Figure 3.5.

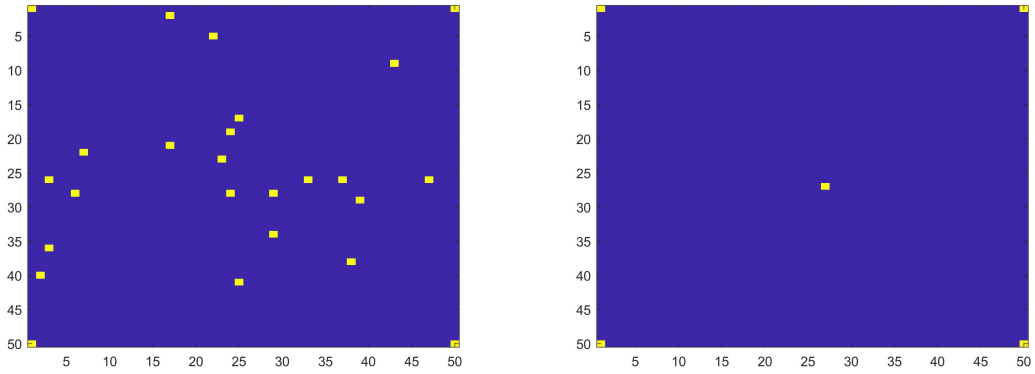


Figure 3.5: Figures of observation points in experiment 1. Left: 25 nodal basis case. Right: 5 nodal basis case. Republished with permission of Society of Petroleum Engineers (SPE), from "Deep Model Reduction-Model Learning for Reservoir Simulation", by Jingyan Zhang, Siu Wun Cheung, Yalchin Efendiev, Eduardo Gildin, and Eric T. Chung, 2019. SPE Reservoir Simulation Conference, Copyright [2019] by Society of Petroleum Engineers; permission conveyed through Copyright Clearance Center, Inc.

Basically, under the same number of neurons, all models of the same type perform about the same in terms of one-step prediction. The pure-observation networks ($\mathcal{N}_o$) indicate a decline in error with the rise of the number of layers. The results also demonstrated that when observation data is available, the performance of our networks can be improved. On the other hand, the return of increasing number of layers is eventually marginal, but shallow networks are not able to fully interpret all the intrinsic data.

3.3.1.1 25 nodal basis case

In this experiment, we use 25 nodal POD basis based on the assumption that we have available data from 25 observation spots. The observational locations are demonstrated in Figure 3.5. We train two set of neural networks in this experiment, both with different numbers of hidden layers. One set of neural network is with 300 neurons and with 300 neurons in each hidden layer, the other set is with 64 neurons in each hidden layer. The results are listed in Table 3.1 and Table 3.2.

Hidden layer #	1-step			Final time		
	$\mathcal{N}_s$	$\mathcal{N}_m$	$\mathcal{N}_o$	$\mathcal{N}_s$	$\mathcal{N}_m$	$\mathcal{N}_o$
1	11.94%	7.71%	3.37%	16.43%	7.25%	8.14%
3	11.66%	7.30%	2.50%	15.53%	12.89%	4.01%
5	11.55%	7.40%	2.20%	14.61%	10.60%	3.25%
10	11.59%	6.80%	2.15%	14.98%	8.93%	6.38%

Table 3.1: Mean of observation percentage error with different training data in 64 neuron network, Experiment 1. Republished with permission of Society of Petroleum Engineers (SPE), from "Deep Model Reduction-Model Learning for Reservoir Simulation", by Jingyan Zhang, Siu Wun Cheung, Yalchin Efendiev, Eduardo Gildin, and Eric T. Chung, 2019. SPE Reservoir Simulation Conference, Copyright [2019] by Society of Petroleum Engineers; permission conveyed through Copyright Clearance Center, Inc.

Layer	1-step			Final time		
	Simulation	Mixed	Observation	Simulation	Mixed	Observation
1	1.15%	2.25%	1.77%	18.98%	11.49%	2.77%
3	0.59%	1.51%	1.33%	14.97%	8.14%	1.65%
5	0.78%	1.33%	1.05%	15.00%	8.07%	1.06%
10	0.80%	1.26%	1.52%	14.80%	8.13%	1.66%

Table 3.2: Mean of observation percentage error with different training data in 300 neuron network, Experiment 1. Reprinted with permission from "Deep Global Model Reduction Learning in Porous Media Flow Simulation" by Siu Wun Cheung, Eric T. Chung, Yalchin Efendiev, Eduardo Gildin, Yating Wang and Jingyan Zhang, 2020. Computational Geosciences, Volume 24, Pages 261–274, Copyright [2020] by Springer.

3.3.1.2 5 nodal basis case

In this experiment, we use 5 nodal POD basis based on the assumption that we have available data from 5 observation spots. The observational locations are demonstrated in Figure 3.5. We remark that the observational locations are all near the injector and producers. We have two sets of models for this experiment, one has 120 neurons in each hidden layer, the other is with 12 neurons in each hidden layer. We also tried 300-neuron models but the results are almost the same as the 120-neuron ones. The errors of predictions are listed in Table 3.3.

Neuron	Hidden layer #	1-step			Final time		
		$\mathcal{N}_s$	$\mathcal{N}_m$	$\mathcal{N}_o$	$\mathcal{N}_s$	$\mathcal{N}_m$	$\mathcal{N}_o$
120	1	25.04%	13.90%	1.29%	21.07%	13.01%	6.28%
	3	25.21%	13.68%	1.25%	21.89%	14.23%	8.81%
	5	25.16%	13.90%	0.96%	22.09%	12.84%	1.49%
	10	25.08%	13.56%	1.02%	22.08%	13.00%	2.12%
12	1	24.95%	14.65%	2.38%	23.81%	14.09%	5.44%
	3	25.27%	13.95%	2.41%	23.64%	9.20%	8.11%
	5	25.25%	13.90%	1.87%	22.66%	11.79%	3.67%
	10	25.21%	13.94%	1.80%	22.01%	13.03%	8.19%

Table 3.3: Mean of observation percentage error with different training data in 5 nodal basis case in Experiment 1. Republished with permission of Society of Petroleum Engineers (SPE), from "Deep Model Reduction-Model Learning for Reservoir Simulation", by Jingyan Zhang, Siu Wun Cheung, Yalchin Efendiev, Eduardo Gildin, and Eric T. Chung, 2019. SPE Reservoir Simulation Conference, Copyright [2019] by Society of Petroleum Engineers; permission conveyed through Copyright Clearance Center, Inc.

3.3.2 Experiment 2

In this experiment, we use all four producers in the corners and, naturally, the injector in the center. All well rates are random step functions restricted by a scale ranging from 1 to 4. For each permeability field and under each scale, we generate 20 random rate schedules. An example of training and testing production rate schedule is given in Figure 3.4. In all, we have 15400 samples, among which we take 13860 for training and 1540 for testing. The data are based on both 5 and 25 nodal POD basis. The observation locations are demonstrated in Figure 3.6.

3.3.2.1 25 nodal basis case

Same as the previous experiments, we assume there are 25 observational locations available. Two sets of neural network models are trained in this case, one with 300 neurons in each hidden layer and one with 64 neurons in each hidden layer. The results are summarized in Table 3.4.

3.3.2.2 5 nodal basis case

In this experiment, we use 5 nodal POD basis based on the assumption that we have available data from 5 near-well observation spots. Same as the previous example, we have two sets of model

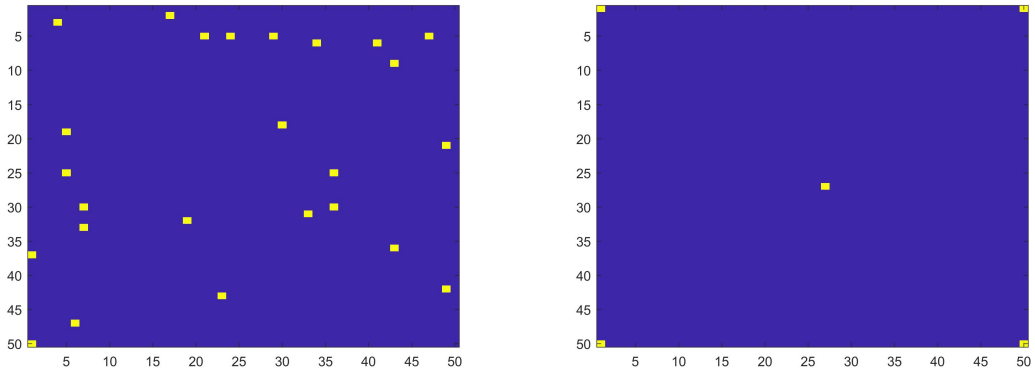


Figure 3.6: Figures of observation points for Experiment 2. Left: 25 nodal basis case. Right: 5 nodal basis case. Republished with permission of Society of Petroleum Engineers (SPE), from "Deep Model Reduction-Model Learning for Reservoir Simulation", by Jingyan Zhang, Siu Wun Cheung, Yalchin Efendiev, Eduardo Gildin, and Eric T. Chung, 2019. SPE Reservoir Simulation Conference, Copyright [2019] by Society of Petroleum Engineers; permission conveyed through Copyright Clearance Center, Inc.

Neuron	Hidden layer #	1-step			Final time		
		$\mathcal{N}_s$	$\mathcal{N}_m$	$\mathcal{N}_o$	$\mathcal{N}_s$	$\mathcal{N}_m$	$\mathcal{N}_o$
300	1	15.83%	8.91%	1.33%	18.81%	21.10%	4.13%
	3	15.91%	8.51%	1.22%	17.63%	9.13%	1.93%
	5	15.85%	8.59%	1.10%	18.13%	9.46%	1.53%
	10	15.79%	8.64%	1.05%	17.45%	10.19%	1.67%
64	1	16.02%	10.12%	4.10%	25.75%	22.93%	12.68%
	3	15.82%	9.76%	2.90%	21.12%	17.50%	7.81%
	5	15.64%	9.74%	2.41%	19.77%	16.38%	7.58%
	10	15.84%	9.48%	2.11%	18.06%	13.84%	5.65%

Table 3.4: Mean of observation percentage error with different training data in 25 nodal basis case in Experiment 2. Republished with permission of Society of Petroleum Engineers (SPE), from "Deep Model Reduction-Model Learning for Reservoir Simulation", by Jingyan Zhang, Siu Wun Cheung, Yalchin Efendiev, Eduardo Gildin, and Eric T. Chung, 2019. SPE Reservoir Simulation Conference, Copyright [2019] by Society of Petroleum Engineers; permission conveyed through Copyright Clearance Center, Inc.

for this experiment, one has 120 neurons in each hidden layer, the other has 12 neurons in each hidden layer. The errors of prediction are summarized in Table 3.5.

Neuron	Hidden layer #	1-step			Final time		
		$\mathcal{N}_s$	$\mathcal{N}_m$	$\mathcal{N}_o$	$\mathcal{N}_s$	$\mathcal{N}_m$	$\mathcal{N}_o$
120	1	25.91%	11.34%	0.57%	33.93%	16.95%	6.17%
	3	25.73%	11.22%	0.33%	30.38%	12.79%	0.94%
	5	26.05%	11.13%	0.25%	31.74%	13.54%	0.55%
	10	25.73%	11.27%	0.29%	29.65%	13.85%	0.71%
12	1	26.17%	11.98%	1.67%	33.29%	16.63%	2.80%
	3	26.17%	11.63%	1.35%	55.96%	11.50%	3.03%
	5	26.77%	11.59%	1.26%	30.46%	14.59%	2.26%
	10	26.42%	12.13%	1.17%	20.34%	19.94%	2.31%

Table 3.5: Mean of observation percentage error with different training data in 5 nodal basis case in Experiment 2. Republished with permission of Society of Petroleum Engineers (SPE), from "Deep Model Reduction-Model Learning for Reservoir Simulation", by Jingyan Zhang, Siu Wun Cheung, Yalchin Efendiev, Eduardo Gildin, and Eric T. Chung, 2019. SPE Reservoir Simulation Conference, Copyright [2019] by Society of Petroleum Engineers; permission conveyed through Copyright Clearance Center, Inc.

Both experiments demonstrate the efficiency of the neural networks. One-step prediction is not much affected by the architecture of the neural networks while, with a relatively larger number of layers of neurons, final time predictions will show a higher and good accuracy, even when the error of pure-simulation networks ($\mathcal{N}_s$) are nearly dominated by the simulation-observation error. This experiment also indicates that the available observation data provides improvement to the quality of prediction of final-time dynamics.

3.3.3 Experiment 3

Apart from using deep learning algorithm for modeling nonlinear forward dynamics, one can also use neural network for downscaling to reconstruct information. In this experiment, we briefly investigate this possibility.

From the fine-scale solutions using the same permeability and well rate settings as in Experiment 1, 991 samples are randomly chosen. We split these data into 891 training samples and 99 testing samples respectively. We build two neural networks for this downscaling procedure. The first neural network has 5 and 25 nodes as inputs and outputs respectively. The second neural network has 25 and 50 as inputs and outputs respectively. The 5 and 25 observation points are

demonstrated in Figure 3.5. The 50 observation points are selected using orthogonal-triangular decomposition and is shown in Figure 3.7. We depict an illustration of our first neural network in Figure 3.8.

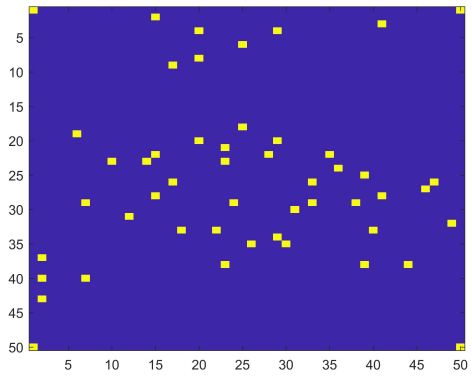


Figure 3.7: An illustration of 50 observation points for the second neural network used in Experiment 3.

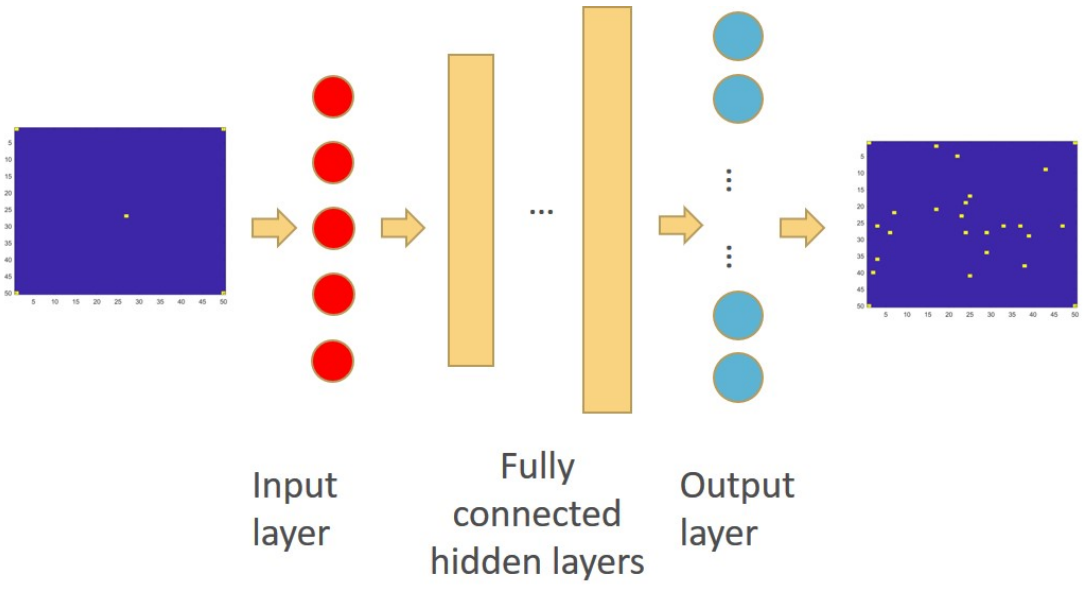


Figure 3.8: An illustration of neural network used in Experiment 3.

Since this procedure does not involve approximating temporal evolution of fluid dynamics, we can construct a relatively simple neural network to tackle the task. To be more precise, neural networks with only 2 hidden layers are used. Different from previous experiments, the number of neurons differs in the two hidden layers. For the neural network predicting 25 degrees of freedom, we use 64 neurons in the first hidden layer and 100 neurons in the second hidden layer. For the neural network predicting 50 nodal values, we use 64 and 200 neurons in the first and second hidden layers respectively. Leaky ReLU is still used as activation functions for faster convergence. Given available nodal basis functions, we are able to reconstruct fine-scale solutions from the inputs and outputs and compare them with the original fine-scale solution. The errors of predictions are listed in Table 3.6. A demonstration of data reconstruction and noise deduction is shown in Figure 3.9.

Output dimension	e_{L^1}	e_{L^2}	e_{L^∞}
25	4.02	2.78	2.73
50	1.69	1.05	1.03

Table 3.6: Mean of different percentage errors in Experiment 3.

.

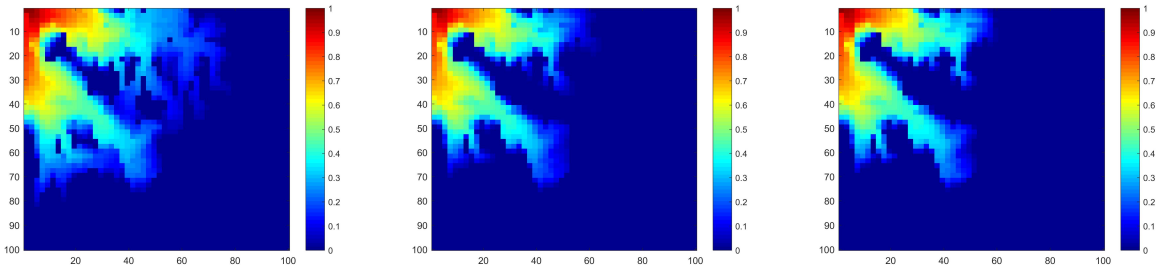


Figure 3.9: Figures of reconstructed and reference fine-scale solutions in Experiment 3. Left: fine-scale solution reconstructed from 5 nodal parameter input. Middle: fine-scale solution reconstructed from 25 nodal parameter output. Right: reference solution.

By applying neural networks to downscaling problems, we can breakdown the flow approximation into a two-step procedure. One can first forward approximate fluid dynamics with a small neural network and reconstruct information by downscaling neural networks. This can further decrease the training cost of the main dynamic model while maintaining accuracy.

4. IMAGE-BASED PHYSICS-CONSTRAINT WORKFLOW FOR MULTI-PHASE FLOW SIMULATION IN HETEROGENEOUS MEDIA

Physics-based simulations for multi-phase flow usually suffer from expensive computational costs induced by its nonlinearity and The trade-offs between efficiency and accuracy always attract attention from researchers. The development of deep learning techniques also shed lights on the possibility of maintaining fidelity in fluid dynamics with lower computational costs. In this chapter, we explore this possibility for nonlinear multi-phase fluid.

By breaking down the coupled system into pressure and saturation equations, we describe a hybrid workflow in predicting the evolution of pressure and simulating saturation. As opposed to an expensive implicit pressure solver, we construct deep neural networks for predicting pressure fields, leveraging the advantages of different image-based neural networks in capturing spatial and temporal patterns. The model takes inputs including permeability and production information, along with initial fluid status and predict images of pressure of different time steps. To maintain spatial connectivity of output pressure, physics-based loss penalties are used during the training process. The predicted pressure fields are fed into explicit numerical solvers for saturation simulation. This way, the accuracy of the workflow is good while the high computational cost is reduced. Performance and effectiveness of the aforementioned workflow is also discussed.

The chapter is organized as follows. In Section 4.1 we present the physics of two-phase flow and summarize the background methodology of the numerical simulation method. In Section 4.2, the idea of image-based neural networks are introduced and the construction of our proposed model is presented. Numerical experiments are explored in Section 4.3 to check the performance of our model. We will also include discussions of model performance in this section.

4.1 Preliminaries

As a benchmark ground truth reference for our to-be-developed neural networks, we carry out a sequential IMplicit Pressure Explicit Saturation (IMPES) [70, 71] formulation of the two-phase

flow system as follows.

4.1.1 Governing equation

Let Ω be the reservoir domain with two phases, water (denoted by subscript w) and oil (denoted by subscript o). Similar to Chapter 3, we assume that the flow is immiscible and the displacement is dominated by viscous effects, which enables us to disregard gravitational acceleration, capillary forces and compressibility effects. The Darcy's law yields the following initial-boundary value problem, i.e. the pressure-saturation equations:

$$\begin{aligned} -\nabla \cdot (\lambda(s)\kappa\nabla p) &= q \quad \text{in } \Omega, \\ \phi\frac{\partial s}{\partial t} + \nabla \cdot (f_w(s)u) &= \frac{q_w}{p_w} \quad \text{in } \Omega, \\ u \cdot n &= 0 \quad \text{on } \partial\Omega, \\ s &= 0 \quad \text{at } t = 0. \end{aligned} \tag{4.1}$$

Here, κ is permeability; p is pressure; ϕ is the porosity; u is total velocity; q_w and q_o are volumetric source term for water-phase and oil-phase respectively. By injecting water and producing mixture of water and oil, the source term for the saturation equation is simplified as

$$\frac{q_w}{\rho_w} = \max(q, 0) + f(s)\min(q, 0). \tag{4.2}$$

λ is the total mobility analytically defined as

$$\begin{aligned} \lambda(s) &= \lambda_w(s) + \lambda_o(s), \\ \lambda_w(s) = \frac{(s^*)^2}{\mu_w}, \quad \lambda_o(s) &= \frac{(1-s^*)^2}{\mu_o}, \quad s^* = \frac{s - s_{wc}}{1 - s_{or} - s_{wc}}. \end{aligned} \tag{4.3}$$

Here s_{or} is the irreducible oil saturation and s_{wc} is the connate water saturation.

4.1.2 IMPES

In this section, we briefly present the numerical simulator for IMPES formulation on the aforementioned equations. We first solve the pressure equation in (4.1) by the two-point flux-approximation (TPFA) finite-volume scheme. As the name suggests, this method uses the averages of two points to approximate the interface flux

$$v_{ij} = -\int_{\gamma_{ij}} (\lambda(s)\kappa\nabla p)\cdot n \mathrm{d}\nu, \tag{4.4}$$

where γ_{ij} denotes the edge of two adjacent cells Ω_i and Ω_j. For convenience and simplicity, we use an alternative notation $\Lambda := \lambda(s)\kappa$ and define transmissibilities by:

$$t_{ij} = 2|\gamma_{ij}| \left(\frac{\Delta x_i}{\Lambda_{i,ij}} + \frac{\Delta x_j}{\Lambda_{j,ij}} \right)^{-1}. \tag{4.5}$$

Here, Δx_i and Δx_j denotes the cell dimensions in the x-coordinate direction and $\Lambda_{i,ij} = n_{ij} \cdot \Lambda_i n_{ij}$ with n_{ij} as the unit vector pointing out of Ω_i associated to γ_{ij}. The pressure can therefore be solved by

$$\mathbf{A}p = \mathbf{Q}, \tag{4.6}$$

where the element-wise values of matrix $\mathbf{A} = [a_{ik}]$ and $\mathbf{Q} = [Q_i]$ can be calculated by what follows.

$$a_{ik} = \begin{cases} \sum_j t_{ij} & \text{if } k = i, \\ -t_{ik} & \text{if } k \neq i, \end{cases} \qquad Q_i = \int_{\Omega_i} q \mathrm{d}\Omega, \quad \forall \Omega_i \subset \Omega. \tag{4.7}$$

On the discretization of saturation, the IMPES uses an explicit finite-volume scheme by

$$S^{n+1} = S^n + \delta_x^t (\mathbf{B} f(S^n) + Q^+), \tag{4.8}$$

where $(\delta_x^t)_i = \Delta t / |\Omega_i|$ and $\mathbf{B}$ is a matrix implementing $[f(s)Q^- - \nabla \cdot (f(s)v)]$ on a cell-by-cell basis with v as the interface flux derived from TPFA scheme. The explicit saturation solver is only

stable provided the following CFL condition

$$\Delta t \leq \frac{|\Omega_i|}{v_i^{\text{in}} \max_{0\leq s\leq 1} f'(s)}. \tag{4.9}$$

Here v_i^{in} is the flux flows into cell Ω_i.

4.2 Methodology

The first part of our proposed workflow is to build physics-guided surrogate models to predict pressure or interface flux fields using image-based neural networks. Generally speaking, the features of constructed neural networks are image-based and may include information on

- Fluid states at certain time instant:
  - saturation s,
  - pressure p,
  - flux fields v_x and v_y;
- Permeability fields κ;
- Time dependent information: well rates q, time instant indicator t.

All initial state features are fine-grid images of the state of interest. The feature using well rate information is images reflecting both locations of injection and production wells and well rates. The time instant indicators are added as part of input features to avoid errors caused by recursive prediction. To fit with the image-based neural network, the time instant indicators are transformed as homogeneous images with the value of specific time instants. The output of neural networks are images of the fluid state at certain time instant of interest. The predicted pressure or interface flux is used to simulate saturation information using the explicit numerical solver.

4.2.1 Image-based neural networks

Image-based neural networks have drawn much attention in solving fluid dynamics on account of their capability in capturing spatial features. Compared to traditional fully connected neural

networks, image-based neural networks usually have sparser connectivity, which benefits training efficiency, especially under circumstances with large dimensional data. On the other hand, image-based neural networks are usually similar to a typical DNN in that it still consists of an input layer, some hidden layers and an output layer.

In a typical feed-forward convolutional neural network (CNN), which is a common type of image-based neural network, input images evolve through some hidden layers by convolution operation. Given an image I of dimension (n_H, n_W, n_C) and a filter K of dimension (w,w,n_C), the forward mapping of a convolution operation can be formulated as

$$\mathrm{conv}(I, K)_{x,y} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_W} \sum_{k=1}^{n_C} K_{i,j,k} I_{x+i-1,y+j-1,k}. \tag{4.10}$$

Here, subscripts H, W, C denote the height, width and channel of an image respectively. w denotes the width of a filter. If we consider (4.10) as a particular form of matrix multiplication, a hidden convolutional layer can be analytically written as

$$I_{l+1} = \sigma(W_l \circ I_l + b_l), \tag{4.11}$$

where l denotes the l-th layer, $W_l \circ I_l = \mathrm{conv}(I_l, K_l)$, σ and b_l are the activation function and bias respectively.

In this chapter, we mainly construct surrogate models using two specific types of image-based neural networks as what follows.

4.2.1.1 U-Net

The first type of image-based neural network we choose is the U-net [57], which has achieved great performance in various problems involving image and video recognition, medical image segmentation and regression. U-net is a non-fully-connected convolutional neural network consisting of a contracting path and an expansive path.

The general architecture of the U-nets constructed in this chapter is depicted in Figure 4.1. At

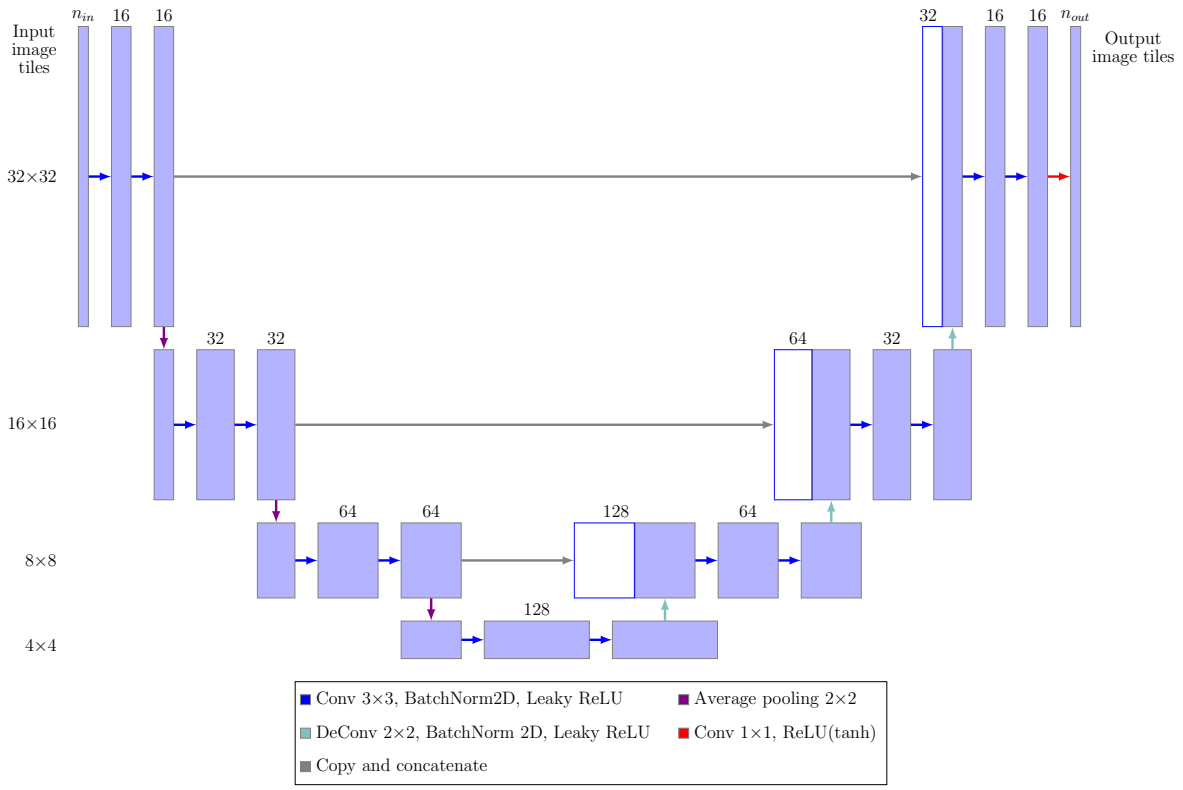


Figure 4.1: An illustration of U-net architecture used in this work. Boxes represents multi-channel features in the forward feeding network. Blank filled boxes represents output features before each average pooling phase in the extracting path. The image sizes of each level are labeled on the left of the graph. The number of filters, which is also the number of output properties of each extracting or expansive step is labeled above each box. n_{in} and n_{out} are the numbers of channels of the input and output images respectively.

each extracting step, features are downsampled through two convolution operations followed by an average pooling layer. Each time after the pooling operation, the height and width of the features are halved while the number of filters or channels are doubled. At the beginning of each expansive path, features first regain height and width through an up-convolution operation. A stack of features of same resolution from the extracting path will be attached to the expanded output, which will be followed by two consecutive convolutional layers. The number of filters or channels are halved with the doubling in height and width. The number of channels will be force to the number of output features by the output layer.

In this architecture, the specific Leaky ReLU activation function is defined in (4.12) with $\alpha =$

0.1.

$$LeakyReLU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0. \end{cases} \tag{4.12}$$

Compared to the original design of U-net, both the numbers of filters and the depth of the extracting/expansive paths are reduced such that less parameters will be needed for training. In this way, the efficiency of network training can be improved.

We also remark that there are some slight differences between the networks when predicting different fluid states of interest. For predicting pressure fields, the number of input channels n_{in} is set to be 4; the output channel number n_{out} is set to be 1 and the last activation function before output is ReLU, which can be considered as a particular case of Leaky ReLU with $\alpha = 0$ in (4.12). On the other end, when the state of interest is the interface fluxes, the image will consist of 2 channels. Thus, n_{in} is set to 5; n_{out} will be 2 and the last activation function will be the hyperbolic tangent function (simplified as Tanh) defined in (4.13), to best match positive and negative signs of the output.

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{4.13}$$

The training and testing of all the U-nets are undertaken by the Python deep learning API Keras.

4.2.1.2 Fourier Neural Operator

A second type of image-based neural network of interest is Fourier neural operator (FNO) [72]. This architecture formulates a new neural operator in Fourier space to operate directly on the images of input features and has good capacity in approximating PDEs with high non-linearity.

Given an input feature $\mathcal{X}$ as a stack of images. FNO first projects the $\mathcal{X}$ into a desired higher dimensional space $\mathcal{X}_0$ by a linear transform layer ("Fully Connected" layer). What follows are named as Fourier layers containing integral operators defined in Fourier space. Mathematically, for each layer *l*, we have

$$\mathcal{X}_l = \sigma(\mathcal{W}(\mathcal{X}_{l-1}) + \mathcal{K}(\mathcal{X}_{l-1})). \tag{4.14}$$

In the above formulation, σ is the nonlinear activation function, $\mathcal{W}$ is a local linear transform

operator and $\mathcal{K}$ is the neural operator defined in Fourier space as

$$\mathcal{K}(\mathcal{X}_{l-1}) = \mathcal{F}^{-1}(\mathcal{R}\mathcal{F}(\mathcal{X}_{l-1})). \tag{4.15}$$

In this operation, $\mathcal{K}$ first apply Fourier transform $\mathcal{F}$ to $\mathcal{X}_{l-1}$. A linear transform $\mathcal{R}$ is then applied to filter out higher Fourier modes. As a final step, an inverse Fourier transform $\mathcal{F}^{-1}$ is applied. The output of the last Fourier layer will eventually be transformed to the desired output by some fully connected layers.

An illustration of the basic architecture used in this chapter is shown in Figure 4.2. As demon-

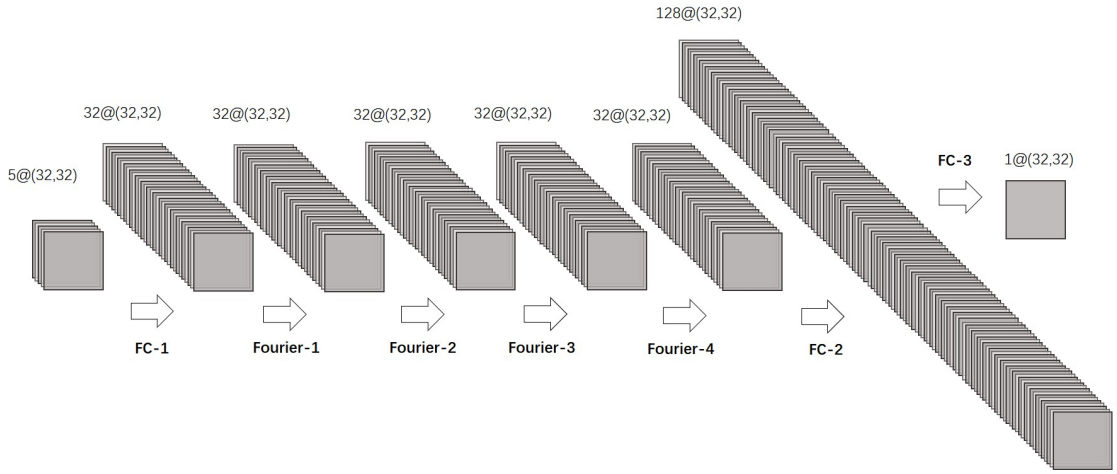


Figure 4.2: Figure of the Fourier neural operator. The network consists of 3 fully connected layers and 4 Fourier layers. The feature information is labeled on the top of each layer in the format of $n_C@(n_H, n_W)$.

strated, we use 5 features as input and predict 1 fluid state of interest, which is the pressure field. No activation functions are used after FC-1 or FC-3 in Figure 4.2. Between Fourier layers, Leaky ReLU (4.12) is used with $\alpha = 0.01$. ReLU is taken as the activation function after FC-3. In this whole procedure, the height and width of features stay unchanged. All training and testing of FNO are run using the Python library Pytorch [73].

4.2.2 Loss function

First of all, we recap two common types of loss functions which are commonly used in neural network training.

Consider a neural network $\mathcal{N}$ modeling the evolution of pressure. The ground truth of certain case is denoted as p^i and the predicted label is denoted as p^i_{pred}. The first common type is the Mean Squared Error (MSE) loss $\mathcal{L}_{\text{mse}}$. Mathemtically, $(\mathcal{L}_{mse})_i$ is defined as

$$(\mathcal{L}_{mse})_i = \|p^i_{\text{pred}} - p^i_{\text{ref}}\|^2_{L^2}. \tag{4.16}$$

Therefore, given M training data samples, $\mathcal{L}_{mse}$ is defined as the mean of the all errors for each data point as follows.

$$\mathcal{L}_{mse} = \frac{1}{M}\sum_{i=1}^{M}(\mathcal{L}_{mse})_i. \tag{4.17}$$

A second commonly used loss function is the Root Mean Squared Error (RMSE), which is similar in performance to MSE. The advantage of RMSE lies in that it is directly interpretable in terms of the distance under the input measurement and thus can be a better measure as to the goodness of model fitting. The definition of RMSE is clearly suggested by its name. Mathematically, one can write

$$\mathcal{L}_{\text{rmse}} = \sqrt{\mathcal{L}_{\text{mse}}}. \tag{4.18}$$

In this chapter, we will utilize both MSE and RMSE in model training, depending on the network structure chosen.

Apart from the basic loss functions listed above, which can provide primary value match between ground truth and prediction, we introduce a physics-guided loss function to improve the training efficiency and to maintain the physical features of the output fluid states. While we can rely on the advantage of image-based neural networks in maintaining spatial and temporal pattern, the combination of neural networks and explicit numerical saturation solver requires good consistency of predicted fluid states so that smoother saturation solutions can be acquired. The priority

gives to the target on the maintaining of cell-wise mass balance property:

$$v_i^{\text{in}} = \max(q_i, 0) - \sum_j \min(v_{ij}, 0) = -min(q_i, 0) + \sum_j \max(v_{ij}, 0) = v_i^{\text{out}}, \tag{4.19}$$

To do so, in a U-net designed to predict pressure fields, we add penalty terms to promote the match of the gradient between p^i_{pred} and p^i_{ref}. This will help the consistency in both flow direction and velocity and benefit the simulation of saturation. Thus, we continue on introducing second parts into the loss function $\mathcal{L}_{\text{flux}}$. Recall from equations (4.4) and (4.5), one calculate both "predicted" and reference interface flux based on saturation, pressure and permeability. The flux loss $(\mathcal{L}_{\text{flux}})_i$ for the ith data sample and the overall flux loss for all training samples defined as

$$\begin{aligned} (\mathcal{L}_{\text{flux}})_i &= \|F_{\text{flux}}(\nabla p^i_{\text{pred}}, s^i_{\text{in}}, \kappa) - F_{\text{flux}}(\nabla p^i_{\text{ref}}, s^i_{\text{in}}, \kappa)\|^2_{L^2}, \\ \mathcal{L}_{\text{flux}} &= \frac{1}{M}\sum_{i=1}^{M} (\mathcal{L}_{\text{flux}})_i. \end{aligned} \tag{4.20}$$

Here, s_{in^i} is the saturation at the same time instant with the desired pressure field. We remark that this saturation information is only used in constructing the flux loss to regularize the training procedure and will not participate in the prediction of pressure fields. In all, the first type of physics-constraint loss function is defined as

$$\mathcal{L}_1(\eta) = \mathcal{L}_{\text{mse}} + \gamma_1 \mathcal{L}_{\text{flux}}, \tag{4.21}$$

where η is the set of all trainable parameters in the neural network, γ_1 is the weight parameter.

In FNO, we consider a different approach. To begin with, instead of MSE-type error, we use RMSE-type error so that hyperparameter tuning can be more efficient. Next, we design loss terms to promote the match of both the gradient and the second partial derivatives between p_{pred} and p_{ref}. This will stimulate the consistency in both flow direction and velocity and benefit the simulation of saturation. Thus, we introduce loss functions $\mathcal{L}_{\text{grad}}$ and $\mathcal{L}_H$. For the i-th data sample, $(\mathcal{L}_{\text{grad}})_i$

and $(\mathcal{L}_H)_i$ are defined by

$$\begin{aligned}(\mathcal{L}_{\text{grad}})_i &= \left\|\nabla p^i_{\text{pred}} - \nabla p^i_{\text{ref}}\right\|^2_{L^2}, \\ (\mathcal{L}_H)_i &= \left\|\Delta p^i_{\text{pred}} - \Delta p^i_{\text{ref}}\right\|^2_{L^2}.\end{aligned} \tag{4.22}$$

Similar to MSE, we define $\mathcal{L}_{\text{grad}}$ and $\mathcal{L}_H$ by

$$\begin{aligned}\mathcal{L}_{grad} &= \frac{1}{M}\sum_{i=1}^{M}(\mathcal{L}_{grad})_i, \\ \mathcal{L}_H &= \frac{1}{M}\sum_{i=1}^{M}(\mathcal{L}_H)_i.\end{aligned} \tag{4.23}$$

The loss function is hereby defined as follows,

$$\mathcal{L}_2(\eta) = \mathcal{L}_{\text{rmse}} + \gamma_{2,1}\sqrt{\mathcal{L}_{\text{grad}}} + \gamma_{2,2}\sqrt{\mathcal{L}_H}. \tag{4.24}$$

Here, η is the still set of all trainable parameters in the neural network; $\gamma_{2,1}$ and $\gamma_{2,2}$ are weight parameters.

Same as Chapter 3, the training of all proposed method is to find the parameter η^* such that for all training samples, and for the particular type of loss function $\mathcal{L}_i$ we choose, we have

$$\eta^* = \underset{\eta}{\operatorname{argmin}}\, \mathcal{L}_i. \tag{4.25}$$

4.3 Numerical results

In this section, we present two experiments utilizing different types of image-based neural networks discussed in Section 4.2. We use simulation results from IMPES simulator as a reference solution.

In all experiments, we take the spatial domain as $\Omega = (0,1)^2$, with a fine mesh size of $h = 1/32$. We run fine-scale numerical simulation using 8 different realizations of permeability fields. Illustrations of all permeability fields used is depicted in Figure 4.3.

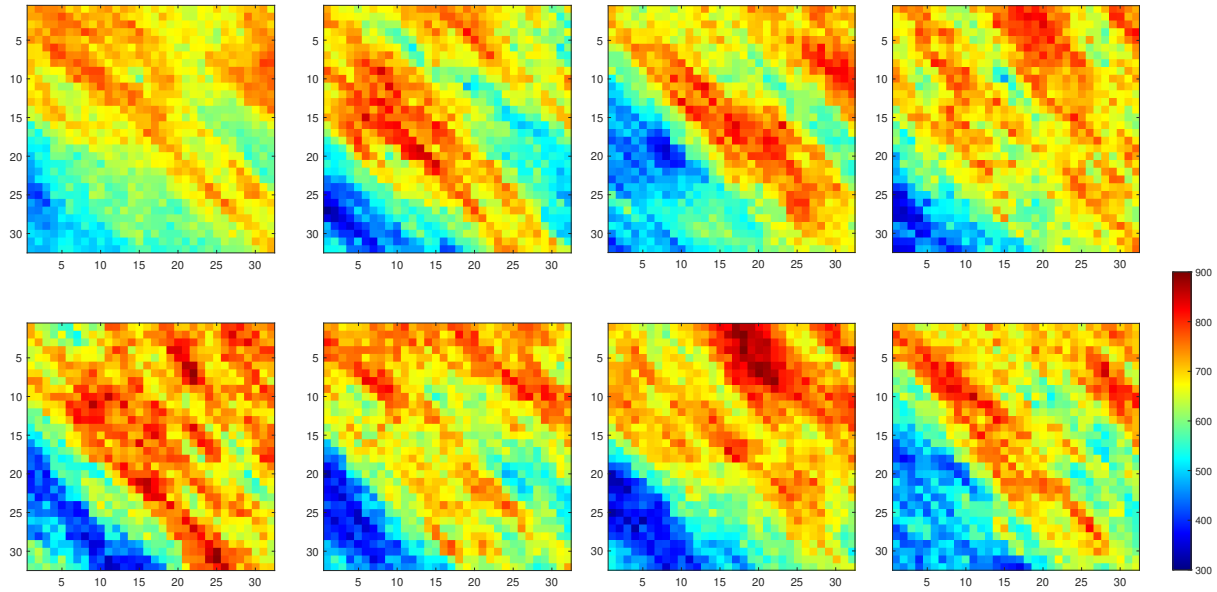


Figure 4.3: Realizations of permeability fields used in all experiments.

As for well rate, we use a five-spot setting with four injectors on four corners and one producer in the center. We randomly generate 4 different dimensionless pore volume well rate for the four injectors and the well rate for the producer in the center will be the sum of the injection rates. Overall, 5 different settings of well rates are generated. An example of well rate is given in Figure 4.4

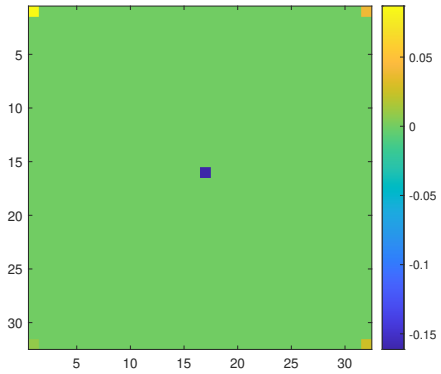


Figure 4.4: An example of pore volume rate randomly generated for the two experiments.

For each source term and permeability field, we run IMPES simulation on the reservoir for 760 time steps and collected images of states including pressure, interface flux and saturation every 19 steps. Every realization of this simulation process is unique in terms of well rates combined with permeability field. For each realization, we will group the 40 data points into 4 groups of 10 data points of consecutive time steps. The last fluid states of the previous group or the initial condition of the corresponding realization will be considered as initial fluid states in the input feature. In all, we have 1600 data samples available, among which we take 5% as testing cases and the rest as training cases. This gives 1520 training samples and 80 testing samples. We remark that the testing samples are complete realizations from particular permeability and well rate combined settings. This way, we are able to observe the performance of our workflow on both one step and recursive simulation for saturation.

To evaluate the effectiveness of our workflow, we will compute errors of both the predicted state of interest and recursively simulated saturation. Taking the prediction of pressure as an example, given initial fluid states p^0, s^0 and κ, for each time instant n, $n = 1, 2, \ldots, 10$, we can predict the pressures by

$$p^n_{\text{pred}} = \mathcal{N}(p^0, s^0, \kappa, q^n, t^n; \eta^*). \tag{4.26}$$

Denoting the numerical saturation solver as $\mathcal{U}$, we can solve for s^n_{pred} by the following scheme

$$s^n_{\text{pred}} = \mathcal{U}(\ldots \mathcal{U}(s^0, p^1_{\text{pred}}, q^1), \ldots, p^n_{\text{pred}}, q^n). \tag{4.27}$$

Therefore, we can compute the errors by

$$e^n_p = \frac{\left\| p^n_{\text{ref}} - p^n_{\text{pred}} \right\|_{L^2(\Omega)}}{\left\| p^n_{\text{ref}} \right\|_{L^2(\Omega)}}, \quad e^n_s = \frac{\left\| s^n_{\text{ref}} - s^n_{\text{pred}} \right\|_{L^2(\Omega)}}{\left\| s^n_{\text{ref}} \right\|_{L^2(\Omega)}}. \tag{4.28}$$

In what follows, we will present two experiments on two set of designs on the hybrid workflow for predicting and simulating the dynamics of two-phase flow. The performances of the neural network architectures and overall workflow will be tested. The efficiency of each workflow and

cross comparison will be briefly discussed in section 4.3.3.

4.3.1 Experiment 1

In this experiment, we evaluate the performance of proposed hybrid workflow based on U-net. Two neural network architectures are constructed in this experiment.

1. A network that predicts pressure field at time instant n, $\mathcal{N}_p^{\text{UNet}}$.

$$p^n_{\text{pred}} = \mathcal{N}_p^{\text{UNet}}(p^0, \kappa, q^n, t^n; \eta_p^*), \tag{4.29}$$

2. A network that predicts the change of interface velocity fields at time instant n, $\mathcal{N}_v^{UNet}$.

$$[v_{x}{}^n_{\text{pred}}, v_{y}{}^n_{\text{pred}}] = [v_x^0, v_y^0] + \mathcal{N}_v^{\text{UNet}}(v_x^0, v_y^0, \kappa, q^n, t^n; \eta_v^*). \tag{4.30}$$

During the training of $\mathcal{N}_p^{\text{UNet}}$, $\mathcal{L}_1$ defined in (4.21) is used as loss function while for the training of $\mathcal{N}_v^{\text{UNet}}$, we utilize a variation of MSE loss. Unless otherwise specified, all the networks in this experiment are trained using a training scheme of 20 batch size and 200 epochs, with an early stopping criterion of 20 epochs.

We first present the performance of pressure prediction using U-Net. For this experiment, all images are scaled to the range of $[0, 1]$. The plots of errors of different realizations in test cases are depicted in Figure 4.5. As suggested in Figure 4.5, both loss function schemes can perform evenly matched on predicting the pressure field. It can be observed that the prediction of model trained by plain MSE performs slightly better on the testing cases. Situation is overturned when we proceed to the second phase of numerically simulating saturation. As depicted in Figure 4.6, the saturation results based on plain MSE model performs far more worse than the ones with $\mathcal{L}_1$.

However, neither network can yield satisfactory prediction enough for the simulation of saturation. This pushes towards the construction of a second neural network $\mathcal{N}_v^{\text{UNet}}$. As described in (4.30), we construct a U-net based neural network that predicts the difference between the interface flux at the time instant of interest and the initial state. The data are properly scaled such that

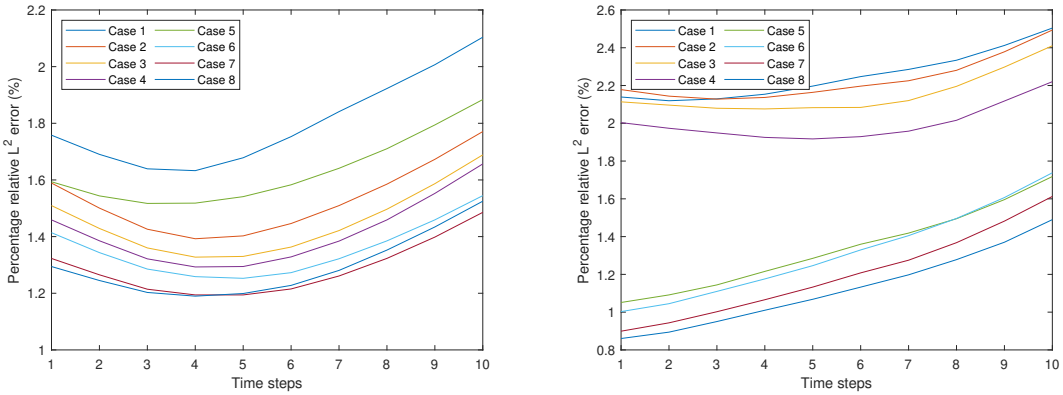


Figure 4.5: Plots of percentage relative L^2 errors in predicting pressure fields with U-net using different loss function schemes. Left: $\mathcal{L}_{\text{mse}}$, average of e_p^n is 1.47%, average of e_p^{10} is 1.71%. Right: $\mathcal{L}_1$, average ofe_p^n is 1.70%, average of e_p^{10} is 2.02%.

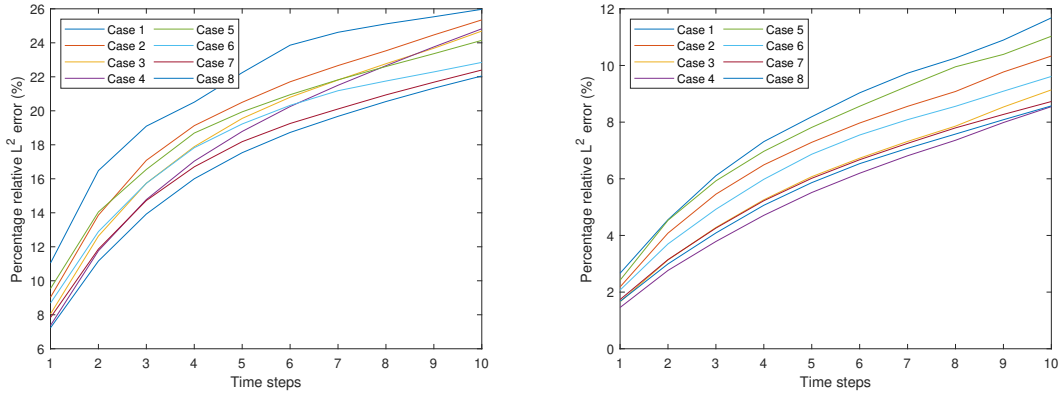


Figure 4.6: Plots of percentage relative L^2 errors in recursively simulating saturation fields with U-net-predicted pressure using different loss function schemes. Left: $\mathcal{L}_{\text{mse}}$, average of e_p^n is 18.73%, average of e_p^{10} is 24.03%. Right: $\mathcal{L}_1$, average of e_p^n is 6.58%, average of e_p^{10} is 9.71%.

the maximum and minimum of the flux difference lie in the range of $[-1, 1]$. We also make sure the interface flux matrix are properly padded to be equal in height and width before putting into the U-net as an input feature. Considering the fact that interface flux is of the same order as the gradient of pressure, we only use plain MSE loss function to monitor the training of the neural network. The errors of the predicted interface flux and simulated saturation are demonstrated in Figure 4.7. An example of comparison on final time fluid states is depicted in Figure 4.8.

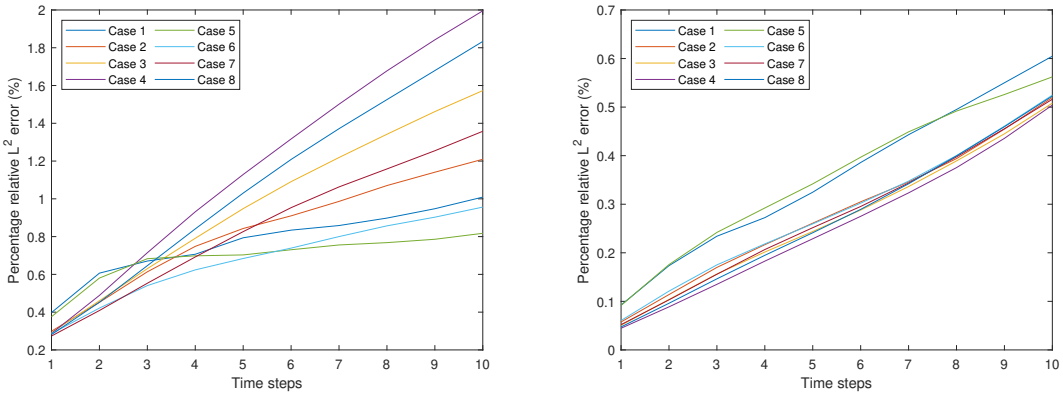


Figure 4.7: Plots of percentage relative L^2 errors of predicted interface flux and recursively simulated saturation fields with in experiment 1. Left: interface flux prediction errors, average error for all v^n_{pred} is 0.89%; for v^{10}_{pred} is 1.34%. Right: saturation simulation errors, average error for all cases is 0.30%; for time step 10 is 0.53%.

To summarize, in this experiment, we designed two U-net based neural network $\mathcal{N}_p^{\text{UNet}}$ and $\mathcal{N}_v^{\text{UNet}}$ to predict pressure and interface flux respectively. Thanks to the features of U-net and the introduce of physics-related loss function, both networks can provide approximation on the targeted output feature with good accuracy and certain dependency on time. On the other hand, as an intermediate surrogate model serving for the simulation of saturation, $\mathcal{N}_v^{\text{UNet}}$ achieves better performance in dynamics consistency.

4.3.2 Experiment 2

In this experiment, we construct a second workflow based on FNO, $\mathcal{N}_p^{\text{FNO}}$. Mathematically, for a pressure field at time instant n of interest, we have

$$p^n_{\text{pred}} = \mathcal{N}_p^{\text{FNO}}(p^0, s^0, \kappa, q^n, t^n; \eta^*) \tag{4.31}$$

We take the loss function $\mathcal{L}_2$ defined in (4.24) for model training. All the images are scaled to a range of $[1, 10]$. All the neural networks are trained with a scheme of batch size 20 and 100 epochs. Max stagnant epochs to stop training is 20. To evaluate the effectiveness of the physics-based loss function designed, we train the neural networks with different setting of weight parameters

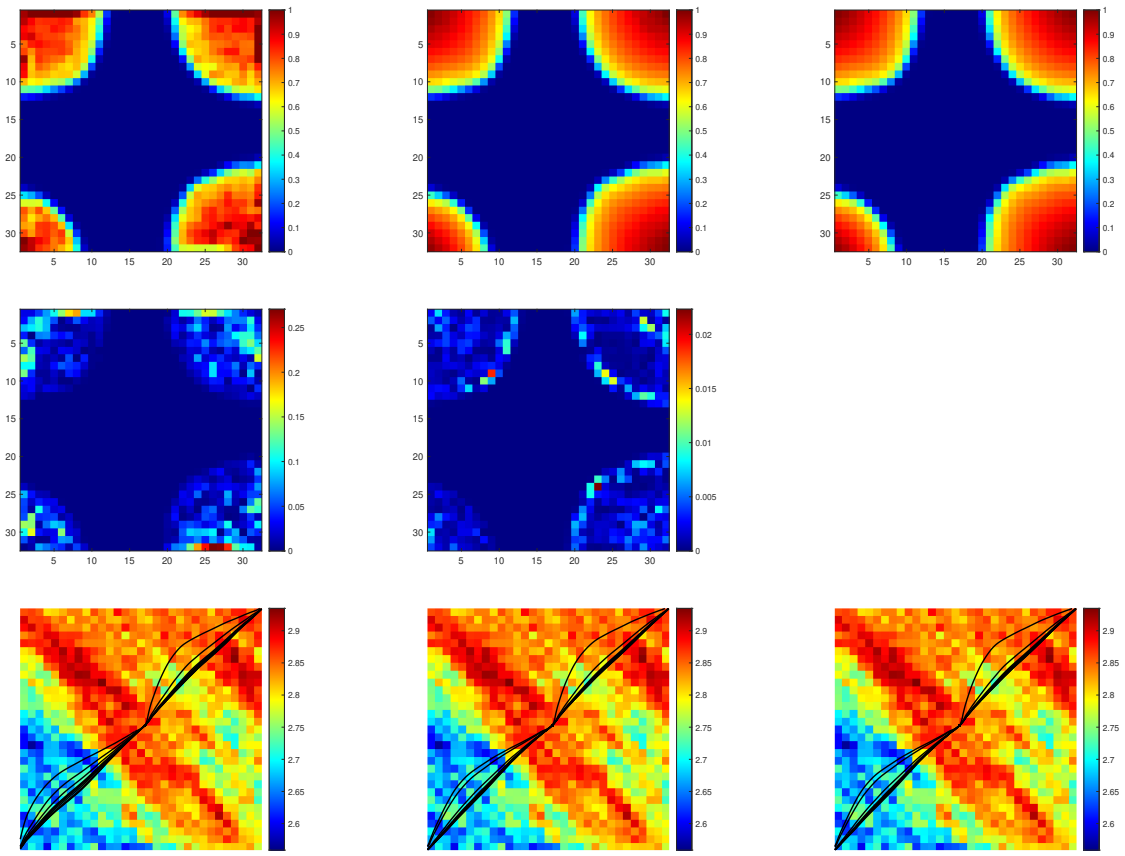


Figure 4.8: Figures of final time saturation and streamline for test realization case 8 in Experiment 1. Top row: saturation. Second row: error of saturation. Bottom row: streamline. Left: U-net pressure model with $\mathcal{L}_1$. Middle: interface flux model. Right: fine-scale reference.

$(\gamma_{2,1}, \gamma_{2,2})$ in (4.24). The performance of different weight settings is summarized in Table 4.1. As can be observed from the table, although the reason of the blow up for the second setting is to be discovered, penalizing loss function by both $\mathcal{L}_{\text{grad}}$ and $\mathcal{L}_H$ undoubtedly improves both efficiency in training and accuracy in predicting of the neural network structure.

We also briefly compare the impact of one of the most important content in FNO, i.e. the number of filters in each Fourier layer, under the same $\mathcal{L}_2$ loss function. The results is summarized in Table 4.2.

Similar to the situation in Experiment 1, we can observe a slight trade-off between the average prediction errors for pressure and average simulation errors for saturation. Figure 4.9 gives an

$(\gamma_{2,1}, \gamma_{2,2})$	e_p^n	e_p^{10}	e_s^n	e_s^{10}
$(0, 0)$	0.26%	0.32%	4.69%	6.76%
$(2, 0)$	blow up			
$(0, 4)$	0.26%	0.32%	4.69%	6.76%
$(2, 4)$	**0.18%**	**0.19%**	**2.09%**	**3.02%**

Table 4.1: Average percentage prediction and simulation errors for all test cases and final time cases given different setting of loss weight parameter in Experiment 2.

Filter #	e_p^n	e_p^{10}	e_s^n	e_s^{10}
12	0.18%	**0.19%**	2.09%	3.02%
16	0.18%	**0.19%**	2.08%	2.93%
20	0.18%	0.21%	**2.05%**	**2.87%**

Table 4.2: Average percentage prediction and simulation errors for all test cases and final time cases given different number of filters in Experiment 2.

illustration on the extent of accuracy on FNOs of 12 and 20 filters inside each Fourier layer.

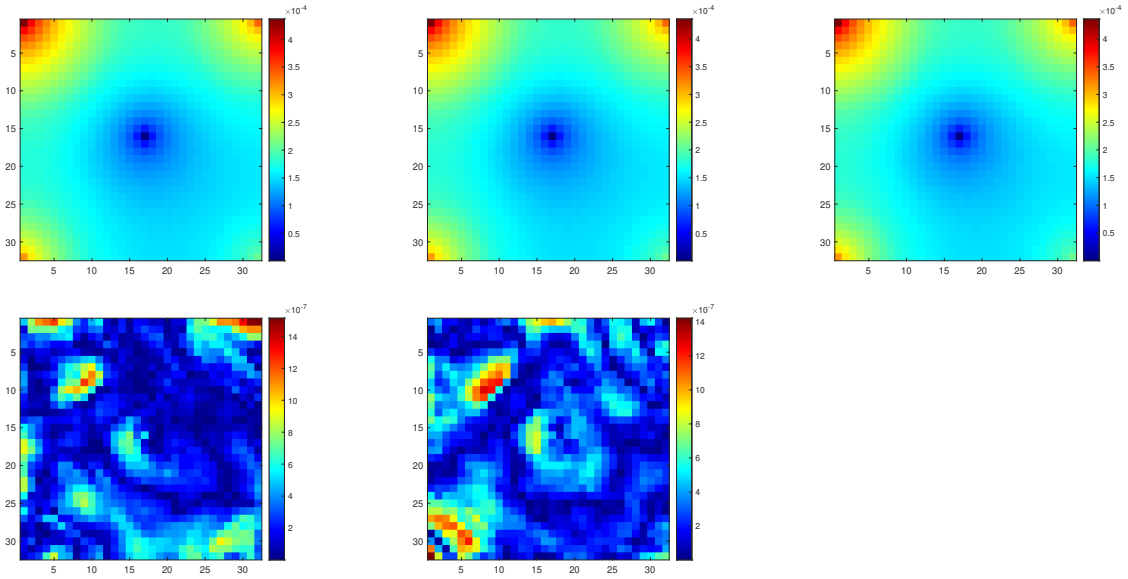


Figure 4.9: Figures of pressure and errors of p_{pred}^{10} in the test realization 4 in Experiment 2. Top: pressure. Bottom: absolute difference of pressure compared to reference. First column: FNO with 12 filters. Second column: FNO with 20 filters. Right: fine-scale pressure p_{ref}^{10}.

The temporal stability trend of different number of filters is depicted in Figure 4.10. As shown in the Figure, although the FNO using 12 filters in the Fourier layers gains better stability in time, the FNO with 20 filters in each Fourier layer tends to be more stable in later phase of saturation simulation. On the other end, in general, all the FNO trained can yield an acceptable output for the simulation. The only fly in the ointment is that even by introducing second order derivatives into loss function, a neural network can still have "peak" errors at particular areas as shown in Figure 4.9, which will likely result in some discontinuity on saturation fields on near well locations as shown in Figure 4.11. Nevertheless, the predicted pressure fields, even ones predicted by FNO consisting of only 12 filters can maintain clear water front when simulating for saturation.

4.3.3 Discussion

In the above two experiments, the usefulness of U-net and FNO in our proposed workflow has been validated. Both workflow can meet primary requirement on the prediction of intermediate fluid states such as pressure or flux velocity. By specifying physics-guided loss function, the second hybrid methodology utilizing FNO can serve as a surrogate model in the IMPES approach for simulation of saturation.

In this section, we give a brief discussion on the complexity and efficiency of different workflows developed compared with the benchmark simulator used (IMPES). For all neural networks constructed in this chapter, the training and testing process are undertaken by the same CPU (Intel Core i7-5700HQ CPU @ 2.7GHz). In Table 4.3, we list related CPU costs regarding different modules in the proposed workflows. For the convenience of comparison, we also summarize the predicting and successive simulation metrics in Table 4.4.

Even though $\mathcal{N}_p^{\text{FNO}}$ is the neural network with the largest number of parameters, it has superior performance in both training efficiency over $\mathcal{N}_p^{\text{UNet}}$ and $\mathcal{N}_v^{\text{UNet}}$. For mild fidelity of the two-phase flow dynamics, the workflow combining FNO and explicit solver will be an attractive choice, balancing accuracy and efficiency.

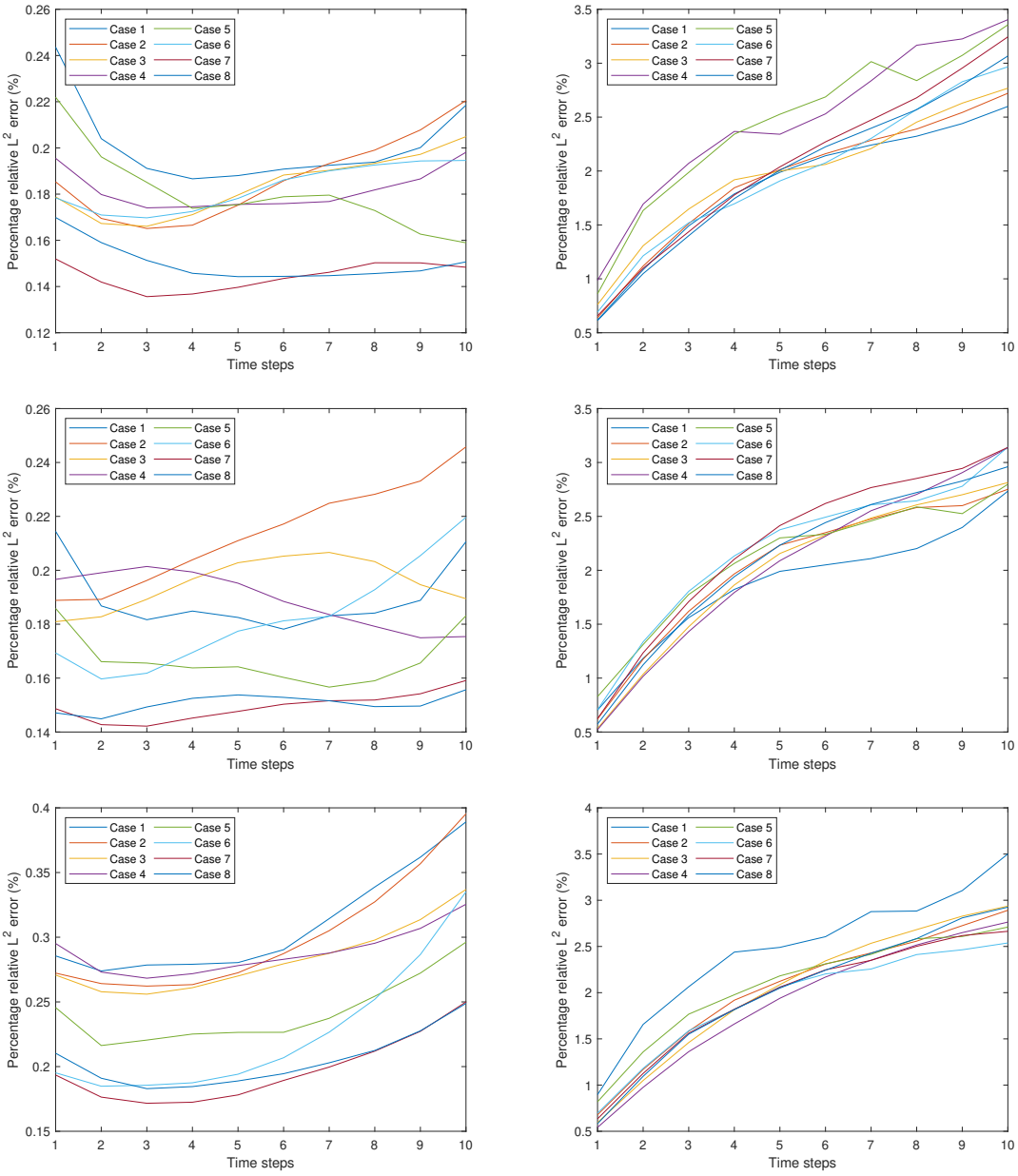


Figure 4.10: Figures of temporal FNO prediction and simulation error for different test realizations in Experiment 2. Left: pressure prediction error. Right: saturation simulation error. First row: 12 filters. Second row: 16 filters. Third row:20 filters.

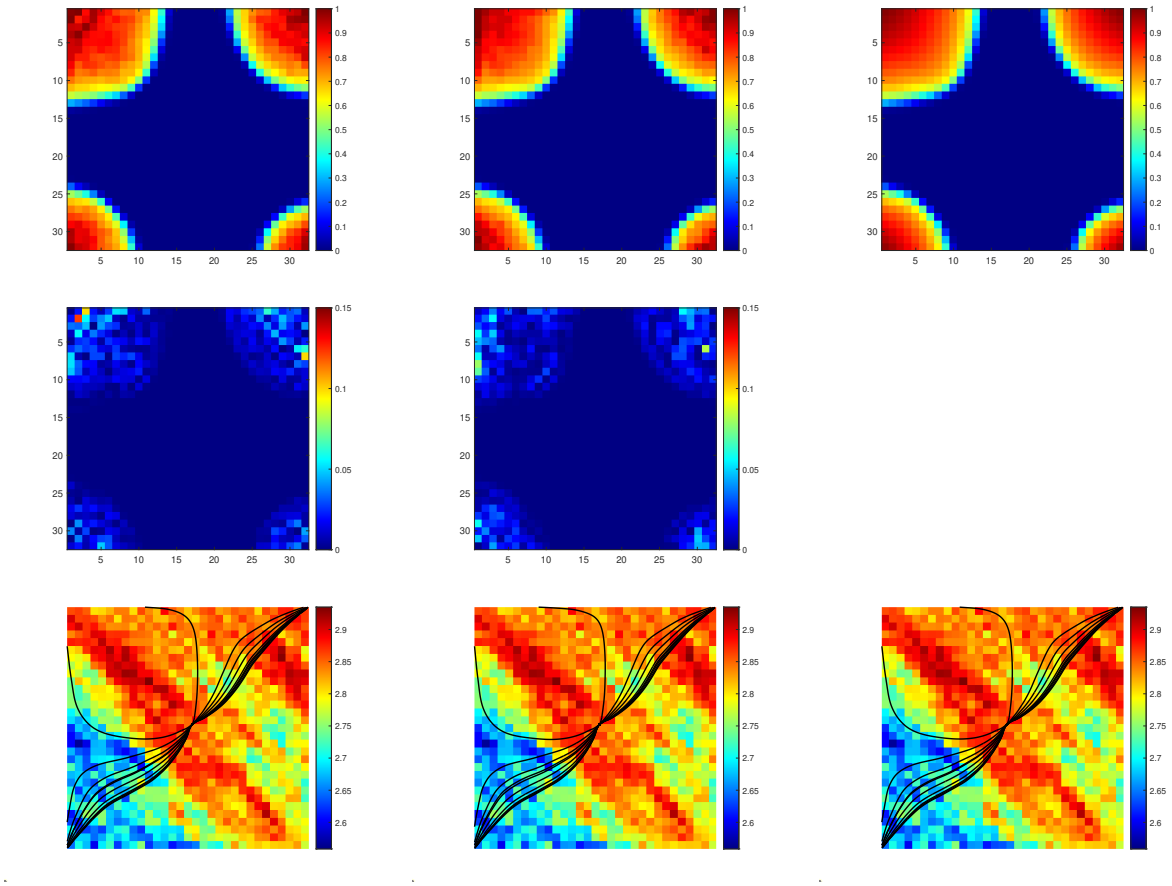


Figure 4.11: Figures of final time saturation and streamline for test realization case 4 in Experiment 2. Top row: saturation. Second row: error of saturation. Bottom row: streamline. First column: FNO with 12 filters. Second column: FNO with 20 filters. Third column: fine-scale reference.

Module name	Trainable parameter #	Training time (s)	Prediction time (s)	
			p	v
$\mathcal{N}_p^{\text{UNet}}$	**483,585**	6088.76	**0.26**	—
$\mathcal{N}_v^{\text{UNet}}$	485,154	4622.84	—	0.27
$\mathcal{N}_p^{\text{FNO}}$	1,188,417	**3984.79**	0.78	—

Table 4.3: CPU time cost related to different modules in proposed workflow. Prediction time is based on a scale of 80 test cases. The time costs involving numerical simulator is calculated by an average of 10 simulation on realizations.

Model	Mean prediction errors(%)		Mean simulation errors(%)	
	e_p^n or e_v^n	e_p^{10} or e_v^{10}	e_s^n	e_s^{10}
$\mathcal{N}_p^{\text{UNet}}$	1.47	1.71	6.58	9.71
$\mathcal{N}_v^{\text{UNet}}$	0.89	1.34	**0.30**	**0.53**
$\mathcal{N}_p^{\text{FNO}}$	**0.18**	**0.19**	2.05	2.87

Table 4.4: Summary of the best prediction and simulation metrics of different neural network modules.

5. SUMMARY AND CONCLUSIONS

Hereinafter, we conclude this dissertation with a brief summary.

In Chapter 2, we investigated the non-local multicontinuum (NLMC) upscaling method for a dual continuum model in fractured porous media. Localized multiscale basis functions that separate each continuum are constructed. To find the basis, we solve local problems subject to energy minimization constraints in oversampling coarse regions. It is showed that the basis functions equip the method with coarse mesh convergence. Some numerical examples are presented to support the theory. The numerical examples also indicate that the proposed method provides accurate and efficient coarse-grid approximation.

In Chapter 3, we explore the connections between POD-based model reduction and Deep Learning methodologies for reservoir simulation. In our novel approach, the construction of the POD modes leads to degrees of freedom having physical meanings (e.g., represent the solution values at selected locations). This is an important step as this provides a natural framework for applying deep learning techniques in the context of reservoir simulation. Especially, nodal basis functions make the interpolation between data-rich and data-deficient models possible, which is also tested. Our results show that multi-layer network provides an accurate approximation for the solution of the two-phase flow problem. Moreover, by taking in available observation data combined with computational data, the reduce-order model is modified.

In Chapter 4, we propose hybrid workflow utilizing both deep learning techniques and traditional numerical simulating methodology to improve the efficiency of reservoir simulation for multi-phase flow. The coupled nonlinear system is decomposed into pressure/flux and saturation solvers. The efficiency of explicit saturation simulator is preserved. On constructing a surrogate model for predicting pressure/flux, the advantage of image-based neural networks in maintaining spatial and temporal patterns is utilized. The output of the neural networks are fed into an explicit numerical solvers for saturation simulation. Physics-constraint is introduced as loss penalty to impose the match of network output with embedded physics in fluid dynamics. Our results show that,

by training image-based neural networks with physics-based loss functions, the evolution of fluid dynamics can be predicted accurately and with certain temporal stability maintained. The results also suggest that the proposed hybrid deep learning numerical workflow is capable of providing accurate approximation for solutions in nonlinear multi-phase flow problems with considerable efficiency.

REFERENCES

[1] G. Voneiff, S. Sadeghi, P. Bastian, B. Wolters, J. Jochen, B. Chow, K. Chow, M. Gatens, *et al.*, "Probabilistic forecasting of horizontal well performance in unconventional reservoirs using publicly-available completion data," in *SPE Unconventional Resources Conference*, Society of Petroleum Engineers, 2014.

[2] M. Karimi-Fard and A. Firoozabadi, "Numerical simulation of water injection in 2d fractured media using discrete-fracture model," *SPE-71615-MS*, 2001.

[3] M. Karimi-Fard, L. Durlofsky, and K. Aziz, "An efficient discrete-fracture model applicable for general-purpose reservoir simulators," *SPE-88812-PA*, 2004.

[4] T. Garipov, M. Karimi-Fard, and H. Tchelepi, "Discrete fracture model for coupled flow and geomechanics," *Computational Geosciences*, vol. 20, no. 1, pp. 149–160, 2016.

[5] E. Gildin, T. Lopez, *et al.*, "Closed-loop reservoir management: Do we need complex models?," in *SPE Digital Energy Conference and Exhibition*, Society of Petroleum Engineers, 2011.

[6] Z. M. Alghareeb and J. Williams, "Optimum decision-making in reservoir management using reduced-order models," *SPE*, 2013.

[7] L. Durlofsky, "Numerical calculation of equivalent grid block permeability tensors for heterogeneous porous media," *Water Resour. Res.*, vol. 27, pp. 699–708, 1991.

[8] X. Wu, Y. Efendiev, and T. Hou, "Analysis of upscaling absolute permeability," *Discrete and Continuous Dynamical Systems, Series B.*, vol. 2, pp. 158–204, 2002.

[9] 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.

[10] G. Barenblatt, I. P. Zheltov, and I. Kochina, "Basic concepts in the theory of seepage of homogeneous liquids in fissured rocks [strata]," *Journal of applied mathematics and mechanics*, vol. 24, no. 5, pp. 1286–1303, 1960.

[11] H. Kazemi, L. Merrill Jr, K. Porterfield, P. Zeman, *et al.*, "Numerical simulation of water-oil flow in naturally fractured reservoirs," *Society of Petroleum Engineers Journal*, vol. 16, no. 06, pp. 317–326, 1976.

[12] K. Pruess and T. Narasimhan, "On fluid reserves and the production of superheated steam from fractured, vapor-dominated geothermal reservoirs," *Journal of Geophysical Research: Solid Earth*, vol. 87, no. B11, pp. 9329–9339, 1982.

[13] J. Warren, P. J. Root, *et al.*, "The behavior of naturally fractured reservoirs," *Society of Petroleum Engineers Journal*, vol. 3, no. 03, pp. 245–255, 1963.

[14] Y.-S. Wu, K. Pruess, *et al.*, "A multiple-porosity method for simulation of naturally fractured petroleum reservoirs," *SPE Reservoir Engineering*, vol. 3, no. 01, pp. 327–336, 1988.

[15] W. E and B. Engquist, "Heterogeneous multiscale methods," *Comm. Math. Sci.*, vol. 1, no. 1, pp. 87–132, 2003.

[16] A. Abdulle, "On a priori error analysis of fully discrete heterogeneous multiscale fem," *SIAM J. Multiscale Modeling and Simulation*, vol. 4, no. 2, pp. 447–459, 2005.

[17] W. E, P. Ming, and P. Zhang, "Analysis of the heterogeneous multiscale method for elliptic homogenization problems," *J. Amer. Math. Soc.*, vol. 18, no. 1, pp. 121–156, 2005.

[18] T. Hughes, G. Feijóo, L. Mazzei, and J.-B. Quincy, "The variational multiscale method - a paradigm for computational mechanics," *Comput. Methods Appl. Mech Engrg.*, vol. 127, pp. 3–24, 1998.

[19] T. Hughes and G. Sangalli, "Variational multiscale analysis: the fine-scale Green's function, projection, optimization, localization, and stabilized methods," *SIAM Journal on Numerical Analysis*, vol. 45, no. 2, pp. 539–557, 2007.

[20] O. Iliev, R. Lazarov, and J. Willems, "Variational multiscale finite element method for flows in highly porous media," *Multiscale Model. Simul.*, vol. 9, no. 4, pp. 1350–1372, 2011.

[21] V. Calo, Y. Efendiev, and J. Galvis, "A note on variational multiscale methods for high-contrast heterogeneous porous media flows with rough source terms," *Advances in water resources*, vol. 34, no. 9, pp. 1177–1185, 2011.

[22] 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.

[23] Y. Efendiev, T. Hou, and X. Wu, "Convergence of a nonconforming multiscale finite element method," *SIAM J. Numer. Anal.*, vol. 37, pp. 888–910, 2000.

[24] 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.

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

[26] E. Chung, Y. Efendiev, and G. Li, "An adaptive GMsFEM for high-contrast flow problems," *Journal of Computational Physics*, vol. 273, pp. 54–76, 2014.

[27] E. Chung, Y. Efendiev, and T. Y. Hou, "Adaptive multiscale model reduction with generalized multiscale finite element methods," *Journal of Computational Physics*, vol. 320, pp. 69–95, 2016.

[28] Y. Efendiev, E. Gildin, and Y. Yang, "Online adaptive local-global model reduction for flows in heterogeneous porous media," *Computation*, vol. 4, no. 2, p. 22, 2016.

[29] E. T. Chung, Y. Efendiev, T. Leung, and M. Vasilyeva, "Coupling of multiscale and multi-continuum approaches," *GEM-International Journal on Geomathematics*, vol. 8, no. 1, pp. 9–41, 2017.

[30] M. Wang, S. W. Cheung, E. T. Chung, M. Vasilyeva, and Y. Wang, "Generalized multiscale multicontinuum model for fractured vuggy carbonate reservoirs," *Journal of Computational and Applied Mathematics*, vol. 366, p. 112370, 2020.

[31] S. W. Cheung, E. T. Chung, Y. Efendiev, W. T. Leung, and M. Vasilyeva, "Constraint energy minimizing generalized multiscale finite element method for dual continuum model," *Communications in Mathematical Sciences*, vol. 18, pp. 663–685, 2020.

[32] J. S. R. Park, S. W. Cheung, T. Mai, and V. H. Hoang, "Multiscale simulations for upscaled multi-continuum flows," *Journal of Computational & Applied Mathematics*, vol. 374, p. 112782, 2020.

[33] J. S. R. Park, S. W. Cheung, and T. Mai, "Multiscale simulations for multi-continuum richards equations," *arXiv preprint, arXiv:2010.09181*, 2020.

[34] E. T. Chung, Y. Efendiev, and W. T. Leung, "Constraint energy minimizing generalized multiscale finite element method," *Computer Methods in Applied Mechanics and Engineering*, vol. 339, pp. 298–319, 2018.

[35] E. Chung, Y. Efendiev, and W. T. Leung, "Constraint energy minimizing generalized multiscale finite element method in the mixed formulation," *Computational Geosciences*, vol. 22, no. 3, pp. 677–693, 2018.

[36] S. W. Cheung, E. T. Chung, and W. T. Leung, "Constraint energy minimizing generalized multiscale discontinuous galerkin method," *Journal of Computational & Applied Mathematics*, vol. 380, p. 112960, 2020.

[37] S. W. Cheung, E. T. Chung, Y. Efendiev, and W. T. Leung, "Explicit and energy-conserving constraint energy minimizing generalized multiscale discontinuous galerkin method for wave propagation in heterogeneous media," *arXiv preprint, arXiv:2009.00991*, 2020.

[38] C. Xiao, O. Leeuwenburgh, and H. e. a. Lin, "Non-intrusive subdomain pod-tpwl for reservoir history matching," *Computational Geosciences*, 2018.

[39] J. Jansen and L. Durlofsky, "Use of reduced-order models in well control optimization," *J. Optim Eng (2017) 18: 105.*, vol. 18, no. 105, 2017.

[40] L. J. D. Rui Jiang, "Implementation and detailed assessment of a gnat reduced-order model for subsurface flow simulation,," *Journal of Computational Physics*, vol. 379, pp. 192–213, 2019.

[41] X. Tan, E. Gildin, and H. e. a. Florez, "Trajectory-based deim (tdeim) model reduction applied to reservoir simulation," *Comput Geosci*, 2018.

[42] T. S, C. KT, and D. LJ., "Error modeling for surrogates of dynamical systems using machine learning.," *Int J Numer Meth Engng.*, vol. 112, pp. 1801–1827., 2017.

[43] J. N. Kani and A. Elsheikh, "Reduced-order modeling of subsurface multi-phase flow models using deep residual recurrent neural networks," *Transp Porous Media*, 2018.

[44] M. Ghasemi and E. Gildin, "Model order reduction in porous media flow simulation using quadratic bilinear formulation," *Computational Geosciences*, vol. 20, no. 3, pp. 723–735, 2016.

[45] Trehan, Sumeet, and L. J. Durlofsky, "Trajectory piecewise quadratic reduced-order model for subsurface flow, with application to pde-constrained optimization," *Journal of Computational Physics*, vol. 326, pp. 446–473., 2016.

[46] G. Cybenko, "Approximations by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 183–192, 1989.

[47] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[48] B. C. Csáji, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, p. 48, 2001.

[49] M. Telgarsky, "Benefits of depth in neural networks," *arXiv preprint arXiv:1602.04485*, 2016.

[50] H. Mhaskar, Q. Liao, and T. Poggio, "Learning functions: when is deep better than shallow," *arXiv preprint arXiv:1603.00988*, 2016.

[51] B. Hanin, "Universal function approximation by deep neural nets with bounded width and relu activations," *arXiv preprint arXiv:1708.02691*, 2017.

[52] 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.

[53] Z. Li and Z. Shi, "Deep residual learning and pdes on manifold," *arXiv preprint arXiv:1708.05115*, 2017.

[54] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[55] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[57] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[58] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[59] Z. Zhong, A. Y. Sun, and H. Jeong, "Predicting co2 plume migration in heterogeneous formations using conditional deep convolutional generative adversarial network," *Water Resources Research*, vol. 55, no. 7, pp. 5830–5851, 2019.

[60] S. Mo, Y. Zhu, N. Zabaras, X. Shi, and J. Wu, "Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media," *Water Resources Research*, vol. 55, no. 1, pp. 703–728, 2019.

[61] H. Klie and H. Florez, "Data-driven prediction of unconventional shale-reservoir dynamics," *SPE Journal*, vol. 25, no. 05, pp. 2564–2581, 2020.

[62] 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.

[63] M. Tang, Y. Liu, and L. J. Durlofsky, "A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems," *Journal of Computational Physics*, vol. 413, p. 109456, 2020.

[64] J. Douglas Jr and T. Arbogast, "Dual porosity models for flow in naturally fractured reservoirs," *Dynamics of Fluids in Hierarchical Porous Media*, pp. 177–221, 1990.

[65] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30.

[66] 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, 2011.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[68] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[69] F. Chollet *et al.*, *Deep Learning with Python - 1st Edition*. Manning Publications, 2017.

[70] J. Sheldon, W. Cardwell Jr, *et al.*, "One-dimensional, incompressible, noncapillary, two-phase fluid flow in a porous medium," *Transactions of the AIME*, vol. 216, no. 01, pp. 290–296, 1959.

[71] H. Stone, A. Garder Jr, *et al.*, "Analysis of gas-cap or dissolved-gas drive reservoirs," *Society of Petroleum Engineers Journal*, vol. 1, no. 02, pp. 92–104, 1961.

[72] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Graph kernel network for partial differential equations," 2020.

[73] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.