Multiscale Model Reduction Methods for Deterministic and Stochastic Partial Differential Equations

Thesis by

Maolin Ci

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2014

(Defended March 7th, 2014)

To my parents

Acknowledgments

I would like to express my gratitude to my advisor Professor Yizhao Thomas Hou, who has always provided me with great support and guidance in my work and life. His enthusiasm for knowledge and mathematical rigor is the perfect role model for me during my reach. He has also helped me a lot in overcoming the difficulties in every aspect of my life.

Besides my advisor, I would like to thank the rest of my thesis committee: Professor James Beck, Professor Oscar Bruno, and Professor Houman Owhadi, for their encouragement and insightful comments. I would also like to thank the staff of the ACM department - Sydney Garstang, Carmen Nemer-Sirois, and Sheila Shull, for their kind help over the years.

I am grateful to my collaborators and colleagues Zhiwen Zhang, Zuoqiang Shi, Guo Luo, Mulin Cheng, Xin Hu and Peyman Tavallali, for their stimulating discussions and help. I also wish to thank many other friends who made my graduate studies memorable, including Hongchao Zhou, Yu Zhao, Debbie Yu, Da Yang, Xuan Zhang, Sinan Zhao, Jiang Li, Daiqi Linghu, Liling Gu, Molei Tao, and Tong Chen.

Last but not the least, I would like to thank my parents, Xiumei Shan and Dejun Ci, for their endless love and support. I dedicate this thesis to my family, with all my heart.

Abstract

Partial differential equations (PDEs) with multiscale coefficients are very difficult to solve due to the wide range of scales in the solutions. In the thesis, we propose some efficient numerical methods for both deterministic and stochastic PDEs based on the model reduction technique.

For the deterministic PDEs, the main purpose of our method is to derive an effective equation for the multiscale problem. An essential ingredient is to decompose the harmonic coordinate into a smooth part and a highly oscillatory part of which the magnitude is small. Such a decomposition plays a key role in our construction of the effective equation. We show that the solution to the effective equation is smooth, and could be resolved on a regular coarse mesh grid. Furthermore, we provide error analysis and show that the solution to the effective equation plus a correction term is close to the original multiscale solution.

For the stochastic PDEs, we propose the model reduction based data-driven stochastic method and multilevel Monte Carlo method. In the multiquery, setting and on the assumption that the ratio of the smallest scale and largest scale is not too small, we propose the multiscale data-driven stochastic method. We construct a data-driven stochastic basis and solve the coupled deterministic PDEs to obtain the solutions. For the tougher problems, we propose the multiscale multilevel Monte Carlo method. We apply the multilevel scheme to the effective equations and assemble the stiffness matrices efficiently on each coarse mesh grid. In both methods, the Karhunen-Loève expansion plays an important role in extracting the main parts of some stochastic quantities.

For both the deterministic and stochastic PDEs, numerical results are presented to demonstrate the accuracy and robustness of the methods. We also show the

computational time cost reduction in the numerical examples.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

A broad range of scientific and engineering problems involve partial differential equations (PDEs) with multiple scales. Such disparities appear in virtually all areas of modern science and engineering: composite materials, porous media, turbulence transport in high Reynolds number flows, atmosphere/ocean science, finance, and so on. Also, in recent years, there has been an increasing interest in the simulation of systems with uncertainties. Many physical and engineering applications involving uncertainty quantification can be described by stochastic partial differential equations (SPDEs), and another challenge in uncertainty quantification is solving SPDEs involving multiple scales.

For example, the difficulty in analyzing groundwater transport is mainly caused by the heterogeneity of subsurface formations spanning over many scales, and there is no apparent scale separation. We need to solve PDEs to perform some reliable simulations. Consider the following PDE

$$\nabla \cdot (a(x)\nabla u(x)) = f(x). \tag{1.1}$$

Such a steady-state heterogeneous diffusion equation governs the pressure $u(x)$ in the porous media with permeability $a(x)$ and with source term $f(x)$. The heterogeneity is often represented by the multiscale fluctuations in the permeability $a(x)$. We often need to solve the equation (1.1) many times for different source terms, which is known to be the multiquery setting. Also, due to lack of knowledge of the media

flow, the media properties often contain uncertainties. These uncertainties are usually parameterized, and one deals with a large set of permeability fields with a multiscale nature. We are interested in some expected quantities of the solutions, and need to solve the corresponding SPDEs.

Due to the wide range of scales in these solutions, it is extremely challenging to resolve the small scales of the solutions by direct numerical simulation. Tremendous computational resources are required to solve for the small scales of the solution, which makes it prohibitively expensive to solve such problems. Even for today's computing resources, it is easy to exceed the limit of computer memory or CPU time. Sometimes, from an application perspective, it is often sufficient to predict the macroscopic properties of the multiscale systems, and therefore, we are interested in the large scale solutions. Furthermore, if we want to find out the information at all scales, we can construct the small scale solutions from the large scale solutions by exploring the coupling between them. Thus, finding an effective equation that governs the large scale solution is very important. It is very difficult to derive an effective equation since the coupling between the small scale solution and the large scale solution is in general nonlinear and nonlocal. SPDEs involving multiple scales become more complicated. We not only need to use a very fine mesh to resolve the small scales of the solution in the physical space, but also need to approximate the solution in the stochastic space of which the dimension could be high. Thus, we need to seek accurate numerical methods for PDEs and SPDEs with multiple scales, and reduce the computational cost.

1.2 Literature review

Many multiscale methods have been developed in the literature to solve deterministic and stochastic PDEs. We will discuss some existing numerical methods that are relevant to our model reduction method.

In the fields of control theory, electrical engineering and mechanical engineering, model reduction is a technique to simplify the simulation of dynamical systems described by differential equations. The idea is to project the original, high-dimensional,

state-space onto a properly chosen low-dimensional subspace to arrive at a smaller system that has properties similar to the original system. Complex systems can thus be approximated by simpler systems involving fewer equations and unknown variables, which can be solved much more quickly than the original problem. Here, we borrow the term 'model reduction' and apply it to the multiscale PDEs/SPDEs. Our idea is to construct an effective equation that is similar to the original equation. The new equation governs the large scale information of the solution, and a correction term is easily computed from the large scale solution. Thus, the new equation could be solved by the standard finite element method on a coarse mesh grid. We could use a low-dimensional finite element space to resolve the solution instead of a high-dimensional one, which achieves the purpose of model reduction. Since the solution to the new equation is smooth, we call it the effective equation. However, in general, the coefficients of the effective equations are not as smooth as the solutions. Although the coefficients still have small scale information, we can still solve the effective equation by the finite element method on a coarse mesh grid. In fact, we take the average of the coefficients on the coarse mesh grid when we do integrations through the finite element basis functions. We will see it clearly in Chapter 2.2.

Homogenization (see e.g. [10]) is a powerful tool in understanding the large scale behaviour of the system under the assumption of scale separation and periodic structures. When the coefficients have scale separation and are periodic with respect to the fast variable, we can construct the homogenized coefficients. The large scale solution will satisfy the same kind of equation with the new homogenized coefficients. However, this method is strongly restricted by the assumptions of scale separation and periodic structures, which is not always satisfied in the applications. Also, to capture the small scale information, the construction of the correction term is not feasible for numerical implementation, since it requires the same computational cost as the original problem; see [48]. In our method, we seek to find the new 'homogenized' coefficients without scale separation or periodic structures, and the correction terms are easily computed from the large scale solutions.

In [8, 6], Babuška et al. propose the use of multiscale basis functions for elliptic equations with a special multiscale coefficient that is the product of one-dimensional

fields. This approach is extended by Hou and Wu [37] to general heterogeneities. They construct the multiscale basis functions that satisfy the local multiscale equations, and every solution can be expressed as a linear combination of the local basic functions. They also show that boundary conditions for constructing such basic functions are important for the accuracy of the method. Further techniques are explored to improve the accuracy and efficiency of the method, including the oversampling technique, nonconformal basis, local-global information exchange, special boundary conditions for high contrast problems, spectrum decomposition of the space, and so on; see [38, 27, 14, 24, 25, 18, 22] for reference. As we mentioned, the boundary conditions for the basis functions are essential in the methods, and also become a constraint for some problems. The basis functions are local, so the boundary conditions are not easily determined beforehand. Unlike their methods, we use global harmonic coordinates which can be approximated on a fine mesh grid whose computational cost is no more than constructing the local basis functions.

Another method that uses the concept of basis functions is the multiscale finite volume method proposed by Jenny et al. [40, 33]. It is based on the finite volume method rather than the finite element method. They also introduce the bubble function to improve accuracy. We notice that both the multiscale finite element method and multiscale finite volume method use some kind of basis functions, and the constructions are purely numerical. For the model reduction method, we build a new effective equation without the need for constructing local multiscale basis functions. We point out that any standard numerical method can be applied to our new equation. Our method provides both a theoretical and numerical understanding of the multiscale problem.

The numerical upscaling procedures have also been developed and shown to be effective in many areas, e.g. the local-global upscaling approach by Chen et al. [13]. The main idea of upscaling techniques is to form coarse scale equations with a prescribed form, and these equations are often formed and solved numerically. Our method falls to the category of upscaling methods. The upscaled coefficients have an analytical expression as well as the correction term, and we can prove rigorously that the approximation is accurate under some assumptions.

The metric based upscaling method proposed by Owhadi et al. [51, 52] shares some common features with our model reduction method. They use the harmonic coordinate to construct a multiscale basis and prove convergence of their multiscale method under some mild conditions on the coefficients (see also [1, 3] for more discussions on harmonic coordinates). Specifically, they transform a standard linear finite element basis in the harmonic coordinates to a multiscale basis in the physical coordinates. The solution can be well represented by the multiscale basis. However, this method requires the harmonic mapping to be invertible. The numerical implementation of their method is more complicated than ours, since the coarse mesh grid in the metric based upscaling method is severely deformed due to the transformation of the harmonic mapping. In our approach, we are interested in deriving a global upscaling equation. Moreover, we do not require the harmonic mapping to be invertible, and our effective equation can be solved by the standard finite element method on a regular coarse mesh grid. This makes our method easier to implement, and also more efficient.

Among other methods for deterministic PDEs with multiple scales, there are the following: the variational multiscale method by Hughes et al. [39], the heterogeneous multiscale method [21] by E et al., the domain decomposition method by Graham et al. [31, 28], the multiscale finite element method for numerical homogenization by Allaire et al. [2], the multiscale mortar mixed finite element method by Arbogast et al. [4], finite point method by Han et al. [34], and so on. We will not list all the details of the methods; instead, we will switch to reviewing the numerical methods for SPDEs.

The stochastic finite element method [30, 54] has a lot of applications, and is very powerful. It uses the spectral expansion of the random functions on some polynomial chaos, for example, Hermite polynomials of independent random variables, and the Galerkin approach to approximate the expansion in the deterministic space. By adopting the techniques of the deterministic Galerkin approach, error estimations can be derived. Furthermore, it enjoys fast convergence when the solution is sufficiently regular. However, a very large algebraic system is typically associated with the Galerkin approach. When we have multiscale coefficients, the resulting linear sys-

tem becomes very large. Both computational cost and memory consumption become prohibitively expensive.

The Wiener chaos expansion or the generalized polynomial chaos (gPC) method [42, 36, 56, 58, 59, 55, 49, 45] employ truncated expansions on a set of polynomial chaos basis functions and drive a system of coupled deterministic PDEs. In the past decade or so, a lot of progress has been made in developing an effective gPC type of methods to solve SPDEs arising from various applications. However, this type of methods still suffers from the curse of dimensionality in the sense that the total number of polynomial chaos basis functions grows quickly as the number of independent random variables becomes larger and larger.

The stochastic collocation method [57, 7, 44] has been developed from the non-intrusive deterministic collocation method and sparse grid techniques. In principle, the stochastic collocation method uses multivariate polynomial interpolations for the integral in the variational formulation of the stochastic system with respect to probability space. A deterministic sequence of points resulting from tensor products of one-dimensional quadrature points is sampled. The exact locations of such points and weights associated with them depend on underlying probability distributions. Moreover, hierarchical construction of a generalized sparse grid has also been developed for the application of the stochastic collocation method.

We can see that the so-called curse of dimensionality is one of the essential challenges in the uncertainty quantification. Recently, a data-driven stochastic method [15, 60] was proposed by constructing a problem-dependent stochastic basis to solve these SPDEs. The SPDEs enjoy a compact representation for a broad range of forcing functions under such a stochastic basis. We will solve a number of coupled deterministic PDEs by projecting the stochastic solution onto the data-driven stochastic basis and obtain the desired quantities. However, when the coefficients have multiple scales, fine mesh grids are needed to resolve the small scale information, and the corresponding large coupled system makes it very difficult to solve. Thus, we combine the data-driven basis idea with the model reduction technique, and greatly reduce the computational cost by considering the effective equations on coarse mesh grids.

The multilevel Monte Carlo method (MLMC) was first introduced by Giles in solv-

ing SDEs arising in mathematical finance [32]. Similar ideas have been introduced by Heinrich for finite-dimensional parametric integration and solving integral equations [35]. Later, the MLMC method was extended to solve elliptic PDEs with random coefficients; see [9, 19]. The MLMC method is very effective when the solutions to SPDEs are smooth. However, if the solutions of SPDEs possess multiscale features, a naive application of MLMC does not give very good performance, since the asymptotic variance reduction between two consecutive levels is not valid at the coarse grids unless we make the coarse grids fine enough to resolve the smallest scale feature of the solution. In [19], the authors remark that, the optimal choice for the coarsest level is that the coarsest mesh size should be slightly smaller than the correlation length of the random field, which becomes a major limitation for the method. We will apply the multilevel scheme to the effective equations, and design an efficient numerical method to alleviate the difficulty.

Some other numerical approaches have been proposed to solve SPDEs by exploring the sparse structure of the solutions; e.g., the dynamically bi-orthogonal method [16, 17] by Cheng et al. Also, Zabaras et al. propose a stochastic variational multiscale method for diffusion in heterogeneous random media [5, 29]. They combined the generalized polynomial chaos method with the variational multiscale method to achieve model reduction. However, when the dimension in stochastic direction is large, this method is inefficient due to the exponential growth of the number of the gPC basis elements.

1.3 Model reduction

We will briefly discuss the model reduction methods for different cases and illustrate our contributions.

1.3.1 Deterministic case

We use the following elliptic equation as an example to illustrate the main idea of the model reduction approach:

$$\begin{cases} -\nabla \cdot (a(x)\nabla u(x)) = f(x), & x \in D, \\ u(x) = 0, & x \in \partial D, \end{cases} \tag{1.2}$$

where $D \in R^d$ is a spatial domain. The multiscale information is described by the coefficient $a(x)$. We assume that $f(x) \in L^2(D)$ is smooth, and $a(x) \in L^\infty(D)$ is a symmetric, positive definite matrix satisfying $\lambda_{min}(x) \geq \gamma > 0$ ($\lambda_{min}(x)$ is the smallest eigenvalue of $a(x)$) for a.e. $x \in D$. For such coefficients, the solutions are only Hölder continuous. If a has multiple scales, the solution will have multiple scales as well.

We would like to design an effective equation in the following form

$$\begin{cases} -\nabla \cdot (a^*(x)\nabla u^*(x)) = f(x), & x \in D, \\ u^*(x) = 0, & x \in \partial D. \end{cases} \tag{1.3}$$

The key is how to construct an effective coefficient a^* so that the solution to the above effective equation approximates the original multiscale solution with some desirable accuracy. We use the harmonic coordinates as a tool, which satisfy the same governing equation with homogeneous source term and linear boundary conditions. We know that the solution in the harmonic coordinates is one order smoother than in the physical coordinates (see e.g. [51]). An important ingredient of our method is to design an appropriate decomposition of the harmonic coordinates, denoted as F, into a smooth component g plus a small component χ, $F = g + \chi$. We will illustrate how we perform the decomposition and what we mean by smooth component and small component in Chapter 2. Our method does not require F to be invertible, but to motivate the derivation of our method, we assume temporarily that F is invertible. Then, we can express u as a function of F. One important property of the harmonic coordinates is that u as a function of F is about one order smoother than u as a function of x (see e.g. [51]). Thus, we can write $u(x) = \tilde{u}(F) = \tilde{u}(g + \chi)$, and

formally expand $\tilde{u}$ around g by assuming that χ is small. By taking the leading order terms and substituting them into the original equation, we obtain an effective equation of form (1.3) after ignoring the high order terms involving χ. The effective coefficient a^* is defined in terms of a, g, and χ as following

$$a^*(x) = a(x)(I + \frac{\partial \chi}{\partial x}(x))\frac{\partial x}{\partial g}(x), \tag{1.4}$$

where I is the identity matrix.

We can show that the effective equation derived by the above formal analysis indeed has the desirable smoothness property. Under some conditions, we will show that the solution to the effective equation is in H^2, which is one order smoother than the original multiscale solution. Thus, we can solve the effective equation on a coarse mesh. Moreover, we can show that the error term is small in the H^1 norm under some conditions. From our derivation, we can see that the decomposition of the harmonic coordinates determines the effective coefficient, a^*. An optimal effective coefficient will determine an optimal decomposition. The relationship between these two terms helps us to design a nearly optimal decomposition of F.

Our method falls into the category of global upscaling methods. To obtain our effective equation, we need to first solve for the harmonic coordinates, which amounts to solving the original equation d times (d is the physical dimension of the problem). If we just solve the elliptic equation once, our method would not save computational cost. However, if we consider the multiquery setting, i.e. we need to solve the equation with the same coefficient many times with different source terms, the cost of constructing the effective coefficient is a small overhead in the offline step. The online step of solving the effective equation with multiple right hand sides gives considerable computational saving, since the effective equation can be solved on a coarse mesh while the original multiscale problem must be solved on a fine mesh. For time dependent problems such as parabolic, hyperbolic, and convection-diffusion equations, if the coefficients are time-independent, our method gives considerable computational savings even for a single forcing, since the overhead of constructing the effective coefficient is negligible compared with the cost of solving the time-dependent equations

on a fine mesh at the subsequent time steps.

1.3.2 Stochastic case with data-driven stochastic method

We propose a multiscale data-driven stochastic method to perform model reduction in both the stochastic and physical spaces. Our new method consists of offline and online stages. In the offline stage, we first derive an effective stochastic equation that can be resolved on a coarse grid. We then construct a data-driven stochastic basis under which the solutions of the effective stochastic equation have a compact representation for a broad range of forcing functions. We consider the following elliptic SPDE with multiscale random coefficients in the multiquery setting

$$\begin{cases} -\nabla \cdot (a(x,\omega)\nabla u(x,\omega)) = f(x), & x \in D, \omega \in \Omega, \\ u(x,\omega) = 0, & x \in \partial D, \omega \in \Omega. \end{cases} \tag{1.5}$$

where $D \in R^d$ is a spatial domain, and Ω is a sample space. The multiscale information is also described by the coefficient matrix $a(x,\omega)$. We assume that $f(x) \in L^2(D)$ is smooth and $a(x,\omega) \in L^\infty(D)$ is a symmetric, positive definite matrix satisfying $\lambda_{min}(x,\omega) \geq \gamma > 0$ ($\lambda_{min}(x,\omega)$ is the smallest eigenvalue of $a(x,\omega)$) for a.e. $x \in D$, $\omega \in \Omega$. We would like to derive a similar effective stochastic equation in the following form

$$\begin{cases} -\nabla \cdot (a^*(x,\omega)\nabla u^*(x,\omega)) = f(x), & x \in D, \omega \in \Omega, \\ u^*(x,\omega) = 0, & x \in \partial D, \omega \in \Omega. \end{cases} \tag{1.6}$$

For each fixed sample $\omega \in \Omega$ (ω can be chosen by the Monte Carlo or stochastic collocation method), the multiscale problem (1.5) becomes deterministic, and we would obtain the effective coefficient as (1.4). Under some conditions, we can show that the solution to the effective equation is smooth, and the difference between the two solutions is small. This is how we perform model reduction in the physical space.

To perform model reduction in the stochastic space, we adopt the Karhunen-Loève expansion [41, 43] for the stochastic coefficients and solutions. It is well known that the Karhunen-Loève expansion can generate an optimal basis in the sense that it minimizes the total mean squared error. Our method consists of two essential parts: a com-

pact parametrization for the effective coefficient $a^*(x,\omega)$, and a problem-dependent compact basis to represent the stochastic solutions to the effective equation (1.6).

In the first part, we compute the truncated Karhunen-Loève expansion of the matrix $a^*(x,\omega)$. This compact representation of the effective coefficient enables us to generate $a^*(x,\omega)$ very efficiently in the online stage. In the second part, we construct a data-driven stochastic basis by applying the data-driven stochastic method [15] to the effective equation (1.6). We first choose a set of forcing functions and solve (1.6) with one forcing function. Then, we use the Karhunen-Loève expansion of the solution to construct the stochastic basis $\{B_i(\omega)\}_{i=0}^m$. An error analysis is used to evaluate the completeness of the data-driven basis $\{B_i(\omega)\}_{i=0}^m$. A greedy-type algorithm combined with a two-level preconditioning [20] is used to reduce the computational cost. First, we solve the error equation on the coarse grid for each trial function $f_k(x)$, $k = 1, 2, \cdots, K$. We identify the trial function f_{k^*}, which gives the maximum error and solve the error equation again with this trial function on the fine grid. After that, the Karhunen-Loève expansion of the residual error can be used to enrich the stochastic basis. This process is repeated until the maximum residual error is below the prescribed threshold ϵ. When this updating process terminates, we obtain a compact data-driven basis $\{B_i(\omega)\}_{i=0}^m$ for the effective stochastic equation (1.6), which applies to all forcing functions. The detailed implementation of this enriching algorithm depends on specific numerical representation of the stochastic basis, which will be elaborated in detail in Chapter 5.

In the online stage, we expand the solution of (1.6) in terms of the data-driven stochastic basis, and solve a set of coupled deterministic PDEs to obtain the solutions. As in the deterministic case, since we need to solve the equation many times with different forcing functions but the same coefficients, our method in the online stage offers considerable computational savings.

1.3.3 Stochastic case with multilevel Monte Carlo method

The multiscale data-driven stochastic method works very well under the assumption that the ratio of the smallest scale and largest scale is not too small. However, for tougher problems, the offline computational cost would become more and more ex-

pensive, and the online coupled system might become bigger and bigger. In this case, we need to find a more efficient method. We again consider the same elliptic SPDE (1.5), and we are interested in the expected value of some functional of the solutions, which could be the mean and high-order moments. In general, we could approximate the expectations by the standard Monte Carlo method. For the multiscale problems, we must choose the mesh size to be fine enough to control the bias error. Also, many realizations are required to reduce sampling errors.

We have already proven that the solution to the effective equation is a good approximation of the original solution. Thus, we could approximate the effective solution and pick the mesh grid much coarser than the original fine mesh grid, which would save a large amount of computational time a lot for each realization.

To further reduce the computational cost, we apply the multilevel scheme proposed in [32] to the effective equation. We first divide the physical domain D into a number of nested coarse mesh grids, i.e., $D_{h_0} \subset \ldots \subset D_{h_{l-1}} \subset D_{h_l} \ldots \subset D_{h_L}$. Here, h_l is the l-th level mesh size ($l = 0, 1, \ldots, L$) and h_0 is the coarsest level mesh size. Denote $\mathbb{E}[u_l^*(x,\omega)] = \mathbb{E}[u_{h_l}^*(x,\omega)]$ to be the mean of the numerical solution on mesh size h_l; linearity of the expectation operator implies that

$$\mathbb{E}[u_L^*(x,\omega)] = \mathbb{E}[u_0^*(x,\omega)] + \sum_{l=1}^{L} \mathbb{E}[u_l^*(x,\omega) - u_{l-1}^*(x,\omega)]. \tag{1.7}$$

The key point is to avoid estimating $\mathbb{E}[u_L^*(x,\omega)]$ on the finest level, but instead to estimate it on the coarsest level. The reduction in cost associated with the multilevel Monte Carlo method over the Monte Carlo method is due to the fact that most of the uncertainty can be captured on the coarse grids ($h >> h_L$), so the number of realizations needed on the grid ($h = h_L$) is greatly reduced due to the variance reduction between two consecutive grids. We will show the computational cost reduction by several numerical examples.

We also design an efficient method in assembling the stiffness matrices on each coarse mesh grid so that our method would provide significant computational time reduction. Thus, compared with MLMC, we can perform the multilevel scheme on the coarse mesh grids instead of fine mesh grids, although the solutions have multiscale

information. As we can see in Chapter 6, the offline computation cost is cheaper than the exact solver, and thus, it even works for the SPDE with one forcing term. In the multiquery setting, we could gain more computational savings. Here, we also need to mention that the MsMLMC could tackle more difficult problems, but the MsDSM is more accurate and fast in the online stage when we have enough computational resources.

Chapter 2

Multiscale model reduction method for elliptic PDEs

2.1 Derivation of effective equations

In this chapter, we consider the elliptic equation

$$\begin{cases} -\nabla \cdot (a(x)\nabla u(x)) = f(x), & x \in D, \\ u(x) = 0, & x \in \partial D. \end{cases} \tag{2.1}$$

Let $F(x) = (F_1(x), ..., F_d(x))$ be the harmonic coordinate associated to (2.1) in d-dimensional space. Then F_k $(k = 1, ..., d)$ satisfies the following elliptic equation

$$\begin{cases} -\nabla \cdot (a(x)\nabla F_k(x)) = 0, & x \in D \\ F_k(x) = x_k, & x \in \partial D, \end{cases} \tag{2.2}$$

where $x = (x_1, ..., x_d)$. Write $\tilde{u}_0 = u \circ F^{-1}$. It is known that the solution u is smooth in terms of the harmonic coordinates, i.e. $\tilde{u}_0$ is in H^2. If we could make a decomposition $F = g + \chi$ such that g is smooth and χ is small with zero boundary conditions, then we obtain by applying a formal Taylor expansion to $\tilde{u}_0$ and ignoring the high order terms,

$$\tilde{u}_0(F) = \tilde{u}_0(g + \chi) \approx \tilde{u}_0(g) + \chi^T \nabla_g \tilde{u}_0(g). \tag{2.3}$$

Let $u_0(x) = \tilde{u}_0(g(x))$, then we get

$$u(x) = \tilde{u}_0(F) \approx u_0(x) + \chi^T \frac{\partial x}{\partial g} \nabla u_0(x). \tag{2.4}$$

Furthermore, we have

$$\nabla u(x) \approx \nabla u_0(x) + \frac{\partial \chi}{\partial x}\frac{\partial x}{\partial g}\nabla u_0(x) + \chi^T \nabla (\frac{\partial x}{\partial g}\nabla u_0(x)). \tag{2.5}$$

By substituting (2.5) into (2.1) and eliminating the high order terms involving $O(\chi)$, we get a new PDE for u_0

$$\begin{cases} -\nabla \cdot (a(x)(I + \frac{\partial \chi}{\partial x}\frac{\partial x}{\partial g})\nabla u_0(x)) = f(x), & x \in D, \\ u_0(x) = 0, & x \in \partial D, \end{cases} \tag{2.6}$$

where I is the identity matrix.

We will show that u_0 is in H^2 so that we can solve the effective equation (2.6) accurately on a coarse mesh. Moreover, we will show that the H^1 norm of the error, $u - (u_0 + \chi^T \frac{\partial x}{\partial g}\nabla u_0)$, is small. Thus we can approximate u by $u_0 + \chi^T \frac{\partial x}{\partial g}\nabla u_0$ with a reasonable accuracy. This suggests the following steps of the model reduction method.

1. Solve the harmonic coordinate (2.2) on a fine mesh to get F.
2. Decompose $F = g + \chi$, here g is smooth and χ is small with $\chi = 0$ on $\partial\Omega$.
3. Solve the effective equation (2.6) on a coarse mesh to get u_0.
4. Approximate u by $u_0 + \chi^T \frac{\partial x}{\partial g}\nabla u_0$.

The first and second steps are offline steps. We can store the necessary information so that we can compute u_0 efficiently for different f. The remaining online steps can be solved very efficiently on a coarse mesh. So far we have not defined what we mean by g being 'smooth' and χ being 'small'. We will discuss the guideline in defining g and χ, and give one effective construction in the next sections.

2.2 Analysis

In this section, we will perform error analysis to show that the H^1 norm of the difference between the effective solution and the original solution is small provided

that the L^∞ norm of χ is small. We will also prove that the solution to the effective equation is in H^2 under some conditions. Before presenting the general analysis, we will start with the simple 1-dimensional elliptic equation to illustrate the main idea.

2.2.1 The one-dimensional case

Consider the one-dimensional elliptic equation on a unit interval $[0, 1]$

$$\begin{cases} (a(x)u'(x))' = f(x), & x \in (0,1), \\ u(0) = u(1) = 0. \end{cases} \tag{2.7}$$

The corresponding harmonic coordinate F is defined as

$$\begin{cases} (a(x)F'(x))' = 0, & x \in (0,1), \\ F(0) = 0,\ F(1) = 1. \end{cases} \tag{2.8}$$

Our effective equation is given by

$$\begin{cases} a(x)F'(x)(u_0'(x)/g'(x))' = f(x), & x \in (0,1), \\ u_0(0) = u_0(1) = 0. \end{cases} \tag{2.9}$$

We can solve these equations analytically and get

$$u(x) = C_0 \left(F(x) \int_0^x f(s)\mathrm{d}s - \int_0^x F(s)f(s)\mathrm{d}s + C_1 F(x) \right), \tag{2.10}$$

$$F(x) = \frac{1}{C_0} \int_0^x \frac{\mathrm{d}s}{a(s)}, \tag{2.11}$$

$$u_0(x) = C_0 \left(g(x) \int_0^x f(s)\mathrm{d}s - \int_0^x g(s)f(s)\mathrm{d}s + C_2 g(x) \right), \tag{2.12}$$

where

$$C_0 = \int_0^1 \frac{\mathrm{d}s}{a(s)}, \tag{2.13}$$

$$C_1 = -\int_0^1 f(s)\mathrm{d}s + \int_0^1 F(s)f(s)\mathrm{d}s, \tag{2.14}$$

$$C_2 = -\int_0^1 f(s)\mathrm{d}s + \int_0^1 g(s)f(s)\mathrm{d}s. \tag{2.15}$$

Since $f \in L^2$ and g is smooth, we can see that u_0 is smooth. Let $u_1 = \chi u_0'/g'$, direct computations give

$$(u - u_0 - u_1)' = C_0\left((C_1 - C_2)F' - \chi f\right). \tag{2.16}$$

We also have

$$C_1 - C_2 = \int_0^1 \chi(s)f(s)\mathrm{d}s, \tag{2.17}$$

and

$$F' = \frac{1}{C_0}\frac{1}{a(x)}. \tag{2.18}$$

By the assumption $a(x) \geq \gamma > 0$, we know that F' is bounded. Thus we can bound $\|u - u_0 - u_1\|_{H^1}$ in terms of $\|\chi\|_{L^\infty}$. This implies that as long as we can decompose F so that the oscillatory part χ is small and g is smooth, then $\|u - u_0 - u_1\|_{H^1}$ is small and we can solve u_0 on a coarse grid. In the next section, we will show that this result is true for general multi-dimensional elliptic equations as well.

2.2.2 An error estimate for the general case

The main result of this section is the following theorem.

Theorem 2.1. *Suppose u, F and u_0 are weak solutions to (2.1), (2.2) and (2.6) respectively. Let $u_1 = \chi^T \frac{\partial x}{\partial g}\nabla u_0$, $F = g + \chi$, and $\chi = 0$ on $\partial\Omega$. Then we have*

$$\|u - u_0 - u_1\|_{H^1(D)} \leq C\|\chi\|_{L^\infty(D)}\|\frac{\partial g}{\partial x}\|_{L^\infty(D)}\|\det(\frac{\partial x}{\partial g})\|_{L^\infty(D)}|\tilde{u}_0|_{H^2(D)}, \tag{2.19}$$

where C is a constant that depends on d, D and a, $\tilde{u}_0 = u_0 \circ g^{-1}$.

Our goal is to estimate the H^1 norm of $z = u - u_0 - u_1$. The zero-boundary

condition of χ implies the zero-boundary condition of z, i.e. $z \in H_0^1(\Omega)$. Since a is a positive definite matrix whose eigenvalues have a positive lower bound, we conclude that $||z||_{H_0^1(\Omega)}$ is equivalent to the energy norm, $< a\nabla z, \nabla z >$, where $< . >$ is the L^2 inner product. Thus, it is sufficient to perform our estimate using the energy norm.

Proof. Define

$$z = u - u_0 - u_1 \in H_0^1(D),$$
$$p = a\nabla u - a\frac{\partial F}{\partial x}\frac{\partial x}{\partial g}\nabla u_0,$$
$$\eta = -a\frac{\partial}{\partial x}\left(\frac{\partial x}{\partial g}\nabla u_0\right)\chi.$$

Then we have

$$\nabla \cdot p = \nabla \cdot (a\nabla u) - \nabla \cdot (a\frac{\partial F}{\partial x}\frac{\partial x}{\partial g}\nabla u_0) = f - f = 0, \tag{2.20}$$

and

$$a\nabla z - p = a\nabla u - a\nabla u_0 - a\nabla u_1 - a\nabla u + a\frac{\partial F}{\partial x}\frac{\partial x}{\partial g}\nabla u_0 = a\frac{\partial \chi}{\partial x}\frac{\partial x}{\partial g}\nabla u_0 - a\nabla u_1.$$

Further, we note that

$$\nabla u_1 = \frac{\partial \chi}{\partial x}\frac{\partial x}{\partial g}\nabla u_0 + \frac{\partial}{\partial x}\left(\frac{\partial x}{\partial g}\nabla u_0\right)\chi.$$

As a result, we get

$$a\nabla z - p = -a\frac{\partial}{\partial x}\left(\frac{\partial x}{\partial g}\nabla u_0\right)\chi = \eta.$$

Thus, we obtain

$$\begin{aligned} < a\nabla z, \nabla z > &= \int_D \nabla z \cdot a\nabla z \mathrm{d}x \\ &= \int_D (a\nabla z - p) \cdot \nabla z \mathrm{d}x + \int_D p \cdot \nabla z \mathrm{d}x \\ &= \int_D \eta \cdot \nabla z \mathrm{d}x - \int_D (\nabla \cdot p) z \mathrm{d}x \\ &= \int_D \eta \cdot \nabla z \mathrm{d}x, \end{aligned}$$

where we have used (2.20). By using the ellipticity assumption of a, we get

$$C_1\|\nabla z\|^2_{L^2(D)} \leq < a\nabla z, \nabla z > \leq \|\eta\|_{L^2(D)}\|\nabla z\|_{L^2(D)},$$

which implies

$$C_1\|\nabla z\|_{L^2(D)} \leq \|\eta\|_{L^2(D)}.$$

Since z vanishes on ∂D, Poincaré's theorem gives

$$\|z\|_{H^1_0(D)} \leq C_2\|\nabla z\|_{L^2(D)}.$$

Thus, we get

$$\|z\|_{H^1_0(D)} \leq C\|\eta\|_{L^2(D)},$$

where C is a constant that depends on d, D and a only.

Let $y = g(x)$ and $\tilde{u}_0(y) = \tilde{u}_0(g(x)) = u_0(x)$, then we have

$$\begin{aligned}
\eta &= -a\frac{\partial}{\partial x}\left(\frac{\partial x}{\partial g}\nabla u_0\right)\chi \\
&= -a\frac{\partial g}{\partial x}\frac{\partial x}{\partial g}\frac{\partial}{\partial x}\left(\frac{\partial x}{\partial g}\nabla u_0\right)\chi \\
&= -a\left(\frac{\partial g}{\partial x}\nabla_y^2\tilde{u}_0\right)\chi.
\end{aligned}$$

As a result, we obtain

$$\|\eta\|_{L^2(D)} \leq C\|a\|_{L^\infty(D)}\|\chi\|_{L^\infty(D)}\|\frac{\partial g}{\partial x}\|_{L^\infty(D)}\|\det(\frac{\partial x}{\partial g})\|_{L^\infty(D)}|\tilde{u}_0|_{H^2(D)}.$$

The determinant of $\frac{\partial x}{\partial g}$ enters the last step of the above estimate due to a change of variables from x to y. Combining all the results, we get

$$\|z\|_{H^1_0(D)} \leq C\|\chi\|_{L^\infty(D)}\|\frac{\partial g}{\partial x}\|_{L^\infty(D)}\|\det(\frac{\partial x}{\partial g})\|_{L^\infty(D)}|\tilde{u}_0|_{H^2(D)}.$$

This completes the proof of Theorem 2.1. □

The error estimate given by Theorem 2.1 provides us with some insight how to choose our decomposition $F = g + \chi$. If $\|\chi\|_{L^\infty(D)}$ is small and g is smooth in the

sense that both $\|\frac{\partial g}{\partial x}\|_{L^\infty(D)}$ and $\|\det(\frac{\partial x}{\partial g})\|_{L^\infty(D)}$ are bounded, then the approximation is accurate provided that $|\tilde{u}_0|_{H^2(D)}$ is bounded. To evaluate $|\tilde{u}_0|_{H^2(D)}$, we make a change of variables from x to $y = g(x)$ in equation (2.6), which has a non-divergence form

$$-\sum_{i,j=1}^{n} B_{ij}(y)\frac{\partial^2 \tilde{u}_0}{\partial y_i \partial y_j} = \tilde{f}(y), \tag{2.21}$$

where

$$B(y) = \left(|\det(\frac{\partial x}{\partial g})|(\frac{\partial g}{\partial x})^T a \frac{\partial F}{\partial x}\right) \circ g^{-1}(y) \tag{2.22}$$

and

$$\tilde{f}(y) = \left(|\det(\frac{\partial x}{\partial g})| f\right) \circ g^{-1}(y). \tag{2.23}$$

Note that the determinant term comes from the change of variables in the integral since we consider weak solutions.

If the matrix B satisfies the Cordes condition (see e.g. [46]),

$$\frac{\sum_{i,j=1}^{n} B_{ij}^2(y)}{(\sum_{i=1}^{n} B_{ii}(y))^2} \leq \frac{1}{n-1+\epsilon}, \tag{2.24}$$

for some $\epsilon > 0$ and $M = \sup(\frac{\sum_{i=1}^{n} B_{ii}(y)}{\sum_{i,j=1}^{n} B_{ij}^2(y)}) < \infty$, we can apply Theorem 1.2.1 in [46] to conclude that

$$|\tilde{u}_0|_{H^2(D)} \leq \frac{M}{1-\sqrt{1-\epsilon}}\|f\|_{L^2(D)}. \tag{2.25}$$

In general, the condition (2.24) is hard to verify. For $d = 2$, we have a more concrete version for ϵ and M. Suppose $\lambda_{\max}(y)$ and $\lambda_{\min}(y)$ are the maximum and minimum eigenvalues of $B(y)$, if $\eta_1 = \sup \frac{\lambda_{\max}(y)}{\lambda_{\min}(y)} < \infty$ and $\eta_2 = \inf \lambda_{\min}(y) > 0$, then we can pick $\epsilon = \frac{1}{\eta_1}$ and $M = \frac{1}{\eta_2}$.

Remark 2.1. Theorem 2.1 is an error estimate for the analytical solutions. One should not use it to study the convergence rate of the numerical method. It does not imply that the smaller $\|\chi\|_{L^\infty(D)}$ is, the smaller the numerical error would become. On one hand, if we choose χ to be 0, in which case $g = F$, the error is zero in theory. However, g is no longer smooth and we will have to use a fine mesh to solve the effective equation, which is not what our method is designed for. Similarly, if we let χ decay to zero, g will pick up more small scales and the derivative of g increases. In

this case, we will not be able to obtain a small overall error if we use a coarse mesh.

From a view point of numerical implementation, it is usually the case that F may become degenerated in some localized region. The 'smoothness' of g depends on the size of the numerical grid. To avoid degeneracy in constructing g, we can use a finer mesh locally to capture some important information in certain local regions, and use a coarser mesh in other regions. By doing this, g is smooth when using a non-uniform mesh, and we can guarantee that χ is small. Choosing an optimal decomposition which would lead to the smallest overall error requires a delicate balance in our decomposition of F. We will discuss more about this issue later.

Remark 2.2. In our analysis, we choose the homogeneous boundary condition. We can still apply our method to nonhomogeneous boundary conditions, as long the the boundary values are smooth, although we do not have the convergence analysis due to the technical reason. When we use the finite element method, we can approximate the boundary values by directly using the finite element basis functions on the boundary. We do not need to transform the nonhomogeneous problem to the homogeneous one by subtracting some function which has the given boundary values, since it will make the new forcing term nonsmooth.

Remark 2.3. The solution in equation (2.6) is in H^2, which is one order smoother than the original solution, and that is why we call it the effective equation. However, the coefficient $a^* = a\frac{\partial F}{\partial x}\frac{\partial x}{\partial g}$ is not smooth in general. For one-dimensional problems, $a^* = aF'/g'$, and aF' is a constant according to our previous analysis. In this case, a^* is smooth since g is smooth. For high dimensional problems, $a\frac{\partial F}{\partial x}$ is no longer constant, so a^* is not smooth. By the Cordes condition, we know that the smoothness of u^0 comes from the fact that we can rewrite equation (2.6) in a non-divergence form. To be more specific, the term $a\frac{\partial F}{\partial x}$ is divergence free, so we have a non-divergence form of equation (2.6).

Although the coefficient a^* is not smooth, we can still solve equation (2.6) by the standard finite element method on a coarse mesh grid. We need to take the average of the coefficient over the coarse mesh grid. Thus, the finite element method is a good tool for us. In fact, taking the average of the coefficient is done implicitly when we

do integrations through the finite element basis functions.

2.3 Comparison with the homogenization method

In this section, we compare our method with the classical homogenization method. First we briefly review the homogenization theory [10]. Consider

$$\begin{cases} -\nabla \cdot (a(\frac{x}{\epsilon})\nabla u(x)) = f(x), & x \in D, \\ u(x) = 0, & x \in \partial D, \end{cases} \tag{2.26}$$

where $a(y)$ is a symmetric, positive definite matrix, and $f \in L^2$. Furthermore, $a_{ij}(y)$ are periodic smooth functions in y in a unit cube Y.

The homogenized coefficients are given by

$$a^*_{ij} = \frac{1}{|Y|}\int_Y a_{ik}(y)(\delta_{kj} + \frac{\partial}{\partial y_k}\chi^j_h)\mathrm{d}y, \tag{2.27}$$

where χ^j_h (we use the notation χ^j_h to distinguish from χ^j) is the solution to the periodic cell problem

$$\nabla_y \cdot (a(y)\nabla_y \chi^j_h(y)) = -\frac{\partial}{\partial y_i}a_{ij}(y) \tag{2.28}$$

with zero mean, i.e. $\int_Y \chi^j_h \mathrm{d}y = 0$.

Let u_0 be the solution to the homogenized equation

$$\begin{cases} -\nabla \cdot (a^*(x)\nabla u_0(x)) = f(x), & x \in D, \\ u_0(x) = 0, & x \in \partial D. \end{cases} \tag{2.29}$$

Then we have

$$||u - u_0||_{L^2(D)} \leq C\epsilon ||u||_{H^2(D)}. \tag{2.30}$$

Further, we define

$$u_1(x) = \chi^j_h(\frac{x}{\epsilon})\frac{\partial u_0}{\partial x_j}(x). \tag{2.31}$$

Note that $u_0 + \epsilon u_1 \neq 0$ on ∂D, so we introduce a first order correction term θ_ϵ

satisfying

$$\begin{cases} -\nabla \cdot (a(\frac{x}{\epsilon})\nabla\theta_\epsilon(x)) = 0, & x \in D, \\ \theta_\epsilon(x) = u_1, & x \in \partial D. \end{cases} \tag{2.32}$$

Then it can be shown that (see e.g. [48])

$$||u - u_0 - \epsilon(u_1 - \theta_\epsilon)||_{H^1(D)} \leq C\epsilon||u_0||_{H^2(D)}. \tag{2.33}$$

The constant C is independent of u_0 and ϵ.

One important advantage of our method is that we can take care of a continuum of scales and do not require periodicity on the microstructure while homogenization theory usually requires scale separation and periodic structures. Moreover, one must include a boundary correction term θ_ϵ to achieve H^1 convergence in the homogenization method. This correction term must be solved on a fine mesh grid and is expensive to compute. In comparison, there is no need to compute a correction term in our method since we require $\chi = 0$ on ∂D.

If only L^2 convergence is needed, both methods do not need correction terms, and the homogenized coefficients are easier to compute (see (2.27) and (2.28)). Our method requires two global solutions on a fine mesh. However, under the conditions for homogenization (periodic smooth $a(y)$), we can modify our method easily so that we can compute the harmonic coordinates with the same cost as the homogenization method. Specifically, the harmonic coordinate F satisfies $\nabla \cdot (a(\frac{x}{\epsilon})\nabla F(x)) = 0$ and $F = g + \chi$. If we set $g = x$, we get the equation for χ as follows

$$\nabla \cdot (a(\frac{x}{\epsilon})\nabla\chi^j(x)) = -\frac{1}{\epsilon}\frac{\partial}{\partial y_i}a_{ij}(\frac{x}{\epsilon}). \tag{2.34}$$

Now we do not require $\chi = 0$ on the boundary, assume χ^j to be periodic in Y and impose the constraint $\int_Y \chi^j(x)\mathrm{d}x = 0$. Equation (2.34) is still global, but comparing (2.34) with (2.28) gives $\chi(x) = \epsilon\chi_h(\frac{x}{\epsilon})$. So we can solve (2.28) instead of (2.34). Following the proof in Theorem 2.1, we can obtain the same error estimate as (2.30).

2.4 Numerical Implementation

2.4.1 Decomposition of the harmonic coordinates

In this section, we discuss how to construct the decomposition of the harmonic coordinates. As we know from the previous sections, the decomposition $F = g + \chi$ plays an essential role in our method. Here we discuss some guidelines in choosing such a decomposition and how to construct it numerically.

The first criterion in choosing our decomposition is to make sure that g is smooth and invertible. We need to define what we mean by g being smooth. The smoothness is relative to the coarse mesh that we will use to solve the effective equation. In our numerical implementation, we use the standard linear finite element method to solve the effective equation. Thus, any linear combination of the nodal basis on the coarse mesh could be considered as a smooth function, and we can choose g in this form.

The second criterion of our decomposition is to make χ small. If we choose the nodal values of g close to those of F, then we expect that the difference between the two would be small. This suggests a natural way to define g, i.e. we can choose the nodal values of g at the coarse mesh points to be the same as F at these coarse mesh points. We can then interpolate g from the coarse mesh points to the fine mesh points using the linear interpolation. Once we have defined g globally through linear interpolation, we have also determined $\chi = F - g$. Since F is linear on the boundary, such decomposition guarantees that $g = F$ on the boundary, which implies that $\chi = 0$ on ∂D.

We also have another guideline to determine whether the decomposition is effective or not from the view point of numerical implementation. Note that $u_g = g$ solves the following equation exactly:

$$\begin{cases} -\nabla \cdot (a(x)(I + \frac{\partial \chi}{\partial x}\frac{\partial x}{\partial g})\nabla u_g(x)) = 0, & x \in D, \\ u_g(x) = x, & x \in \partial D. \end{cases} \tag{2.35}$$

If g is smooth enough, we should be able to solve the above equation (2.35) accurately on a coarse mesh. Thus, we can use the difference between the numerically computed

u_g and g to determine whether we obtain a good decomposition for F. The smaller the difference is, the better the method would perform.

2.4.2 Numerical results

We perform several numerical experiments to test our multiscale model reduction method (MsMRM) for the elliptic equation (2.1). We take $D = [0,1] \times [0,1]$ in 2-dimensional space. Since it is difficult to construct a general enough test problem with an analytic solution, we use well resolved numerical solutions in place of exact solutions. In our computations, we use the standard linear finite element method (FEM). We compare the solution on a 256×256 mesh and a 512×512 mesh. The L^2 relative error is less than 1×10^{-3} and the H^1 relative error is less than 2×10^{-2}. So the numerical solution on the 256×256 mesh is well resolved by the mesh and we can consider it as the reference solution. To implement our method, the coarse meshes are chosen to be 8×8, 16×16 and 32×32 respectively, and we compare the results on different meshes. As we mentioned, the forcing terms should be resolved by the coarse mesh, so we choose $f(x,y) \in \{\sin(k_i \pi x + l_i)\cos(m_i \pi y + n_i)\}_{i \in \mathfrak{J}}$, where k_i, l_i, m_i, and n_i are random numbers uniformly distributed over the interval $[0, 0.5]$. We also choose the FEM as the benchmark and compare our method with it.

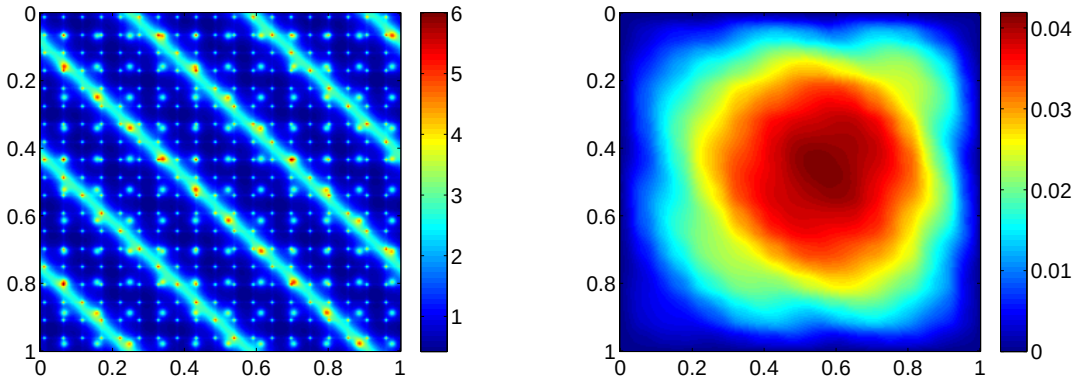
Example 2.1. We consider the case when the elliptic coefficient is a scalar defined by

$$\begin{aligned} a(x,y) =& \frac{1}{2 + 1.6\sin(2\pi(x-y)/\epsilon_1)} + \frac{1}{4 + 1.8(\sin(2\pi x/\epsilon_2) + \sin(2\pi y/\epsilon_2))} \\ &+ \frac{1}{10(2 + 1.8\sin(2\pi(x-0.5)/\epsilon_3))(2 + 1.8\sin(2\pi(y-0.5)/\epsilon_3))}, \end{aligned}$$

where $\epsilon_1 = \frac{1}{3}$, $\epsilon_2 = \frac{1}{11}$, $\epsilon_3 = \frac{1}{19}$.

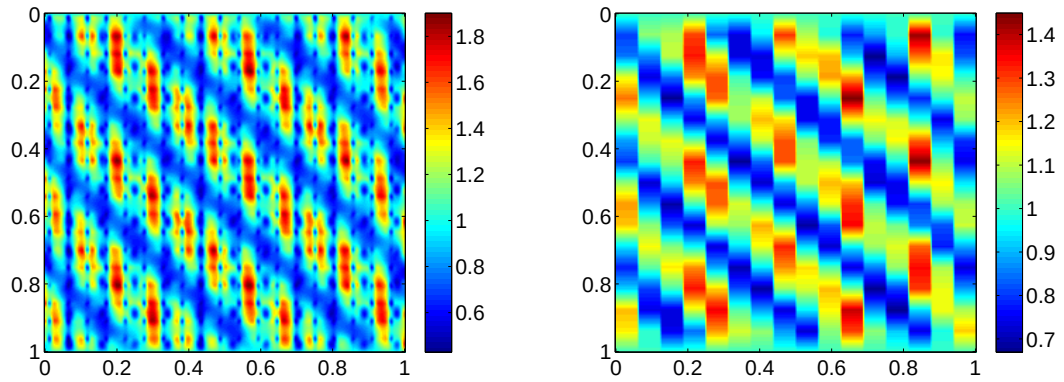
In this example, $f(x,y) = \sin(0.48\pi x + 0.17)\cos(0.29\pi y + 0.11)$. In Figure 2.1-2.2, we plot the coefficient and the decomposition. Relative errors are shown in Tables 2.1 and 2.2. From these figures, we can see that the coefficient oscillates very rapidly, which generates small scale features in the solution (e.g. $\frac{\partial F_1}{\partial x_1}$). The smooth part of F, g, is a summation of some piecewise linear nodal functions, see Figure 2.2. The magnitude of χ is indeed small (around 10^{-2}), see Table 2.4. Thus, the conditions of

Theorem 2.1 are satisfied. We observe convergence in both L^2 and H^1 norms.



(a) a (b) u_e

Figure 2.1: Example 2.1 - The coefficient a and the exact solution u_e



(a) $\frac{\partial F_1}{\partial x_1}$ $(N_c = 16)$ (b) $\frac{\partial g_1}{\partial x_1}$ $(N_c = 16)$

Figure 2.2: Example 2.1 - The derivative of the function F and g $(N_c = 16)$

Remark 2.4. We remark that Theorem 2.1 does not give a specific rate of convergence. It is worthwhile to make the following observations on the convergence property of our method. Denote the exact solution as u_e, the solution constructed from the effective equation as u_m and its numerical approximation as u_n. Then the error consists of two parts, i.e. $||u_e - u_m|| + ||u_m - u_n||$. The first part is controlled by Theorem 2.1, and it gets smaller as we use a finer size. For fixed u_m, the second part converges at order $O(h^2)$ (L^2 norm) or order $O(h^1)$ (H^1 norm). However, as the mesh size h varies, we have different decompositions, so u_m is not fixed. Thus,

Table 2.1: Example 2.1 - L^2 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	2.58×10^{-2}	1.16×10^{-1}
$N_c = 16$	7.27×10^{-3}	6.60×10^{-2}
$N_c = 32$	3.45×10^{-3}	3.58×10^{-2}

Table 2.2: Example 2.1 - H^1 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	1.67×10^{-1}	3.31×10^{-1}
$N_c = 16$	8.26×10^{-2}	2.44×10^{-1}
$N_c = 32$	4.00×10^{-2}	1.74×10^{-1}

the overall rate of convergence is not necessarily $O(h^2)$ or $O(h^1)$. In all our numerical experiments, we observe different rates of convergence for different mesh sizes, especially for the L^2 norm. On the finest mesh, the error will eventually converge to zero (if we take the numerical solution on the finest mesh to be the true solution). However, we want to perform our method on a coarse mesh instead of a fine one. Our main objective is not to find the optimal convergence rate, but to reduce the error on a given coarse mesh. So we are more interested in the error itself rather than the convergence rate. It is important not to be confused with these two issues.

Example 2.2. We choose an anisotropic field a as $a(x, y) = |\theta(x, y)| + 0.5$. Here $\theta(x, y)$ is defined on a 23×23 grid over the domain D, and for each grid cell, the value of $\theta(x, y)$ is distributed according to the standard normal distribution (see Figure 2.3a). The multiscale coefficient, a, is very rough and does not satisfy scale separation or have any periodic structure. Compared with the first example, both the coefficient and the solution are more singular. This presents a challenging test problem for our method. In this example, $f(x, y) = \sin(0.33\pi x + 0.43)\cos(0.43\pi y + 0.38)$. As we can see from the error study presented in Table 2.5-2.6, our method still gives a satisfactory convergence rate and the relative errors are quite small.

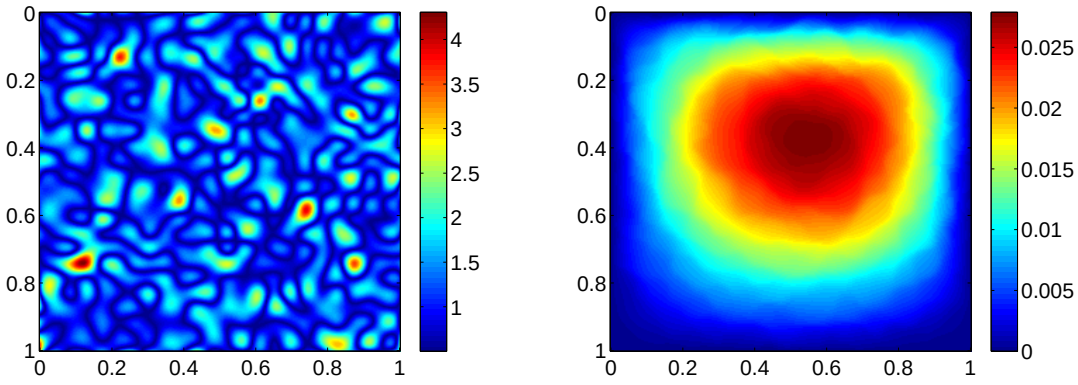
Example 2.3. Next, we consider an example that has a discontinuous and high

Table 2.3: Example 2.1 - L^∞ norm of the function χ

	χ_1	χ_2
$N_c = 8$	1.94×10^{-2}	1.95×10^{-2}
$N_c = 16$	1.30×10^{-2}	1.26×10^{-2}
$N_c = 32$	7.31×10^{-3}	7.12×10^{-3}

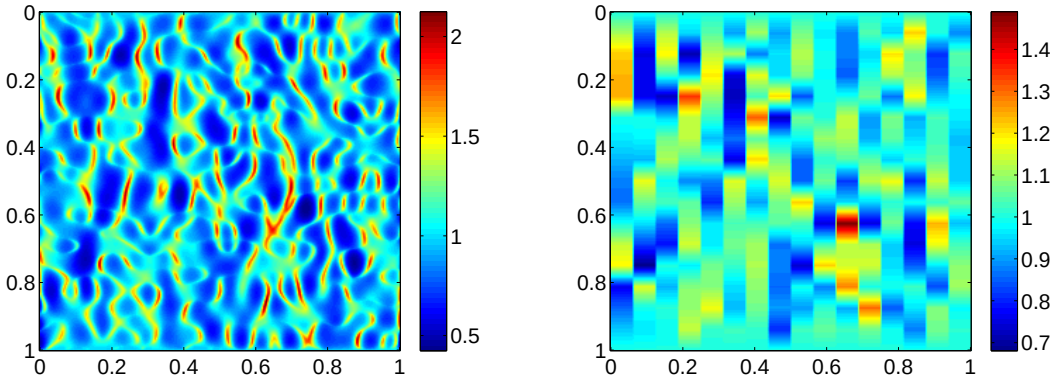
Table 2.4: Example 2.1 - L^2 norm and H^1 norm relative errors of the function g ($N_c = 16$)

	L^2 norm	H^1 norm
g_1 ($N_c = 16$)	2.23×10^{-4}	3.78×10^{-3}
g_2 ($N_c = 16$)	2.31×10^{-4}	3.89×10^{-3}



(a) a (b) u_e

Figure 2.3: Example 2.2 - The coefficient a and the exact solution u_e



(a) $\frac{\partial F_1}{\partial x_1}$ ($N_c = 16$) (b) $\frac{\partial g_1}{\partial x_1}$ ($N_c = 16$)

Figure 2.4: Example 2.2 - The derivative of the function F and g ($N_c = 16$)

Table 2.5: Example 2.2 - L^2 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	3.02×10^{-2}	1.06×10^{-1}
$N_c = 16$	7.33×10^{-3}	6.89×10^{-2}
$N_c = 32$	3.50×10^{-3}	3.89×10^{-2}

Table 2.6: Example 2.2 - H^1 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	1.66×10^{-1}	3.38×10^{-1}
$N_c = 16$	8.47×10^{-2}	2.67×10^{-1}
$N_c = 32$	3.96×10^{-2}	1.95×10^{-1}

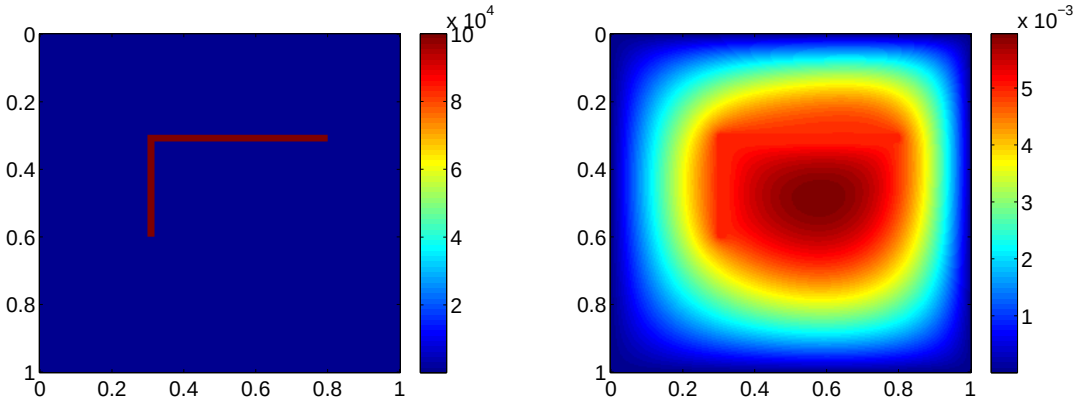
Table 2.7: Example 2.2 - L^∞ norm of the function χ

	χ_1	χ_2
$N_c = 8$	2.42×10^{-2}	2.40×10^{-2}
$N_c = 16$	1.52×10^{-2}	1.36×10^{-2}
$N_c = 32$	7.20×10^{-3}	7.56×10^{-3}

Table 2.8: Example 2.2 - L^2 norm and H^1 norm relative errors of the function g ($N_c = 16$)

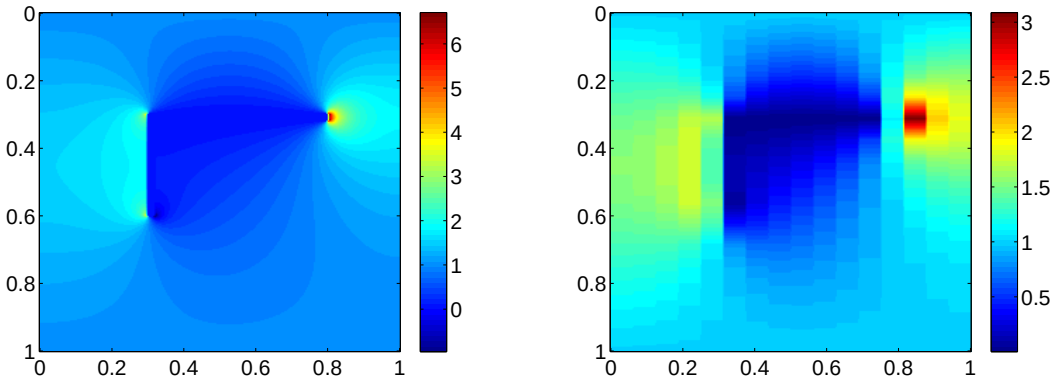
	L^2 norm	H^1 norm
g_1 ($N_c = 16$)	1.86×10^{-4}	4.02×10^{-3}
g_2 ($N_c = 16$)	2.11×10^{-4}	4.49×10^{-3}

contrast coefficient (see Figure 2.5a). The contrast in the coefficient is as high as 10^5. The channel is 0.02 wide in both x and y directions, and 0.5 long in x direction and 0.3 long in y direction. Inside the channel, the coefficient is very large ($= 10^5$), while the coefficient is small outside the channel ($= 1$). There has been a lot of interest in studying multiscale problems with high contrast coefficients in recent years, see e.g. [18, 23, 22, 31]. This is known to be a very difficult problem. In this example, $f(x, y) = \sin(0.33\pi x + 0.43)\cos(0.43\pi y + 0.38)$. Even for such a challenging test problem, our method still captures the important feature of the solution accurately. As we can see from Table 2.9-2.10, the convergence rate remains to be very robust and the relative errors are very small.



(a) a (b) u_e

Figure 2.5: Example 2.3 - The coefficient a and the exact solution u_e



(a) $\frac{\partial F_1}{\partial x_1}$ $(N_c = 16)$ (b) $\frac{\partial g_1}{\partial x_1}$ $(N_c = 16)$

Figure 2.6: Example 2.3 - The derivative of the function F and g $(N_c = 16)$

Table 2.9: Example 2.3 - L^2 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	3.88×10^{-2}	1.30×10^{-1}
$N_c = 16$	9.06×10^{-3}	8.84×10^{-2}
$N_c = 32$	3.99×10^{-3}	3.59×10^{-2}

Table 2.10: Example 2.3 - H^1 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	2.09×10^{-1}	3.04×10^{-1}
$N_c = 16$	8.23×10^{-2}	2.27×10^{-1}
$N_c = 32$	3.96×10^{-2}	1.34×10^{-1}

Table 2.11: Example 2.3 - L^∞ norm of the function χ

	χ_1	χ_2
$N_c = 8$	1.63×10^{-1}	1.18×10^{-1}
$N_c = 16$	7.26×10^{-2}	6.58×10^{-2}
$N_c = 32$	5.80×10^{-2}	3.64×10^{-2}

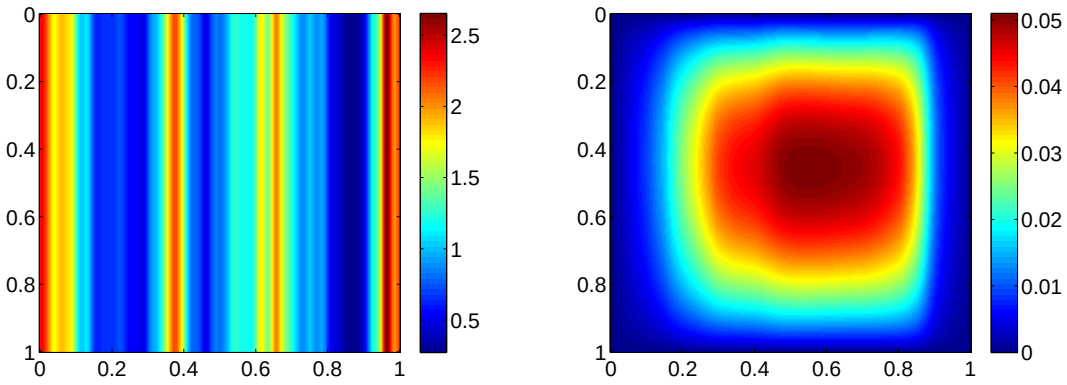
Table 2.12: Example 2.3 - L^2 norm and H^1 norm relative errors of the function g $(N_c = 16)$

	L^2 norm	H^1 norm
g_1 $(N_c = 16)$	1.60×10^{-3}	1.04×10^{-2}
g_2 $(N_c = 16)$	1.11×10^{-3}	6.82×10^{-3}

Example 2.4. We consider the elliptic PDE with the following coefficient

$$a(x,y) = \exp\left(\sum_{i=1}^{10} w_i(\alpha_i \sin(2\pi x/w_i) + \beta_i \cos(2\pi x/w_i))\right),$$

where α_i and β_i are independent uniform random variables in $[-\sqrt{3}, \sqrt{3}]$, and $(w_1, ..., w_{10}) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{11}, \frac{1}{13}, \frac{1}{17}, \frac{1}{19}, \frac{1}{23}, \frac{1}{29}\right)$. As we can see from Figure 2.7a, the coefficient varies rapidly in x direction, and it has many layers, which makes the problem very difficult to solve. In this example, $f(x,y) = \sin(0.44\pi x + 0.46)\cos(0.28\pi y + 0.26)$. For such a problem, our method works very well and captures the small scales in x direction. As we can see from Table 2.13-2.14, the relative errors are very small. We also note that there is no variation in y direction, so the norm of χ_2 and the numerical errors for g_2 are 0, as expected (Table 2.15-2.16).



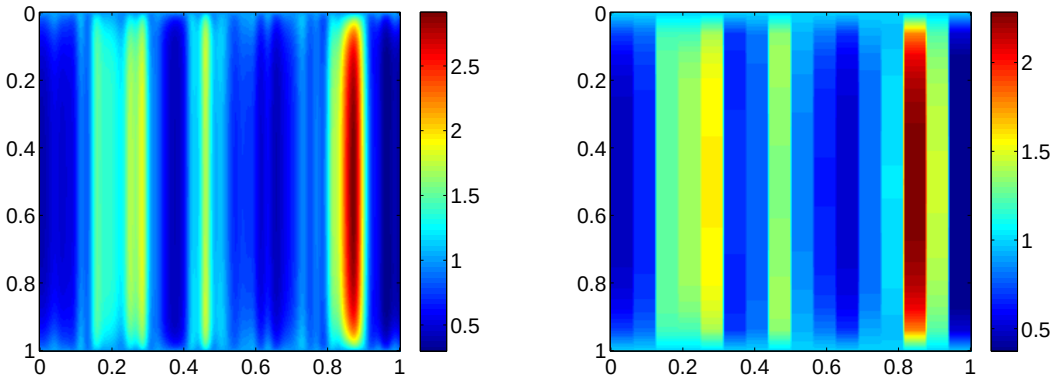
(a) a (b) u_e

Figure 2.7: Example 2.4 - The coefficient a and the exact solution u_e

Table 2.13: Example 2.4 - L^2 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	2.43×10^{-2}	1.35×10^{-1}
$N_c = 16$	7.29×10^{-3}	4.91×10^{-2}
$N_c = 32$	3.03×10^{-3}	1.18×10^{-2}

Example 2.5. Finally, we consider an elliptic problem that also has a discontinuous coefficient with both channels and many small inclusions (see Figure 2.9a). Inside these inclusions and channels, the coefficient is 50, while outside them the coefficient is 1. This is a even harder problem than Example 2.3, since the discontinuities occur



(a) $\frac{\partial F_1}{\partial x_1}$ $(N_c = 16)$ (b) $\frac{\partial g_1}{\partial x_1}$ $(N_c = 16)$

Figure 2.8: Example 2.4 - The derivative of the function F and g $(N_c = 16)$

Table 2.14: Example 2.4 - H^1 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	1.95×10^{-1}	3.90×10^{-1}
$N_c = 16$	9.22×10^{-2}	2.36×10^{-1}
$N_c = 32$	4.15×10^{-2}	1.01×10^{-1}

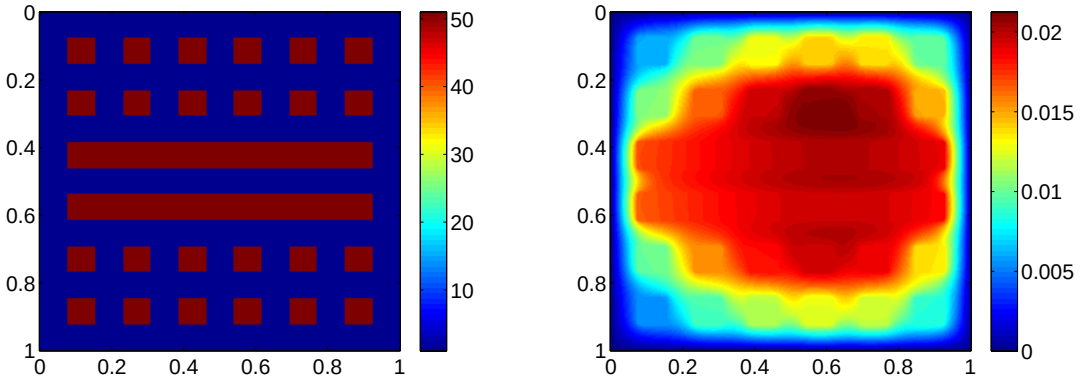
Table 2.15: Example 2.4 - L^∞ norm of the function χ

	χ_1	χ_2
$N_c = 8$	3.95×10^{-2}	0
$N_c = 16$	2.17×10^{-2}	0
$N_c = 32$	6.74×10^{-3}	0

Table 2.16: Example 2.4 - L^2 norm and H^1 norm relative errors of the function g $(N_c = 16)$

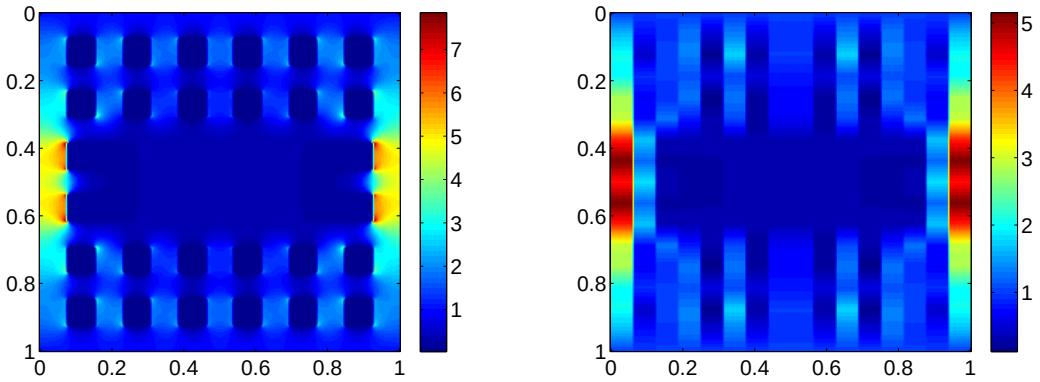
	L^2 norm	H^1 norm
g_1 $(N_c = 16)$	7.55×10^{-4}	7.95×10^{-3}
g_2 $(N_c = 16)$	0	0

in many regions. In this example, $f(x,y) = \sin(0.35\pi x + 0.23)\cos(0.21\pi y + 0.06)$. As we can see from Table 2.17-2.18, our method still provides satisfactory results and the errors are small, which shows that our method can be used to solve challenging multiscale problems without scale separation and with discontinuous coefficients.



(a) a (b) u_e

Figure 2.9: Example 2.5 - The coefficient a and the exact solution u_e



(a) $\frac{\partial F_1}{\partial x_1}$ $(N_c = 16)$ (b) $\frac{\partial g_1}{\partial x_1}$ $(N_c = 16)$

Figure 2.10: Example 2.5 - The derivative of the function F and g $(N_c = 16)$

Table 2.17: Example 2.5 - L^2 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	7.29×10^{-2}	9.84×10^{-1}
$N_c = 16$	3.68×10^{-2}	3.74×10^{-1}
$N_c = 32$	1.12×10^{-2}	1.70×10^{-1}

Table 2.18: Example 2.5 - H^1 norm relative errors of the solution

	MsMRM	FEM
$N_c = 8$	1.78×10^{-1}	9.90×10^{-1}
$N_c = 16$	9.16×10^{-2}	5.38×10^{-1}
$N_c = 32$	6.28×10^{-2}	3.56×10^{-1}

Table 2.19: Example 2.5 - L^∞ norm of the function χ

	χ_1	χ_2
$N_c = 8$	2.11×10^{-1}	6.44×10^{-2}
$N_c = 16$	1.14×10^{-1}	3.46×10^{-2}
$N_c = 32$	8.68×10^{-2}	2.19×10^{-2}

Table 2.20: Example 2.5 - L^2 norm and H^1 norm relative errors of the function g ($N_c = 16$)

	L^2 norm	H^1 norm
g_1 ($N_c = 16$)	1.43×10^{-3}	1.03×10^{-2}
g_2 ($N_c = 16$)	1.59×10^{-3}	1.45×10^{-2}

Chapter 3

Multiscale model reduction method for time-dependent PDEs

3.1 Effective equations

We could apply a similar idea to derive effective equations for time-dependent equations with time-independent coefficients.

3.1.1 Parabolic equation

We first consider a parabolic equation of the form

$$\begin{cases} u_t(x,t) - \nabla \cdot (a(x)\nabla u(x,t)) = f(x), & x \in D, t \in (0,T], \\ u(x,t) = 0, & x \in \partial D, t \in (0,T], \\ u(x,0) = 0, & x \in D. \end{cases} \tag{3.1}$$

We can define the harmonic coordinates F in exactly the same way as we did for the elliptic equation. We then decompose $F = g + \chi$ and solve the following effective equation on a coarse mesh

$$\begin{cases} (u_0)_t(x,t) - \nabla \cdot (a(x)(I + \frac{\partial \chi}{\partial x}\frac{\partial x}{\partial g})\nabla u_0(x,t)) = f(x), & x \in D, t \in (0,T], \\ u_0(x,t) = 0, & x \in \partial D, t \in (0,T], \\ u_0(x,0) = 0, & x \in D. \end{cases} \tag{3.2}$$

Again, we approximate u by $u_0 + \chi^T \frac{\partial x}{\partial g}\nabla u_0$.

3.1.2 Convection-diffusion equation

Next, we consider the convection-diffusion equation with multiscale velocity field (see also [47])

$$\begin{cases} v_t(x,t) + u(x)\cdot\nabla v(x,t) = \nabla\cdot(\alpha\nabla v(x,t)) & x\in D, t\in(0,T], \\ v(x,t) = 0, & x\in\partial D, t\in(0,T], \\ v(x,0) = \phi(x), & x\in D, \end{cases} \tag{3.3}$$

where $u(x)$ is a velocity field that satisfies $\nabla\cdot u(x) = 0$ and α is a positive diffusion constant.

We define the corresponding harmonic coordinates as follows

$$\begin{cases} u(x)\cdot\nabla F_k(x) = \nabla\cdot(\alpha\nabla F_k(x)), & x\in D \\ F_k(x) = x_k, & x\in\partial D. \end{cases} \tag{3.4}$$

By decomposing $F = g + \chi$ as before, we obtain the following effective equation

$$\begin{cases} (v_0)_t(x,t) + u(x)\cdot\nabla v_0(x,t) = \nabla\cdot((\alpha\frac{\partial F}{\partial x} - u\chi^T)\frac{\partial x}{\partial g}\nabla v_0(x,t)), & x\in D, t\in(0,T], \\ \text{or equivalently} \\ (v_0)_t = \sum_{i,j,k=1}^{n}(\alpha\frac{\partial F}{\partial x} - u\chi^T)_{ij}\frac{\partial g_k}{\partial x_i}\frac{\partial^2 v_0}{\partial g_j\partial g_k} & \text{in } D\times(0,T] \\ v_0(x,t) = 0, & x\in\partial D, t\in(0,T], \\ v_0(x,0) = \phi(x), & x\in D. \end{cases} \tag{3.5}$$

Finally, v is approximated by $v_0 + \chi^T\frac{\partial x}{\partial g}\nabla v_0$.

3.1.3 Hyperbolic equation

The multiscale model reduction method proposed in this paper can be extended to study hyperbolic partial differential equations with multiscale coefficients. Specifically,

we consider the following hyperbolic equation

$$\begin{cases} u_{tt}(x,t) - \nabla \cdot (a(x)\nabla u(x,t)) = f(x), & x \in D, t \in (0,T], \\ u = 0, & x \in \partial D, t \in (0,T], \\ u = 0, & x \in D, \\ u_t = 0, & x \in D. \end{cases} \tag{3.6}$$

It is straightforward to generalize the derivation of our effective equation to the hyperbolic equation. The effective equation takes the form

$$\begin{cases} (u_0)_{tt}(x,t) - \nabla \cdot (a(x)(I + \frac{\partial \chi}{\partial x}\frac{\partial x}{\partial g})\nabla u_0(x,t)) = f(x), & x \in D, t \in (0,T], \\ u_0(x,t) = 0, & x \in \partial D, t \in (0,T], \\ u_0(x,0) = 0, & x \in D, \\ (u_0)_t(x,0) = 0, & x \in D, \end{cases} \tag{3.7}$$

where $F = g + \chi$ is defined in the same way as before.

3.2 Error estimate

For the parabolic equation, we have the following theorem.

Theorem 3.1. *Suppose u, F and u_0 are weak solutions to* (3.1), (2.2) *and* (3.2) *respectively. Let $u_1 = \chi^T \frac{\partial x}{\partial g}\nabla u_0$, $F = g + \chi$, and $\chi = 0$ on ∂D. Then we have*

$$\begin{aligned} &\max_{(0,T]} \|u - u_0 - u_1\|_{L^2(D)} + \|u - u_0 - u_1\|_{L^2(0,T;H^1(D))} \\ \leq & C\|\chi\|_{L^\infty(D)}\|\frac{\partial g}{\partial x}\|_{L^\infty(D)}\|\det(\frac{\partial x}{\partial g})\|_{L^\infty(D)}(|\tilde{u}_0|_{H^{2,1}(0,T;D)} + \|(\tilde{u}_0)_t\|_{L^2(0,T;H^1(D))}), \end{aligned} \tag{3.8}$$

where C is a constant that depends on d, D and a, $\tilde{u}_0(y,t) = \tilde{u}_0(g(x),t) = u_0(x,t)$ and $\|u\|^2_{L^2(0,T;H^1(D))} := \int_0^T \int_D (u^2 + |\nabla u|^2)\mathrm{d}x\mathrm{d}t$, $|u|^2_{H^{2,1}(0,T;D)} := \int_0^T \int_D ((u_t)^2 + |\nabla\nabla u|^2)\mathrm{d}x\mathrm{d}t$.

Proof. Let $z = u - u_0 - u_1$, then $z = 0$ at $t = 0$ and on ∂D. We define

$$p = a\nabla u - a\frac{\partial F}{\partial x}\frac{\partial x}{\partial g}\nabla u_0,$$

$$\eta_1 = -a\frac{\partial}{\partial x}\left(\frac{\partial x}{\partial g}\nabla u_0\right)\chi,$$

$$\eta_2 = -\chi^T\frac{\partial x}{\partial g}\nabla(u_0)_t.$$

Then we have

$$\begin{aligned} a\nabla z - p =& a\nabla u - a\nabla u_0 - a\nabla u_1 - a\nabla u + a\frac{\partial F}{\partial x}\frac{\partial x}{\partial g}\nabla u_0 \\ =& - a\nabla u_0 - a\nabla u_1 + a\frac{\partial F}{\partial x}\frac{\partial x}{\partial g}\nabla u_0 \\ =& - a\frac{\partial}{\partial x}\left(\frac{\partial x}{\partial g}\nabla u_0\right)\chi = \eta_1, \end{aligned}$$

and

$$\nabla \cdot p = (u_t - f) - ((u_0)_t - f) = u_t - (u_0)_t = z_t + (u_1)_t = z_t - \eta_2.$$

Then for any $\tau \in (0, T]$ we have

$$\begin{aligned} & \int_0^\tau \int_D \nabla z \cdot a\nabla z \mathrm{d}x\mathrm{d}t \\ =& \int_0^\tau \int_D \nabla z \cdot (a\nabla z - p)\mathrm{d}x\mathrm{d}t + \int_0^\tau \int_D \nabla z \cdot p\mathrm{d}x\mathrm{d}t \\ =& \int_0^\tau \int_D \nabla z \cdot \eta_1 \mathrm{d}x\mathrm{d}t - \int_0^\tau \int_D z(\nabla \cdot p)\mathrm{d}x\mathrm{d}t \\ =& \int_0^\tau \int_D \nabla z \cdot \eta_1 \mathrm{d}x\mathrm{d}t - \int_0^\tau \int_D (zz_t - z\eta_2)\mathrm{d}x\mathrm{d}t \\ =& \int_0^\tau \int_D \nabla z \cdot \eta_1 \mathrm{d}x\mathrm{d}t - \int_D \left(\frac{z^2(\tau, x)}{2} - \frac{z^2(0, x)}{2}\right)\mathrm{d}x + \int_0^\tau \int_D z\eta_2 \mathrm{d}x\mathrm{d}t \\ =& \int_0^\tau \int_D \nabla z \cdot \eta_1 \mathrm{d}x\mathrm{d}t - \frac{1}{2}\int_D z^2(\tau, x)\mathrm{d}x + \int_0^\tau \int_D z\eta_2 \mathrm{d}x\mathrm{d}t. \end{aligned}$$

Rearranging the above equation, we get

$$\begin{aligned}
&\frac{1}{2}\int_D z^2(\tau,x)\mathrm{d}x + \int_0^\tau\int_D \nabla z\cdot a\nabla z\mathrm{d}x\mathrm{d}t\\
=&\int_0^\tau\int_D \nabla z\cdot\eta_1\mathrm{d}x\mathrm{d}t + \int_0^\tau\int_D z\eta_2\mathrm{d}x\mathrm{d}t\\
\le&\int_0^\tau\int_D |\nabla z\cdot\eta_1|\mathrm{d}x\mathrm{d}t + \int_0^\tau\int_D |z\eta_2|\mathrm{d}x\mathrm{d}t\\
\le&\int_0^T\int_D |\nabla z\cdot\eta_1|\mathrm{d}x\mathrm{d}t + \int_0^T\int_D |z\eta_2|\mathrm{d}x\mathrm{d}t.
\end{aligned}$$

Taking the maximum over $\tau\in(0,T]$, we have

$$\begin{aligned}
&\max_{(0,T]}\frac{1}{2}\int_D z^2(\tau,x)\mathrm{d}x + \int_0^T\int_D \nabla z\cdot a\nabla z\mathrm{d}x\mathrm{d}t\\
\le&\int_0^T\int_D |\nabla z\cdot\eta_1|\mathrm{d}x\mathrm{d}t + \int_0^T\int_D |z\eta_2|\mathrm{d}x\mathrm{d}t\\
\le&\|\nabla z\|_{L^2(0,T;D)}\|\eta_1\|_{L^2(0,T;D)} + \|z\|_{L^2(0,T;D)}\|\eta_2\|_{L^2(0,T;D)}.
\end{aligned}$$

where the second inequality is due to the Hölder's inequality.

Application of the Poincaré's inequality gives

$$\begin{aligned}
&\Big(\max_{(0,T]}\|z\|_{L^2(D)} + \|z\|_{L^2(0,T;H^1(D))}\Big)^2\\
\le&C\Big(\max_{(0,T]}\frac{1}{2}\int_D z^2(\tau,x)\mathrm{d}x + \int_0^T\int_D \nabla z\cdot a\nabla z\mathrm{d}x\mathrm{d}t\Big)\\
\le&C\Big(\|\nabla z\|_{L^2(0,T;D)}\|\eta_1\|_{L^2(0,T;D)} + \|z\|_{L^2(0,T;D)}\|\eta_2\|_{L^2(0,T;D)}\Big)\\
\le&C\Big(\max_{(0,T]}\|z\|_{L^2(D)} + \|z\|_{L^2(0,T;H^1(D))}\Big)\Big(\|\eta_1\|_{L^2(0,T;D)} + \|\eta_2\|_{L^2(0,T;D)}\Big).
\end{aligned}$$

Finally we have

$$\begin{aligned}
&\max_{(0,T]}\|z\|_{L^2(D)} + \|z\|_{L^2(0,T;H^1(D))}\\
\le&C\Big(\|\eta_1\|_{L^2(0,T;D)} + \|\eta_2\|_{L^2(0,T;D)}\Big)\\
\le&C\|\chi\|_{L^\infty(D)}\|\frac{\partial g}{\partial x}\|_{L^\infty(D)}\|\det(\frac{\partial x}{\partial g})\|_{L^\infty(D)}(|\tilde{u}_0|_{H^{2,1}(0,T;D)} + \|(\tilde{u}_0)_t\|_{L^2(0,T;H^1(D))}),
\end{aligned}$$

where C is a constant that depends on d, D and a. □

For the convection-diffusion equation, we have the following theorem.

Theorem 3.2. *Suppose v, F and v_0 are weak solutions to* (3.3), (3.4) *and* (3.5) *respectively. Let $v_1 = \chi^T \frac{\partial x}{\partial g} \nabla v_0$, $F = g + \chi$, and $\chi = 0$ on ∂D. Then we have*

$$\begin{aligned}&\max_{(0,T]} \|v - v_0 - v_1\|_{L^2(D)} + \|v - v_0 - v_1\|_{L^2(0,T;H^1(D))} \\ \leq & C\|\chi\|_{L^\infty(D)} \|\frac{\partial g}{\partial x}\|_{L^\infty(D)} \|\det(\frac{\partial x}{\partial g})\|_{L^\infty(D)} (|\tilde{v}_0|_{H^{2,1}(0,T;D)} + \|(\tilde{v}_0)_t\|_{L^2(0,T;H^1(D))}), \end{aligned} \tag{3.9}$$

where C is a constant that depends on d, D, u, and α, $\tilde{v}_0(y,t) = \tilde{v}_0(g(x),t) = v_0(x,t)$.

The proof of the above theorem is analogous to that of Theorem 3.1. We omit the proof here.

In Theorems 3.1 and 3.2, the quantities $|\tilde{u}_0|_{H^{2,1}(0,T;D)}$ and $|\tilde{v}_0|_{H^{2,1}(0,T;D)}$ could be bounded since the equations in both cases can be written in non-divergence forms in variable $y = g(x)$.

For the parabolic equation, the equation has a form similar to that of the elliptic equation

$$(\tilde{u}_0)_t - \sum_{i,j=1}^{n} B_{ij}(y) \frac{\partial^2 \tilde{u}_0}{\partial y_i \partial y_j} = \tilde{f}(y), \tag{3.10}$$

where B and $\tilde{f}$ are the same as those in (2.22) and (2.23).

For the convection-diffusion equation, the effective equation is

$$(v_0)_t + u \cdot \nabla v_0 = \nabla \cdot \left((\alpha \frac{\partial F}{\partial x} - u\chi^T) \frac{\partial x}{\partial g} \nabla v_0 \right). \tag{3.11}$$

Note that

$$\begin{aligned}
&\nabla\cdot((\alpha\frac{\partial F}{\partial x}-u\chi^T)\frac{\partial x}{\partial g}\nabla v_0)\\
=&\nabla\cdot((\alpha\frac{\partial F}{\partial x}-u\chi^T))\frac{\partial x}{\partial g}\nabla v_0+\sum_{i,j,k=1}^{n}(\alpha\frac{\partial F}{\partial x}-u\chi^T)_{ij}\frac{\partial g_k}{\partial x_i}\frac{\partial^2 v_0}{\partial g_j\partial g_k}\\
=&(\alpha\Delta F-u\cdot\frac{\partial \chi}{\partial x})\frac{\partial x}{\partial g}\nabla v_0+\sum_{i,j,k=1}^{n}(\alpha\frac{\partial F}{\partial x}-u\chi^T)_{ij}\frac{\partial g_k}{\partial x_i}\frac{\partial^2 v_0}{\partial g_j\partial g_k}\\
=&(u\cdot\frac{\partial F}{\partial x}-u\cdot\frac{\partial \chi}{\partial x})\frac{\partial x}{\partial g}\nabla v_0+\sum_{i,j,k=1}^{n}(\alpha\frac{\partial F}{\partial x}-u\chi^T)_{ij}\frac{\partial g_k}{\partial x_i}\frac{\partial^2 v_0}{\partial g_j\partial g_k}\\
=&u\cdot\frac{\partial g}{\partial x}\frac{\partial x}{\partial g}\nabla v_0+\sum_{i,j,k=1}^{n}(\alpha\frac{\partial F}{\partial x}-u\chi^T)_{ij}\frac{\partial g_k}{\partial x_i}\frac{\partial^2 v_0}{\partial g_j\partial g_k}\\
=&u\cdot\nabla v_0+\sum_{i,j,k=1}^{n}(\alpha\frac{\partial F}{\partial x}-u\chi^T)_{ij}\frac{\partial g_k}{\partial x_i}\frac{\partial^2 v_0}{\partial g_j\partial g_k}.
\end{aligned}$$

So the equation can be written as

$$(\tilde{v}_0)_t-\sum_{i,j=1}^{n}B_{ij}(y)\frac{\partial^2\tilde{v}_0}{\partial y_i\partial y_j}=0, \tag{3.12}$$

where

$$B(y)=(|\det(\frac{\partial x}{\partial g})|(\alpha\frac{\partial F}{\partial x}-u\chi^T)\frac{\partial g}{\partial x})\circ g^{-1}(y). \tag{3.13}$$

In both cases, if the corresponding coefficient matrix B satisfies

$$\frac{\sum_{i,j=1}^{n}B_{ij}^2+1}{(\sum_{i=1}^{n}B_{ii}+1)^2}\leq\frac{1}{n+\epsilon}, \tag{3.14}$$

where n is the dimension and ϵ is a positive number, we can prove that $|\tilde{u}_0|_{H^{2,1}(0,T;D)}$ (or$|\tilde{v}_0|_{H^{2,1}(0,T;D)}$) is bounded (see e.g. [46]).

As for the terms $\|(\tilde{u}_0)_t\|_{L^2(0,T;H^1(D))}$ and $\|(\tilde{v}_0)_t\|_{L^2(0,T;H^1(D))}$, we cannot provide an analytical bound based on the assumptions that we have so far. Since these terms contain the spacial gradient of $(\tilde{u}_0)_t$ (or $(\tilde{v}_0)_t$), we need stronger requirements on the coefficients as well as the source terms. We will monitor these terms in the numerical experiments. As we will see in the numerical examples, these terms are bounded in the examples that we consider.

The convergence analysis of the effective equation for the hyperbolic equation is more complicated than that for the equations that we have considered so far. A straightforward generalization of our previous convergence analysis to the hyperbolic equation would require a stronger regularity assumption on the effective solution. However, our numerical results indicate that our effective equation still gives a very accurate approximation to the multiscale solution of the hyperbolic PDE in both L^2 and H^1 norms.

3.3 Numerical results

In all the numerical examples for the time-dependent PDEs, the physical domain is chosen to be $D = 8 \times 8$, and we choose a 256×256 mesh for the well-resolved solution. We compute the solutions on 8×8, 16×16 and 32×32 meshes using our multiscale model reduction method (MsMRM), and compare the results with the traditional finite element method (FEM).

Example 3.1. A parabolic equation. In this example, $a = \frac{2+\sin(2\pi x/\epsilon_1)}{2+\cos(2\pi y/\epsilon_1)} + \frac{2+\cos(2\pi x/\epsilon_2)}{2+\sin(2\pi y/\epsilon_2)}$ with $\epsilon_1 = \frac{1}{7}$, $\epsilon_2 = \frac{1}{19}$ and $f = 1$. We compute the solution until $T = 0.1$. Table 3.1-3.2 show the errors versus time with different coarse grid meshes. As we can see from the tables, our method gives qualitatively the same performance as for the case of the elliptic equation. In Figure 3.3, we show the quantity $\int_\Omega |\nabla(\tilde{u}_0)_t|^2 \mathrm{d}x$ as time varies. Although we do not have an analytical bound for it based on the limited assumptions, we can see that it remains bounded.

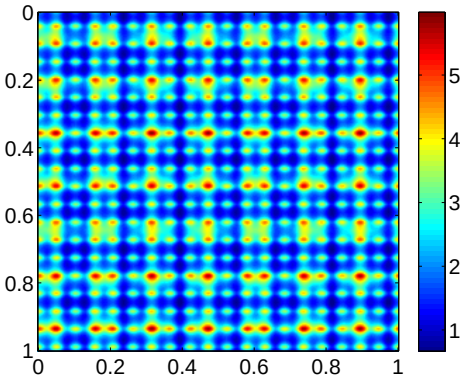
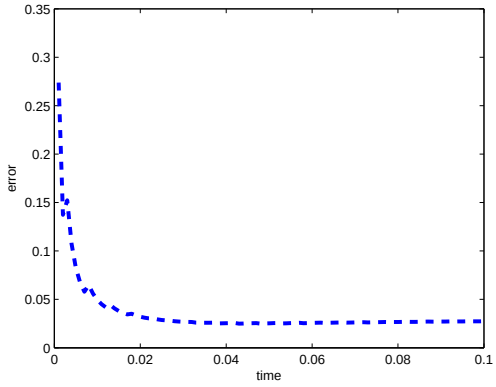
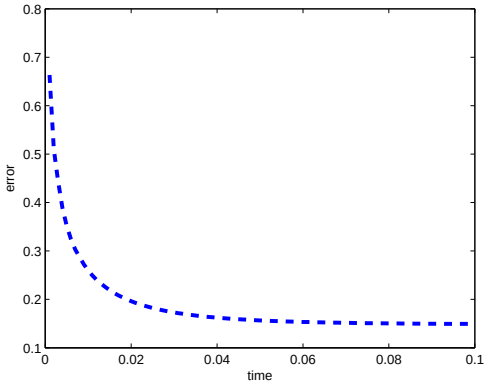


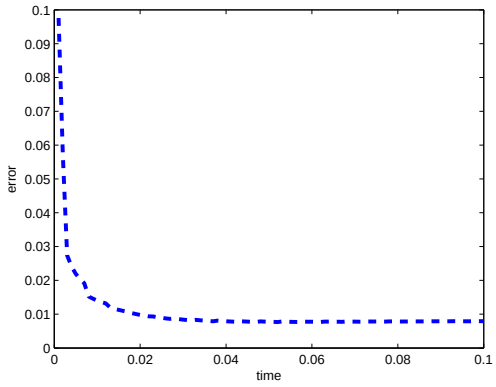
Figure 3.1: Example 3.1 - The coefficient a



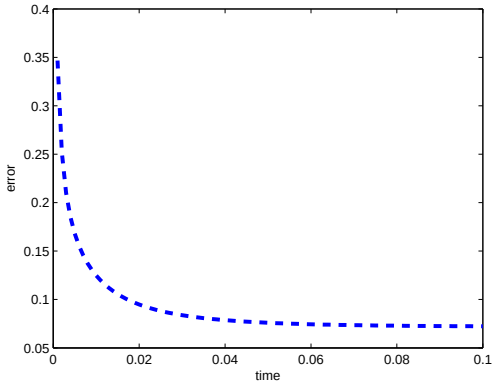
(a) L^2 norm relative errors of the solution at $N_c = 8$



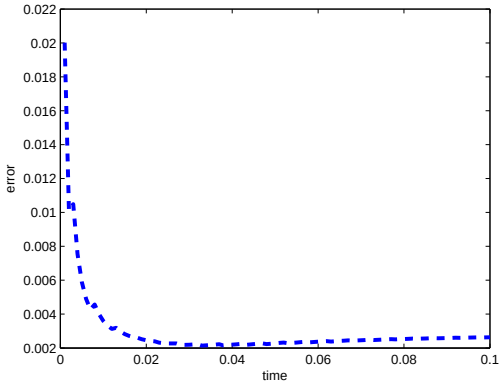
(b) L^2 norm relative errors of the solution at $N_c = 8$



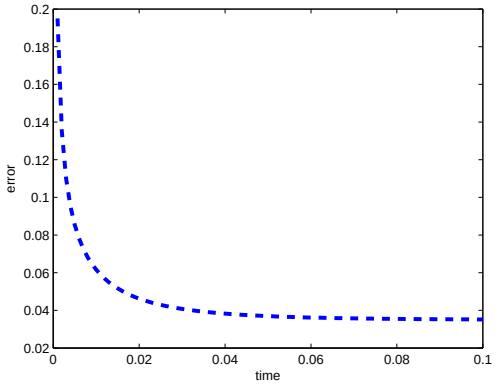
(c) L^2 norm relative errors of the solution at $N_c = 16$



(d) H^1 norm relative errors of the solution at $N_c = 16$



(e) L^2 norm relative errors of the solution at $N_c = 32$



(f) H^1 norm relative errors of the solution at $N_c = 32$

Figure 3.2: Example 3.1 - Relative errors of the solution

Table 3.1: Example 3.1 - L^2 norm relative errors of the solution at $T = 0.1$

	MsMRM	FEM
$N_c = 8$	2.74×10^{-2}	9.79×10^{-2}
$N_c = 16$	7.94×10^{-3}	5.83×10^{-2}
$N_c = 32$	2.63×10^{-3}	3.37×10^{-2}

Table 3.2: Example 3.1 - H^1 norm relative errors of the solution at $T = 0.1$

	MsMRM	FEM
$N_c = 8$	1.49×10^{-1}	3.26×10^{-1}
$N_c = 16$	7.23×10^{-2}	2.48×10^{-1}
$N_c = 32$	3.52×10^{-2}	1.77×10^{-1}

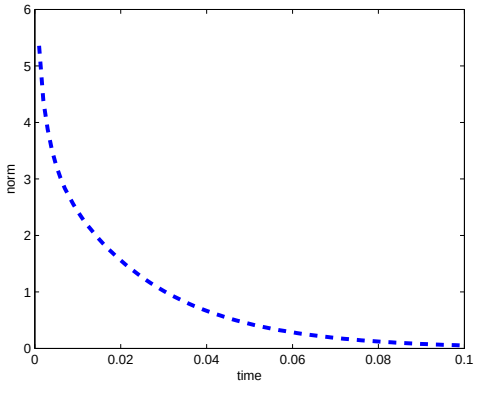
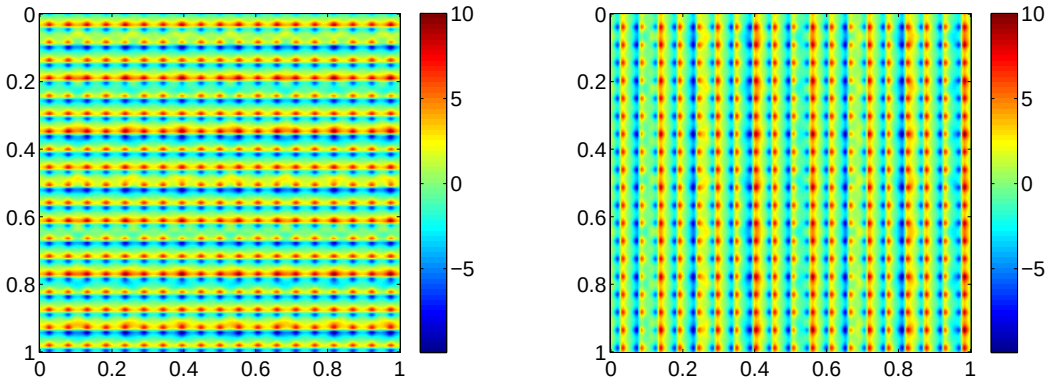


Figure 3.3: Example 3.1 - $\int_\Omega |\nabla(\tilde{u}_0)_t|^2 \mathrm{d}x$ at $N_c = 16$

Example 3.2. A convection-diffusion equation. In this experiment, we choose the stream function $\psi = \frac{1}{(2+\sin(2\pi x/\epsilon_1))(2+\cos(2\pi y/\epsilon_1))} + \frac{1}{(2+\cos(2\pi x/\epsilon_2))(2+\sin(2\pi y/\epsilon_2))}$ with $\epsilon_1 = \frac{1}{7}$ and $\epsilon_2 = \frac{1}{19}$ to generate the velocity filed. $u_0 = (\psi_y, -\psi_x)$ and $u = \frac{10u_0}{||u_0||_{L^\infty}}$. The initial condition is $\phi = xy(1-x)(1-y)$, and $\alpha = 0.05$. The end time is $T = 0.1$. Figure 3.5 shows the errors versus time with different coarse grid meshes. Figure 3.6 shows the boundedness of $\int_\Omega |\nabla(\tilde{v}_0)_t|^2 \mathrm{d}x$. Table 3.3-3.4 shows the error at $T = 0.1$. In this case, we observe that the errors are larger than those presented in the previous examples for the elliptic and the parabolic equations. The reason for this behavior is due to a mild degeneracy of ellipticity since the diffusion coefficient α is relatively small in this convection diffusion problem. From our convergence analysis, the error will be amplified by the smallest eigenvalue of the elliptic coefficient. In this case, the smallest eigenvalue is α, which is smaller than the ones we consider in the previous example. Given that the convection diffusion equation presents a tougher test problem for our method, the performance of our method is still quite encouraging.



(a) u_1 (b) u_2

Figure 3.4: Example 3.2 - The velocity fields u_1 and u_2

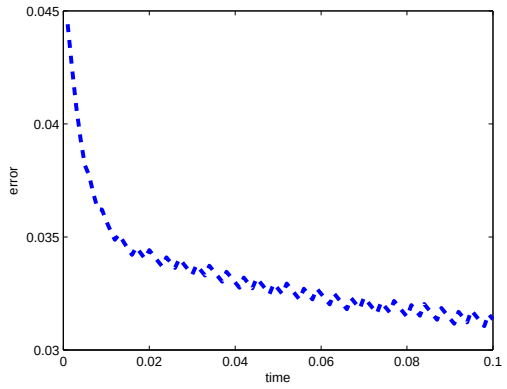
Table 3.3: Example 3.2 - L^2 norm relative errors of the solution at $T = 0.1$

	MsMRM	FEM
$N_c = 8$	3.14×10^{-2}	3.89×10^{-2}
$N_c = 16$	8.51×10^{-3}	2.78×10^{-2}
$N_c = 32$	2.58×10^{-3}	1.62×10^{-2}

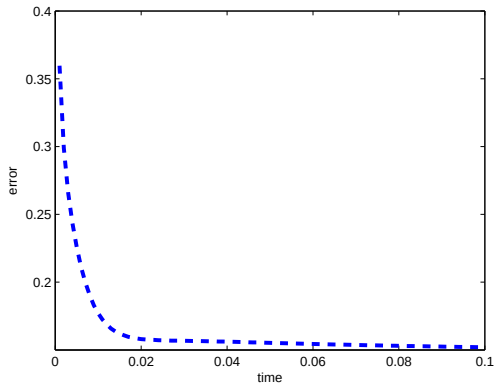
Example 3.3. A hyperbolic equation. In this example, we choose the coefficient as $a = 5 + \sin(2\pi x/\epsilon_1) + \cos(2\pi y/\epsilon_1) + \cos(2\pi x/\epsilon_2) + \sin(2\pi y/\epsilon_2)$, where $\epsilon_1 = \frac{1}{7}$, $\epsilon_2 = \frac{1}{19}$,

Table 3.4: Example 3.2 - H^1 norm relative errors of the solution at $T = 0.1$

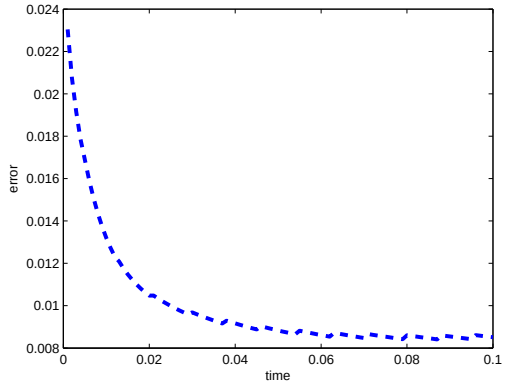
	MsMRM	FEM
$N_c = 8$	1.52×10^{-1}	4.53×10^{-1}
$N_c = 16$	7.04×10^{-2}	3.97×10^{-1}
$N_c = 32$	3.57×10^{-2}	3.12×10^{-1}



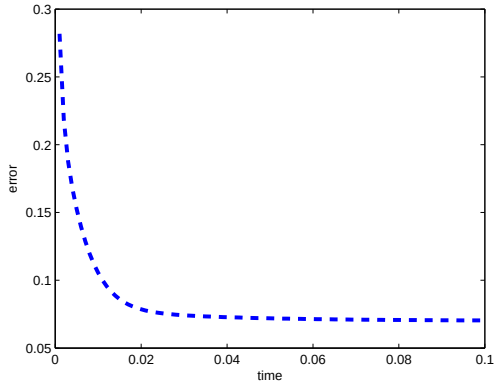
(a) L^2 norm relative errors of the solution at $N_c = 8$



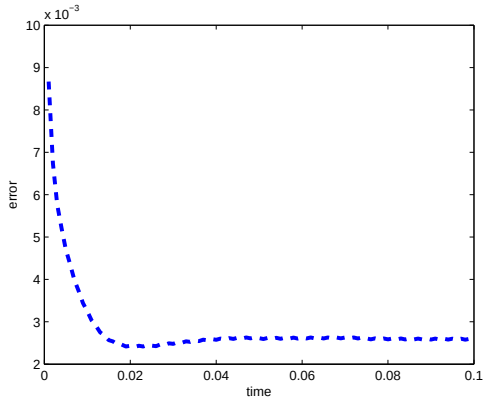
(b) L^2 norm relative errors of the solution at $N_c = 8$



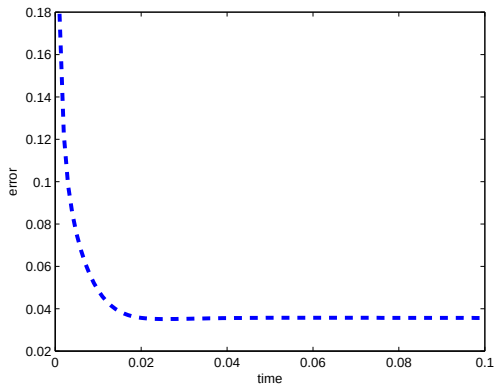
(c) L^2 norm relative errors of the solution at $N_c = 16$



(d) H^1 norm relative errors of the solution at $N_c = 16$



(e) L^2 norm relative errors of the solution at $N_c = 32$



(f) H^1 norm relative errors of the solution at $N_c = 32$

Figure 3.5: Example 3.2 - Relative errors of the solution

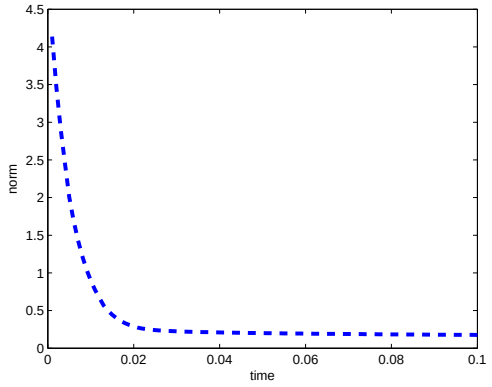


Figure 3.6: Example 3.2 - $\int_\Omega |\nabla(\tilde{v}_0)_t|^2 \mathrm{d}x$ at $N_c = 16$

and $f = 1$. The end time is $T = 0.5$. Figure 3.8 shows the errors versus time with different coarse grid meshes, and Table 3.5-3.6 shows the error at $T = 1$. As we can see from Figure 3.8 and Table 3.5-3.6, our method gives first order convergence in the H^1 norm and better than first order of convergence in the L^2 norm, which is consistent with the convergence rates that we observed earlier for elliptic and parabolic equations.

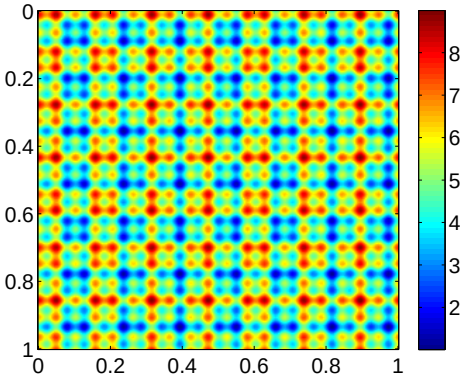


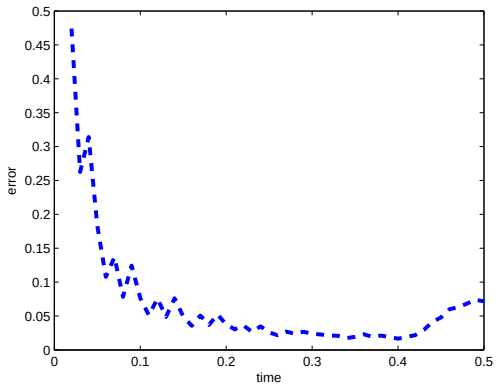
Figure 3.7: Example 3.3 - The coefficient a

Table 3.5: Example 3.3 - L^2 norm relative errors of the solution at $T = 0.5$

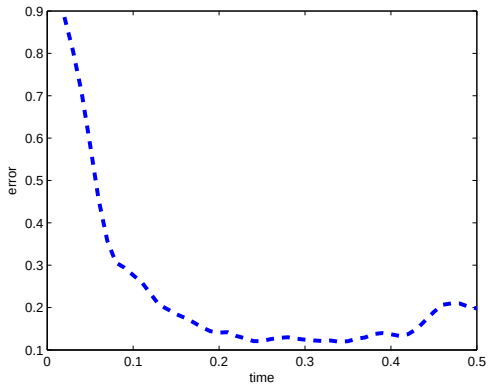
	MsMRM	FEM
$N_c = 8$	7.17×10^{-2}	1.67×10^{-2}
$N_c = 16$	2.93×10^{-2}	1.25×10^{-1}
$N_c = 32$	7.91×10^{-3}	7.60×10^{-2}

Table 3.6: Example 3.3 - H^1 norm relative errors of the solution at $T = 0.5$

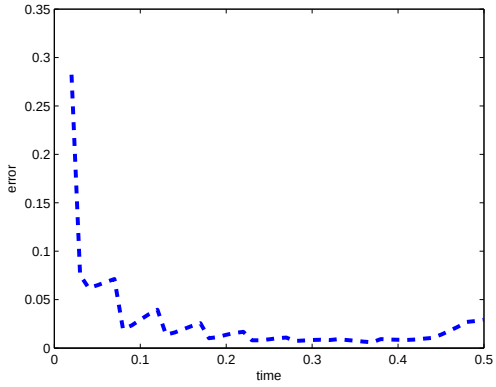
	MsMRM	FEM
$N_c = 8$	1.98×10^{-1}	3.12×10^{-1}
$N_c = 16$	1.21×10^{-1}	2.57×10^{-1}
$N_c = 32$	5.47×10^{-2}	1.77×10^{-1}



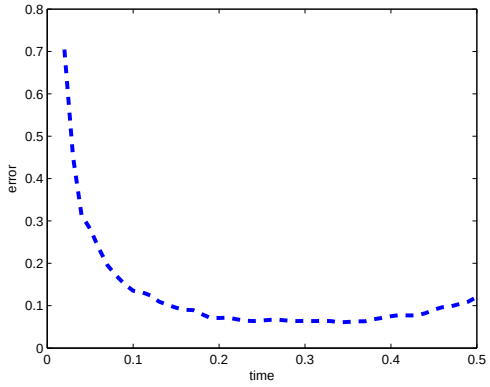
(a) L^2 norm relative errors of the solution at $N_c = 8$



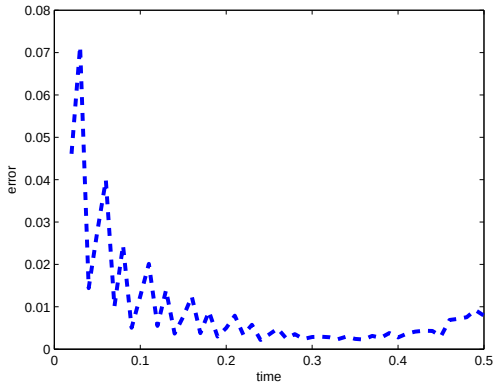
(b) L^2 norm relative errors of the solution at $N_c = 8$



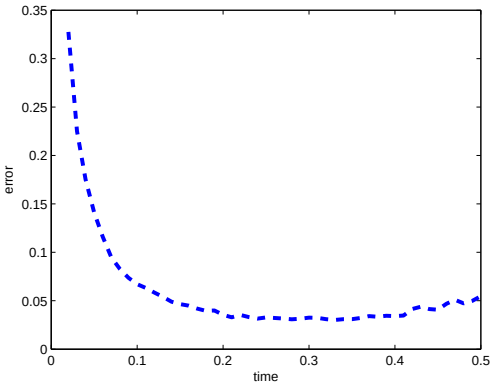
(c) L^2 norm relative errors of the solution at $N_c = 16$



(d) H^1 norm relative errors of the solution at $N_c = 16$



(e) L^2 norm relative errors of the solution at $N_c = 32$



(f) H^1 norm relative errors of the solution at $N_c = 32$

Figure 3.8: Example 3.3 - Relative errors of the solution

Chapter 4

Modified multiscale model reduction method for deterministic PDEs with locally degenerate coefficients

4.1 Difficulties with locally degenerate coefficients

The model reduction method does not apply very well to elliptic problems with degenerate coefficients. Denote $\gamma = \inf_{x\in\Omega}\{\lambda_{\min}(x)\} > 0$. We can show that the error produced by the model reduction method is proportional to $\frac{1}{\gamma}$. In the case $\gamma \ll 1$, the error will become very large and the model reduction method will not produce acceptable results. This is an essential difficulty for almost all other multiscale methods. Thus, upscaling multiscale elliptic problems with degenerate coefficients presents a considerable challenge.

Our error analysis shows that when γ is small, a major source of error is related to the harmonic coordinates. To reduce the upscaling error, we propose to use the harmonic coordinates as a basis to approximate the residual error and perform error correction. Specifically, we first implement the original model reduction method, and then use the harmonic coordinates to approximate the residual error to obtain the improved upscaled solution. We will show that by doing this, the accuracy of the modified model reduction method is significantly improved.

4.2 Analysis for one-dimensional elliptic PDEs

Recall that for the one-dimensional case, we have

$$\begin{cases} (a(x)u'(x))' = f(x), \quad x \in (0,1), \\ u(0) = u(1) = 0. \end{cases} \tag{4.1}$$

The harmonic coordinate F is

$$\begin{cases} (a(x)F'(x))' = 0, \qquad x \in (0,1), \\ F(0) = 0,\ F(1) = 1. \end{cases} \tag{4.2}$$

The effective equation is

$$\begin{cases} a(x)F'(x)(u_0'(x)/g'(x))' = f(x), \quad x \in (0,1), \\ u_0(0) = u_0(1) = 0. \end{cases} \tag{4.3}$$

And the solutions are

$$u(x) = C_0 \left(F(x) \int_0^x f(s)\mathrm{d}s - \int_0^x F(s)f(s)\mathrm{d}s + C_1 F(x) \right), \tag{4.4}$$

$$F(x) = \frac{1}{C_0} \int_0^x \frac{\mathrm{d}s}{a(s)}, \tag{4.5}$$

$$u_0(x) = C_0 \left(g(x) \int_0^x f(s)\mathrm{d}s - \int_0^x g(s)f(s)\mathrm{d}s + C_2 g(x) \right), \tag{4.6}$$

where

$$C_0 = \int_0^1 \frac{\mathrm{d}s}{a(s)}, \tag{4.7}$$

$$C_1 = -\int_0^1 f(s)\mathrm{d}s + \int_0^1 F(s)f(s)\mathrm{d}s, \tag{4.8}$$

$$C_2 = -\int_0^1 f(s)\mathrm{d}s + \int_0^1 g(s)f(s)\mathrm{d}s. \tag{4.9}$$

Denote $u_1 = \chi u_0'/g'$, we give

$$(u - u_0 - u_1)' = C_0 \left((C_1 - C_2)F' - \chi f \right). \tag{4.10}$$

First of all, since $f \in L^2$ and g is smooth, (4.6) implies that u_0 is smooth. Next, we study the error equation (4.10) term by term. Subtracting C_2 (4.9) from C_1 (4.8) we get $C_1 - C_2 = \int_0^1 \chi(s) f(s) \mathrm{d}s$. Since χ is small, we conclude that $(C_1 - C_2)$ and χf are small. We assume that $a(x)$ is small only in a localized region of D so that $\int_0^1 \frac{ds}{a(s)}$ is $O(1)$. This is a reasonable assumption in real applications since the degenerate region is often restricted to some very localized area. Under such an assumption, the term C_0 is bounded. Now we just need to deal with the term involving F'. Differentiating (4.5), we obtain

$$F' = (C_0 a(x))^{-1}. \tag{4.11}$$

By our assumption, F' could be large in some localized regions and thus the L^2 norm of $(u - u_0 - u_1)'$ would be large due to the effect of F'. This is the major source of error.

In order to control the H^1 norm of $u - u_0 - u_1$, we need to handle the F' term in (4.10). Note that

$$\begin{aligned} &(u - u_0 - u_1 - C_0(C_1 - C_2)F)' \\ =&C_0((C_1 - C_2)F' - \chi f) - C_0(C_1 - C_2)F' \\ =&- C_0 \chi f. \end{aligned} \tag{4.12}$$

This suggests that if we add a suitable multiple of F to our approximate solution $u_0 + u_1$, the error would drop significantly to the desired level. On the other hand, we require our solution to vanish on ∂D, but $F = x \neq 0$ on the boundary. To overcome this difficulty, we use $F - x$ to correct the residual error. This gives rise to

$$\begin{aligned} &(u - u_0 - u_1 - C_0(C_1 - C_2)(F - x))' \\ =&C_0((C_1 - C_2)F' - \chi f) - C_0(C_1 - C_2)(F' - 1) \\ =&C_0((C_1 - C_2) - \chi f). \end{aligned} \tag{4.13}$$

Such an approximation is more accurate according to our assumptions. Comparing (4.13) with (4.10), we can see that after adding a suitable multiple of $F - x$, the F' term appearing in (4.10) is now replaced by $x' = 1$ in (4.13). This suggests that we

use the harmonic coordinates to provide an error correction to the upscaled solution. In the two-dimensional space, the harmonic coordinates form a basis for the pseudo-null space. The original model reduction method cannot capture the effect of this null space since $\nabla \cdot (a\nabla F) = 0$. Since the major source of error is proportional to ∇F which is large in the degenerate regions, we need to correct this error. Luckily the harmonic coordinates on the boundary $(x|_{\partial D})$ have a natural smooth extension to the whole domain $(x|_D)$. We can use $F - x$ as a good candidate to improve the upscaled solution. By appropriately choosing a constant vector c, the effect of ∇F in the error term will be replaced by $\nabla x = I$, which is bounded. This leads to a significant drop in the error. To summarize, we have the following modified model reduction method:

1. Apply the original model reduction method to get an approximation u^*.

2. Choose a suitable constant vector $c \in \mathbb{R}^d$ and update the approximation by $u^* + c^T(F - x)$.

Next we will discuss some implementation issues of the modified model reduction method. An important question is how to choose c to reduce the residual error. We can consider $F - x$ as the extra basis functions and substitute $u^* + c^T(F - x)$ into the original PDE (2.1) to solve for c. We will work on a weak form of (2.1). Suppose ϕ is a test function. Multiplying ϕ to both sides of the equation and rearranging the equation, we have

$$c^T \int \nabla \phi^T a \nabla (F - x) \mathrm{d}x = \int f\phi \mathrm{d}x - \int \nabla \phi^T a \nabla u^* \mathrm{d}x. \tag{4.14}$$

Since c has two components, we also need two different test functions. A natural choice for the test functions would be $F - x$. Note that although the integration is performed on the whole domain D, the dimension of the linear system is very small, $d = 2$. Thus the additional cost of this error correction step is negligible. Substituting $\phi = F - x$ into (4.14) and using $\nabla \cdot (a\nabla F) = 0$, we can further simplify (4.14) as follows

$$\left(\int a(I - \nabla F) \mathrm{d}x \right) c = \int (F - x) f \mathrm{d}x + \int a \nabla u^* \mathrm{d}x. \tag{4.15}$$

In the next section we will show that such a numerical scheme works very well for some elliptic problems.

4.3 Generalization to parabolic equations

We can also apply the modified model reduction method to time-dependent problems such as parabolic equations. As discussed before, we could obtain the effective equations and solve for the effective solutions.

For locally degenerate problems, the approximation does not work very well and we need to fix it. Motivated by the elliptic equations, we introduce an error correction of the form $u^* + c(t)^T(F - x)$. By using a similar argument, we substitute this approximation into the PDE, multiply both sides by the test function $F - x$ and integrate it over the space domain. We would get an ODE for $c(t)$ as follows

$$\begin{aligned} &\left(\int (F-x)(F-x)^T \mathrm{d}x\right) c_t + \left(\int a(I - \nabla F)\mathrm{d}x\right) c \\ =&\int (F-x)(f - u_t^*)\mathrm{d}x + \int a\nabla u^* \mathrm{d}x. \end{aligned} \tag{4.16}$$

Multiplying both sides by $\left(\int (F-x)(F-x)^T \mathrm{d}x\right)^{-1}$, equation (4.16) could be written as

$$c_t + Bc = h(t), \tag{4.17}$$

where

$$B = \left(\int (F-x)(F-x)^T \mathrm{d}x\right)^{-1} \left(\int a(I - \nabla F)\mathrm{d}x\right) \tag{4.18}$$

is a constant matrix and

$$h(t) = \left(\int (F-x)(F-x)^T \mathrm{d}x\right)^{-1} \left(\int (F-x)(f - u_t^*)\mathrm{d}x + \int a\nabla u^* \mathrm{d}x\right) \tag{4.19}$$

is a vector function in time.

Note that we can solve the ODE (4.17) analytically, and the solution is

$$c(t) = e^{-Bt} \int_0^t e^{Bs} h(s) \mathrm{d}s, \tag{4.20}$$

where the exponential of a matrix M is defined by $e^M = \sum_{k=0}^{\infty} \frac{M^k}{k!}$. Thus in the numerical implementations, we can compute (4.20) numerically and the final approximation would be $u^* + c(t)^T(F - x)$.

4.4 Numerical results

In this section, we will present several numerical experiments for the two-dimensional elliptic equations and parabolic equations to demonstrate the effectiveness of our method. In all the examples, we choose the domain $D = [0,1] \times [0,1]$ and $f = 1$. Since it is difficult to construct a general enough test problem with an analytic solution, we use well-resolved numerical solutions to serve as the 'exact solutions'. In our computations, we use the standard linear finite element method, and choose a 256×256 mesh to obtain the well-resolved solution for the elliptic equations and a 128×128 mesh for the parabolic equations. To implement our method, the coarse meshes are chosen to be 16×16, and we will compare the results obtained by the original model reduction method with those obtained by the modified model reduction method.

Example 4.1. In this example, we consider the elliptic equation with a coefficient that is small only in a narrow channel. The coefficient is chosen to be

$$a(x,y) = \begin{cases} 1 \times 10^{-2} & \text{inside the channel,} \\ 3 + \cos(\frac{2\pi(x-0.8)}{\epsilon_1}) + \sin(\frac{2\pi(y-0.3)}{\epsilon_2}) & \text{otherwise,} \end{cases}$$

with $\epsilon_1 = \frac{1}{7}$ and $\epsilon_2 = \frac{1}{19}$. The channel is long and narrow, and lies in the region $\{0.2 \le x \le 0.9,\ 0.3 \le y \le 0.31\}$. The coefficient is very small ($a = 0.01$) inside the channel, but it is order $O(1)$ ($a = 1 \sim 5$) outside the channel, see Figure 4.1.

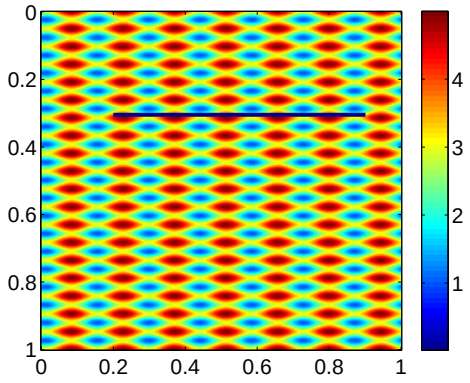


Figure 4.1: Example 4.1 - The coefficient a

We present the errors in the following table. Here MMsMRM stands for the modified model reduction method, MsMRM stands for the original model reduction method and FEM stands for the standard finite element method.

Table 4.1: Example 4.1 - Relative errors of the solution

	MMsMRM	MsMRM	FEM
L^2 norm	9.37×10^{-3}	2.28×10^{-2}	1.73×10^{-1}
H^1 norm	7.98×10^{-2}	1.45×10^{-1}	7.95×10^{-1}

From Table 4.1, we can see that the errors obtained by MsMRM and MMsMRM are smaller than those obtained by the finite element method both in L^2 and H^1 norms. MsMRM still has some reasonable accuracy, but it is not satisfactory. MMsMRM gives more desirable accuracy. We observe that the L^2 error of MMsMRM drops about one half of MsMRM, and the H^1 error of MMsMRM drops about one third of MsMRM.

Example 4.2. In this example, we would like to test how the two methods perform if we remove the degeneracy of the coefficient in the previous example. The purpose of this test is to show that when there is no degeneracy in the elliptic coefficient, the modification is not really needed for the original model reduction method. We choose the coefficient to be

$$a(x,y) = \begin{cases} 1 \times 10^2 & \text{inside the channel,} \\ 3 + \cos(\frac{2\pi(x-0.8)}{\epsilon_1}) + \sin(\frac{2\pi(y-0.3)}{\epsilon_2}) & \text{otherwise,} \end{cases}$$

with $\epsilon_1 = \frac{1}{7}$ and $\epsilon_2 = \frac{1}{19}$. The coefficient is uniformly of order $O(1)$ in the whole domain.

In Table 4.2, we present the relative errors for all the methods. Since the coefficient has multiscale information, both MMRM and MRM offer better accuracy than the finite element method. We can see that MRM already gives satisfactory accuracy. There is almost no improvement in MMRM. This example shows that the modified model reduction method is only needed when the coefficient is degenerate.

Example 4.3. In this example, we consider a slightly more complicated elliptic example in which case the coefficient is small inside a narrow channel and the three

Table 4.2: Example 4.2 - Relative errors of the solution

	MMsMRM	MsMRM	FEM
L^2 norm	9.43×10^{-3}	9.41×10^{-3}	7.15×10^{-2}
H^1 norm	7.46×10^{-2}	7.46×10^{-2}	2.59×10^{-1}

small inclusions. The coefficient is given below.

$$a(x,y) = \begin{cases} 0.005 & \text{inside the channel and some inclusions,} \\ 1 & \text{otherwise.} \end{cases}$$

The channel is the same as in Example 4.1, $\{0.2 \leq x \leq 0.9,\ 0.3 \leq y \leq 0.31\}$. There are three circular inclusions, $\{(x-0.5)^2+(y-0.7)^2 < 0.01^2\}$, $\{(x-0.2)^2+(y-0.8)^2 < 0.01^2\}$ and $\{(x-0.6)^2+(y-0.9)^2 < 0.01^2\}$. Inside these regions, the coefficient is equal to 0.005 while it is equal to 1 outside the regions. See Figure 4.2.

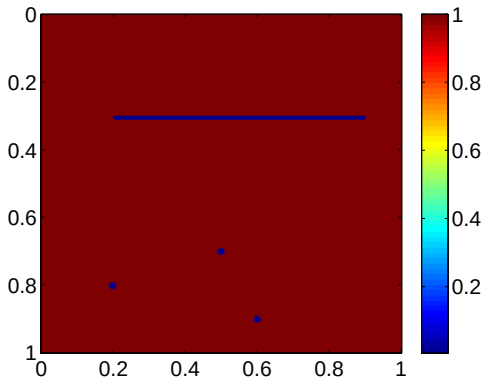


Figure 4.2: Example 4.3 - The coefficient a

Table 4.3: Example 4.3 - Relative errors of the solution

	MMsMRM	MsMRM	FEM
L^2 norm	6.84×10^{-3}	1.96×10^{-2}	1.42×10^{-1}
H^1 norm	7.09×10^{-2}	1.30×10^{-1}	7.77×10^{-1}

From Table 4.3, we observe the same qualitative improvement as in Example 4.1. Both MMRM and MRM improve the accuracy compared with the finite element method. The L^2 error of MMRM drops nearly one half of MRM, and the H^1 error of MMRM drops about one third of MRM, which shows that the modified model reduction method indeed gives more desirable accuracy when there is a degeneracy

in the elliptic coefficients.

Example 4.4. In the last example, we consider the parabolic equation when the coefficient is small in a narrow channel.

$$a(x,y) = \begin{cases} 0.01 & \text{inside the channel,} \\ 3 + \cos(\frac{2\pi(x-0.8)}{\epsilon_1}) + \sin(\frac{2\pi(y-0.3)}{\epsilon_2}) & \text{otherwise,} \end{cases}$$

with $\epsilon_1 = \frac{1}{5}$ and $\epsilon_2 = \frac{1}{13}$. The channel lies in the region $\{0.3 \leq x \leq 0.7,\ 0.3 \leq y \leq 0.32\}$, which is slightly wider than in Example 4.1. We compute the solution to time $T = 0.1$, and exhibit the error at T in the following table.

Table 4.4: Example 4.4 - Relative errors of the solution at $T = 0.1$

	MMsMRM	MsMRM	FEM
L^2 norm	4.92×10^{-3}	1.61×10^{-2}	1.04×10^{-1}
H^1 norm	5.74×10^{-2}	1.15×10^{-1}	5.74×10^{-1}

From Table 4.4, we can see that both MMRM and MRM offer improved accuracy compared with FEM. Moreover, the improvement of MMRM over MRM is even better than the elliptic case. The L^2 error of MMRM is less than one third of MRM, and the H^1 error of MMRM is about one half of MRM. Compared with the finite element method, the MMRM or MRM offers even more computational savings since we only need to compute the harmonic coordinates at $t = 0$. All subsequent computations can be performed on the coarse grid. This shows that our modified model reduction method indeed gives improved accuracy for parabolic equations with locally degenerate coefficients.

Chapter 5

Model reduction based multiscale data-driven stochastic method for elliptic PDEs with random coefficients

5.1 The Karhunen-Loève expansion

In the theory of stochastic processes, the Karhunen-Loève expansion [41, 43] is a representation of a stochastic process as an infinite linear combination of orthogonal functions. The importance of the Karhunen-Loève expansion is that it yields an optimal basis in the sense that it minimizes the total mean squared error.

Consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, whose event space is Ω and is equipped with σ-algebra $\mathcal{F}$ and probability measure $\mathbb{P}$. Suppose $u(x,\omega)$, defined on a compact spatial domain $D \subseteq R^d$, is a second-order stochastic process, i.e. $u(x,\omega) \in L^2(D\times\Omega)$. Its Karhunen-Loève expansion reads as

$$u(x,\omega) = \bar{u}(x) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\xi_i(\omega)\phi_i(x), \tag{5.1}$$

where $\bar{u}(x) = \mathbb{E}[u(x,\omega)]$, $\{\lambda_i, \phi_i(x)\}_{i=1}^{\infty}$ are the eigenpairs of the covariance kernel $C(x,y)$ and they satisfy

$$\int_D C(x,y)\phi(y)\mathrm{d}y = \lambda\phi(x). \tag{5.2}$$

The covariance kernel $C(x, y)$ is defined as

$$C(x, y) = \mathbb{E}[(u(x, \omega) - \bar{u}(x))(u(y, \omega) - \bar{u}(y))]. \tag{5.3}$$

The random variables $\{\xi_i(\omega)\}_{i=1}^{\infty}$ are defined as

$$\xi_i(\omega) = \frac{1}{\sqrt{\lambda_i}} \int_D (u(x, \omega) - \bar{u}(x))\phi_i(x)\mathrm{d}x \tag{5.4}$$

Moreover, these random variables $\{\xi_i(\omega)\}$ are of zero mean and are uncorrelated, i.e. $\mathbb{E}[\xi_i] = 0$, $\mathbb{E}[\xi_i\xi_j] = \delta_{ij}$. Generally, the eigenvalues λ_i's are sorted in descending order and cluster at zero, and their decay rate depends on the regularity of the covariance kernel $C(x, y)$. It has been proven that algebraic decay rate, i.e., $\lambda_k = O(k^{-\beta})$, is achieved asymptotically if the covariance kernel is of finite Sobolev regularity or exponential decay, i.e., $\lambda_k = O(e^{-\beta k})$ for some $\beta > 0$, if the covariance kernel is piecewise analytical [53]. In general, the decay rate depends on the correlation length of the stochastic solution. Small correlation length results in slow decay of the eigenvalues. In any case, an m-term truncated Karhunen-Loève expansion converges in $L^2(D \times \Omega)$ to the original stochastic process $u(x, \omega)$ as m tends to infinity. Let ϵ_m denote the truncation error. We have

$$||\epsilon_m||^2_{L^2(D\times\Omega)} = || \sum_{i=m+1}^{\infty} \sqrt{\lambda_i}\xi_i(\omega)\phi_i(x)||^2_{L^2(D\times\Omega)} = \sum_{i=m+1}^{\infty} \lambda_i \to 0, \quad m \to \infty, \tag{5.5}$$

where we have used the bi-orthogonality of the Karhunen-Loève expansion.

In practical computations, we truncate the Karhunen-Loève expansion into its first m terms, and obtain the following truncated Karhunen-Loève expansion

$$u(x, \omega) \approx \bar{u}(x) + \sum_{i=1}^{m} \sqrt{\lambda_i}\xi_i(\omega)\phi_i(x). \tag{5.6}$$

The truncation error analysis in (5.6) reveals the most important property of the expansion. More specifically, given any integer m and orthonormal basis $\{\psi_i(x)\}_{i=1}^{m}$,

we may approximate the stochastic process $u(x,\omega)$ by

$$u_{m,\psi}(x,\omega) = \bar{u}(x) + \sum_{i=1}^{m} \zeta_i(\omega)\psi_i(x), \tag{5.7}$$

where $\zeta_i(\omega)$, $i = 1, ..., m$, are the expansion coefficients. Among all m-term approximations using an orthonormal basis, the Karhunen-Loève expansion given by (5.6)) is the one that minimizes the total mean square error. In this sense, we say that the Karhunen-Loève expansion gives the optimal (or the most compact) basis to represent the stochastic solution in the energy norm. Due to the bi-orthogonality of the Karhunen-Loève expansion, we will call the stochastic part of the Karhunen-Loève expansion the data-driven basis in the rest of the thesis.

5.2 Derivation of model reduction based multiscale data-driven stochastic method

We consider the stochastic elliptic equation

$$\begin{cases} -\nabla \cdot (a(x,\omega)\nabla u(x,\omega)) = f(x), & x \in D, \omega \in \Omega, \\ u(x,\omega) = 0, & x \in \partial D, \omega \in \Omega. \end{cases} \tag{5.8}$$

We will propose a multiscale data-driven stochastic method to reduce the computational complexity of solving such problems. Our MsDSM consists of offline and online stages. In the offline stage, we derive an effective stochastic equation that can be resolved on a coarse grid. We then construct a data-driven stochastic basis that gives a compact representation for the solutions of the effective stochastic equation for a broad range of forcing functions. In the online stage, we represent the multiscale stochastic solution in terms of this data-driven stochastic basis and we just need to solve a small number of coupled deterministic PDEs. This leads to considerable computational savings when we need to solve the multiscale stochastic PDE under the multiquery settings.

5.2.1 Effective stochastic equations

It is important to note that the effective coefficient depends only on the multiscale coefficient and decompositions of the harmonic coordinates, and does not depend on the forcing term. Therefore, we can apply this idea to upscale the multiscale stochastic coefficient in equation (5.8). For each fixed sample $\omega \in \Omega$ (ω can be chosen by Monte Carlo method or stochastic collocation method), the multiscale problem (5.8) becomes a deterministic PDE. We first solve the corresponding homogeneous problem with specified boundary conditions to obtain the harmonic coordinates F. Then, we decompose the harmonic coordinates F into a smooth part g plus a small part χ, $F = g + \chi$. The effective coefficient can be given in terms of $a(x, \omega)$, g, and χ, i.e.

$$a^*(x, \omega) = a(x, \omega)(I + \frac{\partial \chi}{\partial x}(x, \omega)\frac{\partial x}{\partial g}(x, \omega)), \tag{5.9}$$

where I is the identity matrix. The effective coefficient in equation (5.9) is valid for each sample ω in the sample space Ω. Looping over the samples, we can obtain an effective stochastic equation of the following form

$$\begin{cases} -\nabla \cdot (a^*(x, \omega)\nabla u^*(x, \omega)) = f(x), & x \in D, \omega \in \Omega, \\ u^*(x, \omega) = 0, & x \in \partial D, \omega \in \Omega. \end{cases} \tag{5.10}$$

According to the previous analysis, the solution to the effective equation (5.10) is one order smoother than the original solution. Thus, we can solve the effective equation on a coarse mesh.

To save memory, we compute the truncated Karhunen-Loève expansion of each entry of the $a^*(x, \omega)$ in equation (5.10),

$$a^*(x, \omega) \approx \bar{a}(x) + \sum_{m=1}^{M} \sqrt{\lambda_m}\xi_m(\omega)\phi_m(x). \tag{5.11}$$

Also, we use the sparse grid based stochastic collocation method in paper [11]. We need only to save these Karhunen-Loève expansion results instead of a large amount of samples, which significantly reduces the memory cost.

In addition, we need to compute the correction term $(\chi^T \frac{\partial x}{\partial g})(x, \omega)$ in the offline

stage. Actually, this can be done simultaneously when we derive the effective stochastic equation. For each fixed sample $\omega \in \Omega$, after we decompose the harmonic coordinates F into a smooth part g and a highly oscillatory part χ, we can compute the correction vector term $(\chi^T \frac{\partial x}{\partial g})(x, \omega)$. Based on these samples, we can calculate the truncated Karhunen-Loève expansion of the correction term $\chi^T \frac{\partial x}{\partial g}(x, \omega)$, i.e.

$$\chi^T \frac{\partial x}{\partial g}(x, \omega) = \bar{c}(x) + \sum_{n=1}^{N} \sqrt{\lambda_n} \vartheta_n(\omega) \psi_n(x). \tag{5.12}$$

5.2.2 Data-driven stochastic basis for the effective stochastic equations

Now we consider the effective equation (5.10). We first note that the coefficient $a(x, \omega)$ in equation (5.8) is given in terms of r independent random variables, $a(x, \omega) = a(x, \xi(\omega)) = a(x, \xi_1(\omega), ..., \xi_r(\omega))$. Therefore, by the Doob Dynkin lemma, the harmonic coordinates as well as the effective coefficient $a^*(x, \omega)$ can still be represented by these random variables, i.e. $a^*(x, \omega) = a^*(x, \xi(\omega)) = a^*(x, \xi_1(\omega), ..., \xi_r(\omega))$.

We now begin our construction of the data-driven stochastic basis for the effective stochastic equation (5.10), and we discuss the data-driven stochastic basis in stochastic collocation representation. It consists of two steps, initial leaning and updating steps. See Figure 5.1 for the general framework. We refer to [15] for more details.

In the initial learning step, we first use the stochastic collocation method to generate J collocation points $z_j \in R^r$ according to the distribution of the coefficient $a(x, \omega)$ in equation (5.8) as well as the associated weights $w_j \in R$. Then, we solve (5.10) with the random variable evaluated at the collocation grid points and $f_0(x) = 1$ as the right-hand side

$$\begin{cases} -\nabla \cdot (a^*(x, z_j) \nabla u^*(x, z_j)) = f_0(x), & x \in D, j = 1, ..., J, \\ u(x, z_j) = 0, & x \in \partial D, j = 1, ..., J. \end{cases} \tag{5.13}$$

By solving the above equation (5.13), we can obtain the values of the stochastic solution $u^*(x, \omega; f_0)$ on the collocation points, i.e., $\{u^*(x, z_j; f_0)\}_{j=1}^J$. The m_1-term Karhunen-Loève expansion of the solution $u^*(x, \omega; f_0)$ gives the dominant components in the random space. We use the decaying property of eigenvalues to select parameter

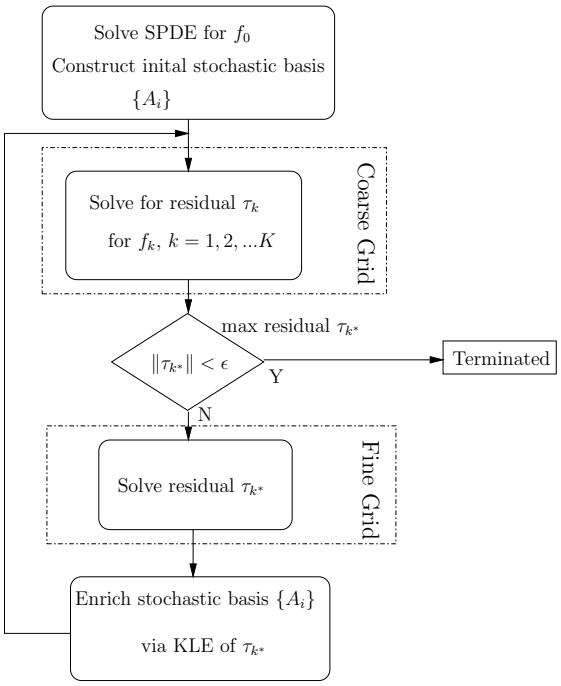


Figure 5.1: Greedy stochastic basis enriching algorithm on a coarse-fine grid hierarchy.

m_1, i.e. the number of stochastic basis m_1 can be chosen such that $\lambda_{m_1+1}/\lambda_1$ is smaller than some pre-defined threshold, say, 10^{-4}. We denote the truncated Karhunen-Loève expansion as

$$u^*(x, \omega; f_0) \approx \bar{u}(x; f_0) + \sum_{i=1}^{m_1} \sqrt{\lambda_i} B_i(\omega) \phi_i(x; f_0). \tag{5.14}$$

We call the stochastic basis $\{B_i(\omega)\}_{i=0}^{m_1}$ in equation (5.14) the data-driven stochastic basis, where $B_0(\omega) = 1$. In general, the stochastic basis constructed by using f_0 may not be adequate to give an accurate approximation of the SPDE for another right hand side. We need to supplement the stochastic basis by using multiple trial functions involving other f_k.

In the preconditioning and update step, we propose a greedy-type algorithm, and adopt a two-level preconditioning strategy [26] to enrich the stochastic basis. First, we perform an error analysis. Given a new right-hand side $f_1(x)$ for some choice of f, we expand the solution in terms of the stochastic basis, $\{B_i(\omega)\}_{i=0}^{m_1}$,

$$u^*(x, \omega; f_1) \approx \bar{u}(x; f_1) + \sum_{i=1}^{m_1} B_i(\omega) u_i(x; f_1) \equiv \sum_{i=0}^{m_1} B_i(\omega) u_i(x; f_1). \tag{5.15}$$

In the rest of this subsection, we also use $u_i(x) \equiv u_i(x; f_1)$ for simplification. We use the standard stochastic Galerkin method to obtain the coefficient $u_i(x)$. Specifically, we substitute the expansion (5.15) into the effective equation (5.10), multiply both sides by $B_j(\omega)$, and take expectations. This gives rise to a coupled PDE system for the expansion coefficient $u_i(x)$,

$$\begin{cases} -\nabla \cdot (\mathbb{E}[a^* B_i B_j] \nabla u_i) = f_1(x)\mathbb{E}[B_j], & x \in D, j = 0, 1, \ldots, m_1, \\ u_i(x) = 0, & x \in \partial D, \end{cases} \tag{5.16}$$

where Einstein summation is assumed. The term $\mathbb{E}[a^* B_i B_j]$ can be calculated by the stochastic collocation method. Solving the coupled deterministic PDE system (5.16) by the standard finite element method, we obtain the expansion coefficient $\{u_i(x)\}_{i=0}^{m_1}$ and an approximate solution for $u^*(x, \omega; f_1)$ given by (5.15). We know that the exact solution can be written as

$$u^*(x, \omega; f_1) = \sum_{i=0}^{m_1} B_i(\omega) u_i(x; f_1) + \tau(x, \omega; f_1), \tag{5.17}$$

where $\tau(x, \omega; f_1)$ is the error. Simple calculations show that the error satisfies the following equation

$$-\nabla \cdot (a^*(x, \omega) \nabla \tau(x, \omega; f_1)) = f_1(x) + \sum_{i=0}^{m_1} \nabla \cdot (a(x, \omega) B_i(\omega) \nabla u_i(x)). \tag{5.18}$$

For a different f_k, we can obtain a similar error equation for the error $\tau(x, \omega; f_k)$ by replacing f_1 by f_k in the above error equation. To verify the effectiveness of the stochastic basis, we solve the residual equation (5.18) on a coarse grid for each $f_k(x)$, $k = 1, ..., K$, and obtain the error $\{\tau(x, \omega; f_k)\}_{k=1}^{K}$. If $\max_{1 \le k \le K} ||\tau(x, \omega; f_k)|| < \epsilon_0$, then we consider this stochastic basis complete. Here, we choose $|| \cdot ||$ as the L^2 norm of the variance of the stochastic solution. Otherwise, we identify the maximum error $\tau_{k^*} = \max_{1 \le k \le K} ||\tau(x, \omega; f_k)||$ and the corresponding trial function $f_{k^*}(x)$. Subsequently, we solve the residual equation (5.18) for this trial function $f_{k^*}(x)$ one more time on a fine grid. Again, we perform the Karhunen-Loève expansion for the residual solution $\tau(x, \omega; f_{k^*})$, and extract several dominant components in the random space,

and supplement them to the current stochastic basis. We use $\{B_i(\omega)\}_{i=0}^{m_2}$ to denote the updated stochastic basis. This process is repeated until the maximum residual is below the prescribed threshold ϵ_0.

In the online stage, for each query $f(x)$ in the original equation (5.8), the corresponding stochastic solution $u(x,\omega)$ can be approximated by the MsDSM solution in two steps. First, with our data-driven stochastic basis $\{B_i(\omega)\}_{i=0}^{m}$, we use the standard stochastic Galerkin method to solve the effective stochastic equation (5.10) to obtain $u^*(x,\omega)$. Then, we obtain the approximate solution by adding correction terms into $u^*(x,\omega)$, i.e.

$$u(x,\omega) \approx u_{MsDSM}(x,\omega) \equiv u^*(x,\omega) + \chi^T \frac{\partial x}{\partial g}(x,\omega)\nabla u^*(x,\omega). \tag{5.19}$$

The construction of the effective stochastic equation (5.10) and the data-driven stochastic basis $\{B_i(\omega)\}_{i=0}^{m}$ could be expensive. However, in a multiple query problem, the MsDSM offers considerable computational savings over traditional methods because of the model reduction in both the physical and stochastic spaces. We will demonstrate this through several numerical examples.

Remark 5.1. It is important to point out that since our methods involve the computation of global harmonic coordinates, the memory consumption becomes a serious issue when the number of random variables become very large. We are currently adopting the multilevel Monte Carlo method to tackle this problem.

5.2.3 Complete algorithm

In this section, we give the complete algorithm of the MsDSM to solve the multiscale stochastic equation under the multiquery setting. Our method consists of offline and online stages. Since the online stage is pretty straightforward and has been presented in a previous section, we state only the offline computation algorithm as follows.

MsDSM offline computation.

- (I) (Derive the effective stochastic equations and calculate the correction terms)
  - Loop over all sparse grids, compute harmonic coordinates, and obtain the effective stochastic equation (5.10).

- Compute the KL expansion of the effective coefficient (5.11) to obtain a compact representation.
- Compute the correction term $(\chi^T \frac{\partial x}{\partial g})$ as well as its Karhunen-Loève expansion (5.12).

• (II) (Construct the MsDSM basis for effective stochastic equation):

- Step II.1 (*Initial learning on the fine grid*):
  * Solve the effective equation(5.10) with $f_0(x)$ as a forcing function to obtain $u^*(x, \omega; f_0)$.
  * Calculate the truncated Karhunen-Loève expansion of $u^*(x, \omega; f_0)$, and use the first m_1 terms of the stochastic modes to obtain the current data-driven basis $\{B_i(\omega)\}_{i=0}^{m_1}$, where $B_0(\omega) = 1$.
- Step II.2 (*Preconditioning on the coarse grid*):
  * For each $f_k(x)$, solve effective equation (5.10) using the current stochastic basis $\{B_i(\omega)\}_{i=0}^{m_1}$ and the stochastic Galerkin method to obtain the solution $u_c^*(x, \omega; f_k)$.
  * For each $f_k(x)$, solve an residual equation (5.18) to obtain the approximate residual error $\tau_k = \tau(x, \omega; f_k)$.
  * Let $k^* = arg\max_{0 \le k \le K} ||\tau_k||$. If $||\tau_{k^*}|| < \epsilon_0$, goto Step II.4; otherwise , and goto Step II.3.
- Step II.3 (*Update on fine grid*):
  * Solve the error equation associated with $f_{k^*}(x)$ to obtain the residual error $\tau_{k^*} = \tau(x, \omega; f_{k^*})$.
  * Enrich the current stochastic basis $\{B_i(\omega)\}_{i=0}^{m_1}$ by the Karhunen-Loève expansion of τ_{k^*}, and use $\{B_i(\omega)\}_{i=0}^{m_2}$ to denote the updated stochastic basis. Goto Step II.2.
- Step II.4 (*Termination*):
  * Save the data-driven stochastic basis $\{B_i(\omega)\}_{i=0}^{m}$ and relevant statistical quantities.

- (III) (Save the relevant data):
  - Save the data-driven stochastic basis $\{B_i(\omega)\}_{i=0}^m$ and the Karhunen-Loève expansion of correction term $(\chi^T \frac{\partial x}{\partial g})$.

5.3 Computational complexity analysis

The computational time of the MsDSM consists of both the offline and online part. The offline computation can be very expensive if we use a brute force way to construct the data-driven basis. In this section, we will demonstrate through computational complexity analysis that the overhead time of offline computation is acceptable, and the online computation is super fast. It is well-known that the stochastic collocation method is very effective in solving SPDE when the stochastic solution is smooth in the stochastic space. Therefore, we choose the stochastic collocation finite element method (SCFEM) as a benchmark, and compare the computational cost of the MsDSM and the SCFEM. We will compare the performance of the MsDSM and the DSM as well.

In [15], the authors have already done a thorough study, and have explained why DSM is superior to the traditional methods, such as gPC, SC, and Monte Carlo method under the multiquery setting. The same property still holds for the MsDSM, since it is designed with the same technique. In our numerical experiments, we find that the offline computational costs of the MsDSM and DSM have the same order of magnitude. However, due to the model reduction in the physical domain, the MsDSM offers more computational savings in the online stage than the DSM.

We will demonstrate this by solving a model problem (5.8) on $D = [0,1] \times [0,1]$ with the coefficient given by

$$\begin{aligned} a =& 0.1 + \xi_1(\omega)\frac{2 + 1.8\sin(2\pi x_1/\epsilon_1)}{2 + 1.8\sin(2\pi x_2/\epsilon_1)} \\ &+ \xi_2(\omega)\frac{2 + 1.8\sin(2\pi x_2/\epsilon_2)}{2 + 1.8\cos(2\pi x_1/\epsilon_2)} + \xi_3(\omega)\frac{2 + 1.8\cos(2\pi x_1/\epsilon_3)}{2 + 1.8\sin(2\pi x_2/\epsilon_3)}, \end{aligned}$$

where $\{\epsilon_i\}_{i=1}^3$ are multiscale parameters, and $\{\xi_i\}_{i=1}^3$ are independent uniform random variables in $[0,1]$.

Table 5.1: Computational time of the linear equation solver for one collocation point. (Time: Sec.)

$N_h = 32^2$	$N_h = 64^2$	$N_h = 128^2$	$N_h = 256^2$	$N_h = 512^2$	$N_h = 1024^2$
0.0065	0.0359	0.2577	1.6490	18.6875	140.0816

Let N_h and J denote the number of the physical grid points and sparse grid points, respectively. We assume that in all tests level six sparse grids in the SCFEM will give an accurate result. Therefore, we choose $J = 135$. All the simulations and comparisons are conducted on a single computing node with 16 GB memory at the Caltech Center for Advanced Computing Research (CACR).

5.3.1 Computational cost of the SCFEM solver

We first show the computational cost of solving the Eq.(5.8) once using the stochastic collocation method in Table 5.1 (the same result can be applied to a Monte Carlo solver). The SCFEM is very effective if the SPDE solution is smooth in the stochastic dimension; however, when the SPDE solution has multiscale features in the physical dimension, the SCFEM becomes very expensive as demonstrated in Table 5.1. For instance, it takes about 1.89×10^4 (135×140.0816) seconds to obtain a single query result on a 1024^2 mesh grid. Let t_{SCFEM} denote the computational time of the SCFEM solver for one forcing function, then t_{SCFEM} is approximately given by

$$t_{SCFEM} \approx 2.45 \times 10^{-7} J N_h^{1.5}. \tag{5.20}$$

5.3.2 Computational cost of the MsDSM and the DSM solvers

Both the MsDSM and the DSM consist of offline and online computational cost. In [15], the authors have performed a complexity analysis for the DSM, and have compared it with other commonly used methods in the multiquery setting, such as the gPC, the gSC, and the MC methods. They have adopted the randomized SVD algorithm and a two-level preconditioning method to reduce the overhead time in the offline computing. Let n_c denote the query number, i.e., the number of forcing functions. They demonstrated through computational complexity analysis and nu-

merical examples that with all the cost-saving measures, the DSM is superior to the traditional method if one needs to solve the elliptic problem (5.8) with a relatively small number of queries. The MsDSM inherits all these cost-saving measures when we construct the DSM basis for the effective SPDE. The only extra computational cost stems from the derivation of the effective SPDE and calculating its correction term. Roughly speaking, this part of computational cost is equivalent to solving equation (5.8) with 2 forcing functions. In Table 5.2, we list the offline computational cost of the MsDSM and the DSM on a different mesh, where we fix the basis number $m = 7$. We also list the cost of SCFEM for one forcing function, where the CPU time on one collocation point is obtained by the time model (5.20) and $J = 135$. One can see that the offline computational costs of the MsDSM and the DSM have the same order of magnitude. In addition, the offline computational cost of the MsDSM or the DSM is approximately equal to the cost of performing SCFEM for several different forcing functions.

We assume that the data-driven basis with 7 modes gives sufficient approximation to the solution space. Let t_{DSMoff} and $t_{MsDSMoff}$ denote the computational time of DSM and MsDSM in the offline stage, respectively. Then, they are approximately given by

$$\begin{aligned} t_{DSMoff} &\approx 7.65 \times 10^{-5} N_h^{1.5}, \\ t_{MsDSMoff} &\approx 1.52 \times 10^{-4} N_h^{1.5}. \end{aligned} \tag{5.21}$$

In online stage of the MsDSM or the DSM, we use the standard Galerkin method to solve equation (5.8). In the multiple query setting, the stiffness matrix S for the DSM or the MsDSM solver is fixed and the load vector b is different for each query. We can compute the Cholesky decomposition of S in advance, and the computational time is decided only by the forward and backward substitutions in solving the linear equation system. Actually, we can do the Cholesky decomposition of the stiffness matrix $S = LL^T$ in the offline stage, and save only the decomposition result L. The computational time of Cholesky decomposition is negligible compared with the training data-driven basis. Thus, we do not consider this part of the cost.

Table 5.2: Computational time of the offline computation. (Time: Sec.). m=7.

Grid Number	$N_h = 336^2$	$N_h = 360^2$	$N_h = 384^2$	$N_h = 408^2$	$N_h = 432^2$	$N_h = 456^2$
DSM	1790.1	2341.8	2640.1	3167.4	3870.5	4522.3
MsDSM	2092.3	2466.6	3188.3	3662.5	4305.8	4960.2
SCFEM	664.3	811.5	978.5	1166.5	1376.7	1610.3

Table 5.3: Computational time of forward/back substitution. (Time: Sec.) m is the basis number. The data marked with an asterisk is obtained by extrapolation.

	$N_h = 64^2$	$N_h = 128^2$	$N_h = 256^2$	$N_h = 512^2$	$N_h = 1024^2$
m=5	0.0626	0.4102	2.4672	17.0917	(*)114.5388
m=7	0.1281	0.8383	5.4933	(*)37.5849	(*)255.0034
m=9	0.2347	1.5620	10.3214	(*)66.6531	(*) 438.6207

Let t_{fb} denote the time of forward and backward substitutions. In Table 5.3, we list the computation time of t_{fb} for different mesh grids and basis numbers. If we choose $m = 7$, then t_{fb} is approximately by

$$t_{fb} \approx 1.27 \times 10^{-6} N_h^{1.4}. \tag{5.22}$$

Roughly speaking, if the MsDSM is applied on a coarse grid with a coarsening factor C in each direction, the speedup would be $\sim (C^2)^{1.4}$ in the online stage for each query. For example, if $C = 16$, the speedup is ~ 2352 $(256^{1.4})$. This essentially reveals the power of the upscaling method.

Remark 5.2. We do not consider the computational time of adding correction terms into the MsDSM solution here. From numerical results in the next section, we can find that this part of cost is also very small compared to the SCFEM solver.

Remark 5.3. The stiffness matrix S is a sparse positive definite matrix; however, the Cholesky decomposition matrix L is not sparse anymore. Before we perform the Cholesky decomposition, we reorder the matrix S using the approximate minimum degree (AMD) algorithm to ensure the least fill-in.

5.4 Numerical examples

In this section, we perform numerical experiments to test the performance and accuracy of the proposed MsDSM. We also demonstrate the computational efficiency of

MsDSM over the traditional method, such as the stochastic collocation finite element method (SCFEM) in solving the multiquery problems with multiscale features. Finally, we compare the computational cost and accuracy of the MsDSM and the DSM in solving multiscale problems.

5.4.1 Comparison of the MsDSM and the SCFEM

Example 5.1. We consider the following stochastic elliptic equation with multiple scales on $D = [0,1] \times [0,1]$ with the multiscale information described by

$$a(x,y,\omega) = 0.1 + \frac{\xi_1(\omega)}{2 + 1.6\sin(2\pi(x-y)/\epsilon_1)} + \frac{\xi_2(\omega)}{4 + 1.8(\sin(2\pi x/\epsilon_2) + \sin(2\pi y/\epsilon_2))} + \frac{\xi_3(\omega)}{10(2 + 1.8\sin(2\pi(x-0.5)/\epsilon_3))(2 + 1.8\sin(2\pi(y-0.5)/\epsilon_3))},$$

where $\epsilon_1 = 1/3$, $\epsilon_2 = 1/11$ and $\epsilon_3 = 1/19$, and $\{\xi_i\}_{i=1}^3$ are independent uniform random variables in $[0,1]$. In Fig. 5.2, we plot four samples of the coefficient $a(x,y,\omega)$. One can see that the coefficient oscillates very rapidly, which will generate small scale features in the stochastic solution. The forcing terms are $f(x,y) \in \{\sin(k_i\pi x + l_i)\cos(m_i\pi x + n_i)\}$, where k_i, l_i, m_i, and n_i are random numbers, where k_i, l_i, m_i and n_i are uniformly distributed over the interval $[0, 0.5]$.

In our computations, we choose a 384×384 fine mesh to well resolve the spatial dimension of the stochastic solution $u(x,y,\omega)$. Since the stochastic solution $u(x,y,\omega)$ is smooth in stochastic space, we use the sparse-grid based stochastic collocation method to discretize the stochastic dimension. First, we conduct a convergence study, and find that the relative errors of mean and STD between the solutions obtained by level seven sparse grids in the SCFEM and higher level sparse grids are smaller than 0.1% both in L^2 and H^1 norm. Therefore, we choose level seven sparse grids with 207 points in the SCFEM and the MsDSM when we compare the computational cost of these two different methods. The reference solution is obtained by using higher-level sparse grids.

To implement the MsDSM, the coarse meshes are chosen to be 8×8, 16×16, 32×32 and 64×64 respectively, and we compare the results on different meshes, and calculate the convergence rate. We remark that in the MsDSM, the forcing function

$f(x, y)$ should be well-resolved by the coarse mesh, otherwise the numerical error will be large. We use this random training strategy to reduce the computational cost.

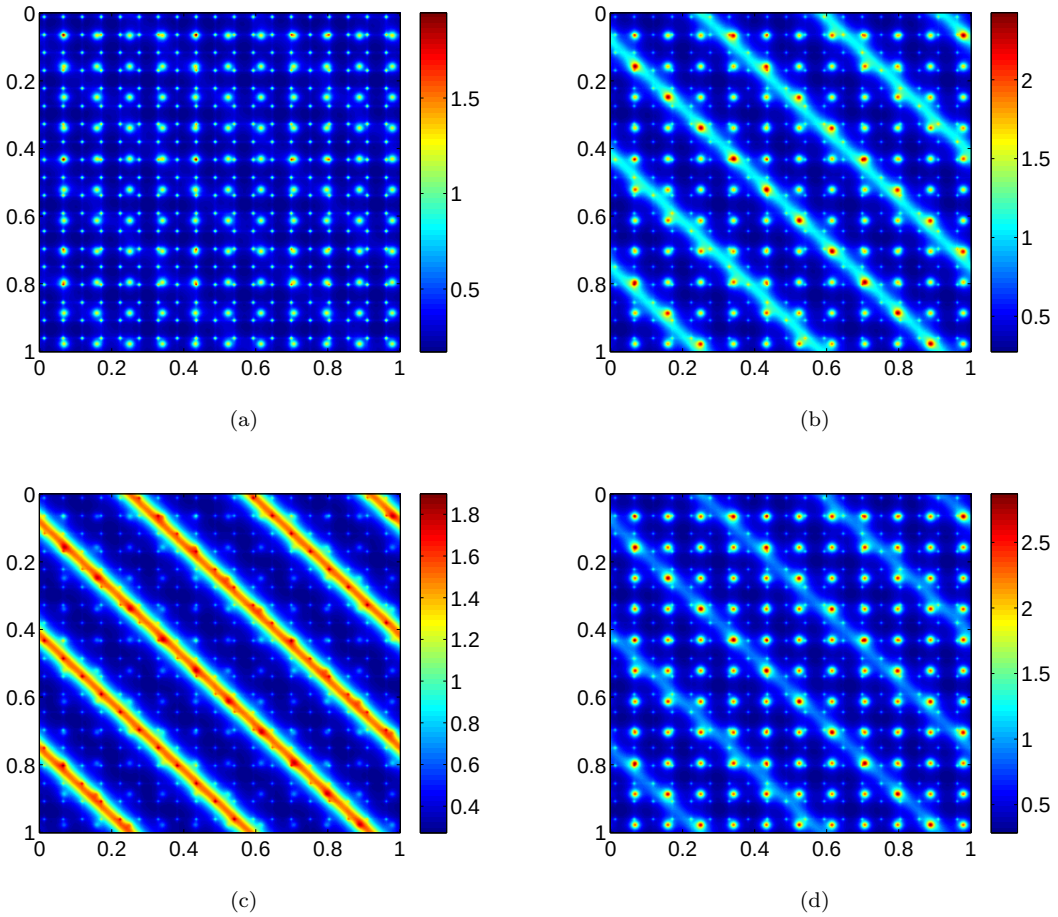


Figure 5.2: Example 5.1 - Some samples of the coefficient a

Multiquery results in the online stage. The MsDSM solver using 207 sparse grids in the computation produces $m = 7$ modes in the data-driven stochastic basis. In the online stage, we use them to solve the effective equation of the multiscale SPDE (5.8). In Table 5.4-5.5, we choose $f(x, y) = \sin(.47\pi x + 0.07)\cos(0.03\pi y + 0.25)$ and list the mean of the relative errors on different coarse mesh grids. It can be seen that the mean and STD of the MsDSM solution match the exact solution very well.

Compare the MsDSM solver with the exact SCFEM solver. For the SCFEM solver on a 384×384 mesh, it will take 1648.38 seconds to solve equation (5.8) with one specific forcing term $f(x, y)$. Thus in a multiquery problem, if we need to solve equation (5.8) with n different forcing term $f(x, y)$, the total computational cost will

Table 5.4: Example 5.1 - L^2 and H^1 norm relative errors of the mean

	L^2 norm	H^1 norm
$N_c = 8$	5.48×10^{-2}	1.58×10^{-1}
$N_c = 16$	2.20×10^{-2}	8.79×10^{-2}
$N_c = 32$	7.65×10^{-3}	5.02×10^{-2}
$N_c = 64$	3.03×10^{-3}	2.25×10^{-2}

Table 5.5: Example 5.1 - L^2 and H^1 norm relative errors of the STD

	L^2 norm	H^1 norm
$N_c = 8$	7.63×10^{-2}	6.28×10^{-2}
$N_c = 16$	2.30×10^{-2}	3.18×10^{-2}
$N_c = 32$	7.60×10^{-3}	9.25×10^{-3}
$N_c = 64$	3.60×10^{-3}	4.42×10^{-3}

be $t_{SCFEM} = 1648.38n$. If we choose $N_c = 64$ in the MsDSM solver, the offline computation will cost 4732.66 seconds, which includes the computational time for deriving effective SPDE, calculating correction term, and constructing DSM basis for the effective SPDE. In the online stage of the MsDSM, it takes 1.27 seconds to compute each query, thus, the total computational cost will be $t_{MsDSM} = 4732.66 + 1.27n$. We plot the total computational time in Figure 5.3. One can see that the MsDSM offers considerable computational savings over the SCFEM, if we need to solve the same stochastic PDE many times with multiple forcing functions. Simple calculation shows that if we need to solve the original SPDE with more than three different forcing functions, the MsDSM will be superior to the SCFEM.

Example 5.2. In this example, we consider the SPDE (5.8) on $D = [0, 1] \times [0, 1]$ with the coefficient given by a random linear combination of five fixed coefficient fields plus

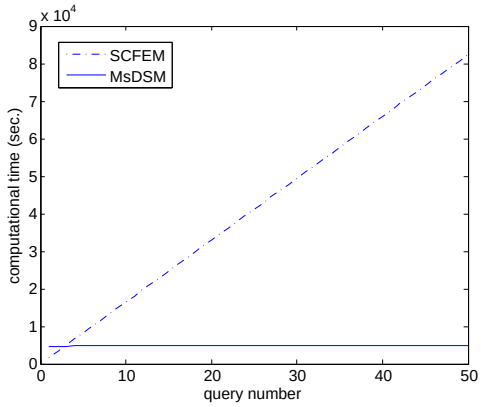


Figure 5.3: Example 5.1 - The computation time comparison

a constant, i.e.,

$$a(x, y, \omega) = \sum_{i=1}^{5} \xi_i(\omega)|\theta_i(x, y)| + 0.5,$$

where $\{\xi_i\}_{i=1}^{5}$ are independent uniform random variables in $[0, 1]$. $\theta_i(x, y)$, $i = 1, ..., 5$ are defined on a 3×3, 5×5, 9×9, 17×17, and 31×31 grids over the domain D. For each grid cell, the value of $\theta_i(x, y)$ is normally distributed. In Figure 5.4, we show four samples of the coefficient $a(x, y, \omega)$. One can see that the coefficients are very rough, and do not satisfy scale separation or have any periodic structure. The implementation of the SCFEM and the MsDSM are exactly the same as in the previous example.

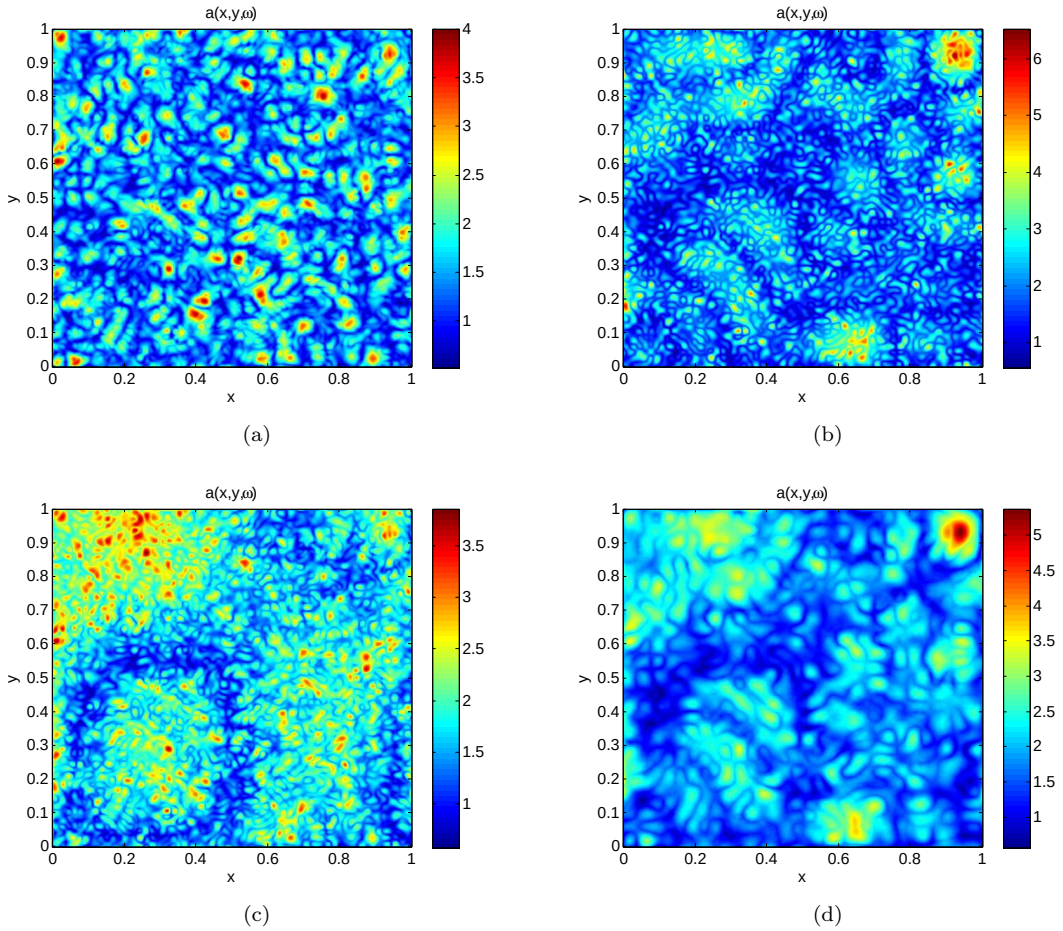


Figure 5.4: Example 5.2 - Some samples of the coefficient a

Multiquery results in the online stage. The MsDSM solver using 903 sparse grids

in the computation produces $m = 8$ modes in the data-driven stochastic basis. In the online stage we use them to solve the effective equation of the multiscale SPDE (5.8). In Table 5.6-5.7, we choose $f(x, y) = \sin(0.33\pi x + 0.15)\cos(0.43\pi y + 0.22)$ and show the relative errors of mean and STD of the MsDSM solution in L^2 norm and H^1 norm, respectively.

Compare the MsDSM solver with the SCFEM solver. For the SCFEM solver, it will take 7626.34 seconds to solve equation (5.8) with one specific forcing term $f(x, y)$. Thus in a multiquery problem, if we need to solve equation (5.8) with n different forcing terms, $f(x, y)$, the total computational cost will be $t_{SCFEM} = 7626.34n$. If we choose $N_c = 64$ in the MsDSM solver, the offline computation will cost 21231.56 seconds. In the online stage of the MsDSM, it takes 1.82 seconds to compute one query, thus the total computational cost will be $t_{MsDSM} = 21231.56 + 1.82n$. The MsDSM offers considerable computational savings over the SCFEM if we need to solve the same SPDE with more than three different forcing functions.

Table 5.6: Example 5.2 - L^2 and H^1 norm relative errors of the mean

	L^2 norm	H^1 norm
$N_c = 8$	1.09×10^{-1}	2.53×10^{-1}
$N_c = 16$	4.83×10^{-2}	1.39×10^{-1}
$N_c = 32$	1.15×10^{-2}	8.76×10^{-2}
$N_c = 64$	4.22×10^{-3}	4.15×10^{-2}

Table 5.7: Example 5.2 - L^2 and H^1 norm relative errors of the STD

	L^2 norm	H^1 norm
$N_c = 8$	1.13×10^{-1}	2.77×10^{-1}
$N_c = 16$	5.22×10^{-2}	1.46×10^{-1}
$N_c = 32$	1.35×10^{-2}	9.88×10^{-1}
$N_c = 64$	4.99×10^{-3}	5.00×10^{-2}

Example 5.3. We consider the SPDE (5.8) on $D = [0, 1] \times [0, 1]$ with a high contrast random coefficient. The elliptic coefficient is given by a random high-contrast field. Specifically, $a(x, y, \omega)$ is a random linear combination of inclusion fields and channel fields plus a constant, i.e.,

$$a(x, y, \omega) = \sum_{i=1}^{3} \xi_i(\omega)\kappa_i(x, y) + 1.0,$$

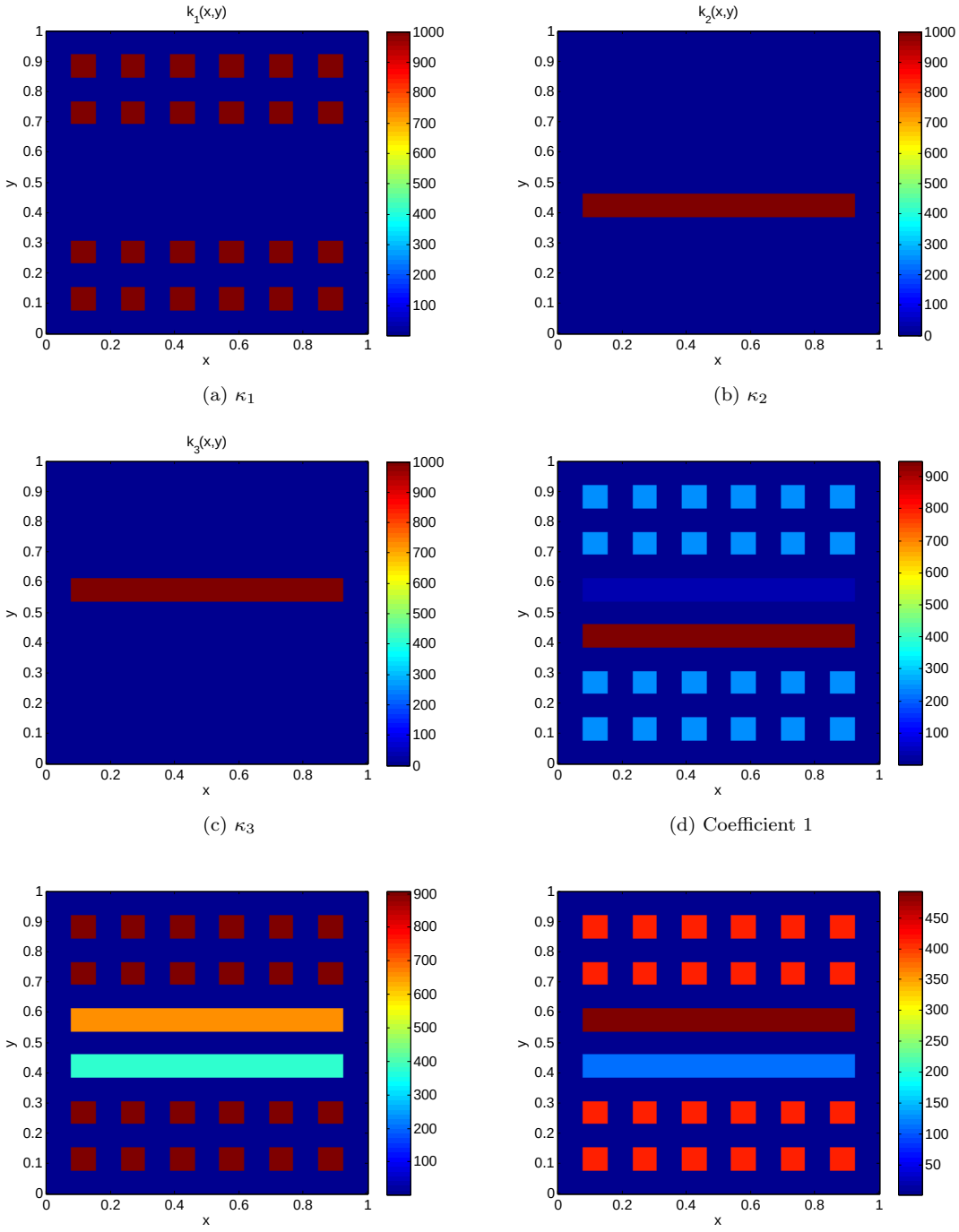
where $\{\xi_i\}_{i=1}^3$ are independent uniform random variables in $[0,1]$, $\kappa_1(x,y)$ is an inclusion field, and $\kappa_2(x,y)$ and $\kappa_3(x,y)$ are two channel fields. In Figure 5.5a-5.5c, we show the inclusion field and channel field, respectively, while in Figure 5.5d-5.5f we show three samples of the coefficient $a(x,y,\omega)$. One can see that the diversity of the random high-contrast coefficient. This presents a challenging test problem for the MsDSM. The implementation of the SCFEM and the MsDSM are exactly the same as in the previous examples.

Multiquery results in the online stage. The MsDSM solver using 207 sparse grids in the computation produces $m = 10$ modes in the data-driven stochastic basis. In the online stage we use them to solve the effective equation of the multiscale SPDE (5.8). In Table 5.8-5.9, we choose $f(x,y) = \sin(0.13\pi x + 0.12)\cos(0.49\pi y + 0.44)$ and show the relative errors of mean and STD of the MsDSM solution in L^2 norm and H^1 norm, respectively. One can observe that the MsDSM solution converges in both the L^2 norm and H^1 norm.

Compare the MsDSM solver with the SCFEM solver. For the SCFEM solver, it will take 1648.04 seconds to solve equation (5.8) with one specific forcing term. Thus in a multiquery problem, if we need to solve equation (5.8) with n different forcing terms, the total computational cost will be $t_{SCFEM} = 1648.04n$. If we choose $N_c = 64$ in the MsDSM solver, the offline computation will cost 7782.68 seconds, which includes the computational time for deriving effective SPDE, calculating correction term and constructing DSM basis for the effective SPDE. In the online stage of the MsDSM, it takes 2.95 seconds to compute one query, thus the total computational cost will be $t_{MsDSM} = 7782.68 + 2.95n$. We plot the total computational time in Fig.5.6. One can see that the MsDSM offers considerable computational savings over the SCFEM if we need to solve the same SPDE with more than five different forcing functions.

Table 5.8: Example 5.3 - L^2 and H^1 norm relative errors of the mean

	L^2 norm	H^1 norm
$N_c = 8$	1.39×10^{-1}	2.11×10^{-1}
$N_c = 16$	6.31×10^{-2}	1.58×10^{-1}
$N_c = 32$	3.97×10^{-2}	8.73×10^{-2}
$N_c = 64$	1.42×10^{-2}	4.59×10^{-2}



(a) κ_1 (b) κ_2

(c) κ_3 (d) Coefficient 1

(e) Coefficient 2 (f) Coefficient 3

Figure 5.5: Example 5.3 - κ_1, κ_2, κ_3 and some samples of the coefficient a

Table 5.9: Example 5.3 - L^2 and H^1 norm relative errors of the STD

	L^2 norm	H^1 norm
$N_c = 8$	1.38×10^{-1}	5.22×10^{-1}
$N_c = 16$	6.52×10^{-2}	3.13×10^{-1}
$N_c = 32$	4.05×10^{-2}	2.01×10^{-2}
$N_c = 64$	2.03×10^{-2}	8.05×10^{-2}

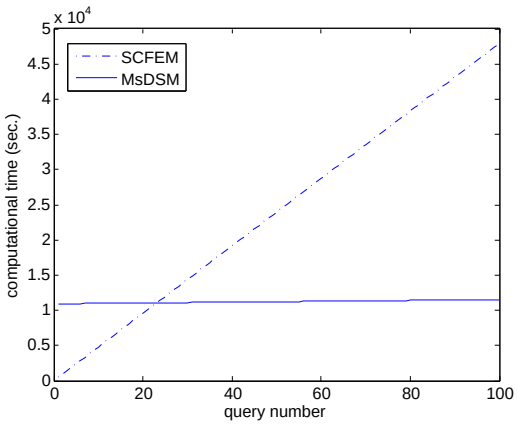


Figure 5.6: Example 5.3 - The computation time comparison

5.4.2 Comparison of the MsDSM and the DSM

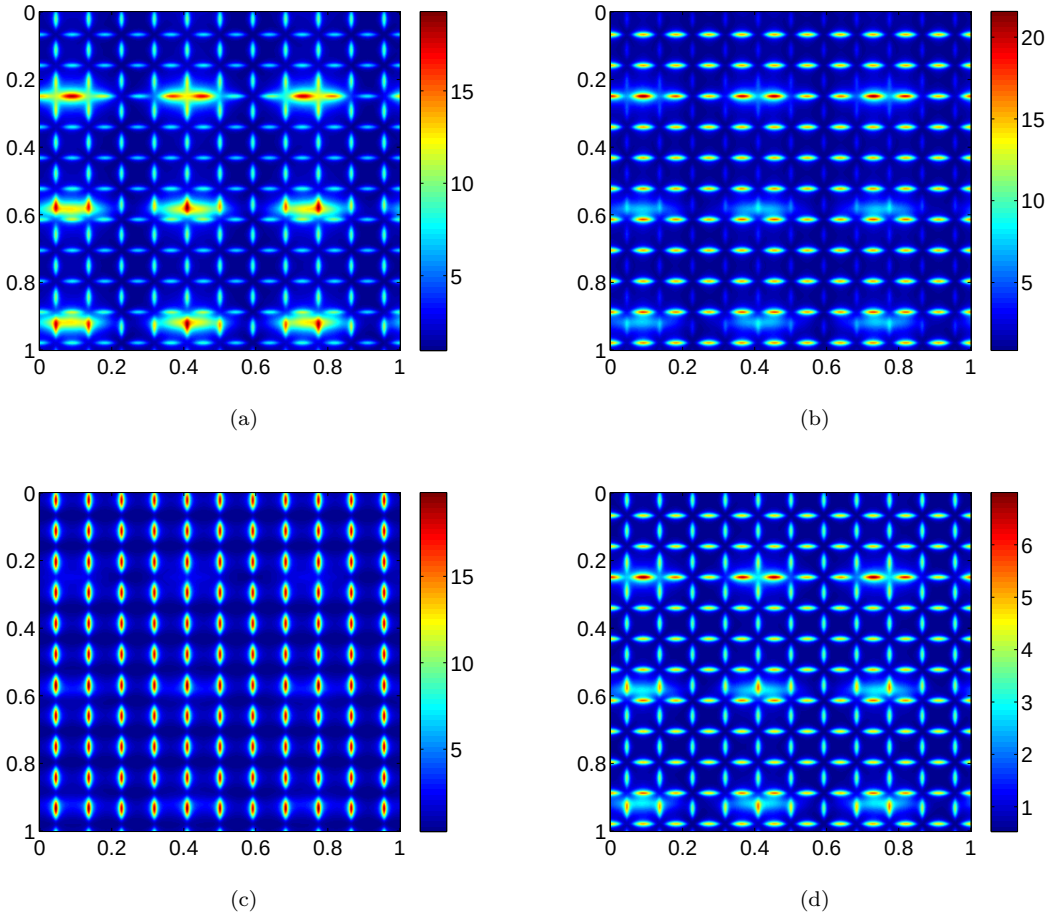
Here we compare the computational cost and accuracy of the MsDSM and the DSM in solving multiscale problems.

Example 5.4. We consider the SPDE (5.8) on $D = [0,1] \times [0,1]$ with the coefficient given by

$$\begin{aligned} a_(x,y,\omega) =& 0.1 + \xi_1(\omega)\frac{2 + 1.8\sin(2\pi x/\epsilon_1)}{2 + 1.8\sin(2\pi y/\epsilon_1)} \\ & \xi_2(\omega)\frac{2 + 1.8\sin(2\pi y/\epsilon_2)}{2 + 1.8\cos(2\pi x/\epsilon_2)} + \xi_3(\omega)\frac{2 + 1.8\cos(2\pi x/\epsilon_3)}{2 + 1.8\sin(2\pi y/\epsilon_3)}, \end{aligned}$$

where $\epsilon_1 = 1/3$, $\epsilon_2 = 1/11$ and $\epsilon_3 = 1/19$, and $\{\xi_i\}_{i=1}^3$ are independent uniform random variables in $[0,1]$. See Figure 5.7.

In our computations, we use the standard FEM to discretize the spatial dimension. We choose a 384×384 fine mesh to well resolve the spatial dimension of the stochastic solution $u(x,y,\omega)$. We choose level six sparse grids in the discretization of stochastic dimension, which has 135 points. The reference solution is obtained by using higher-level sparse grids. The coarse mesh of the MsDSM is chosen to be 64×64. We



(a) (b)

(c) (d)

Figure 5.7: Example 5.4 - Some samples of the coefficient a

implement the DSM on a 384×384 fine mesh and 64×64 coarse mesh, respectively.

The MsDSM generates $m = 7$ modes in the data-driven stochastic basis, while the DSM generates $m = 9$ modes. In the online stage we use them to solve the effective equation of the multiscale SPDE (5.8). In Table 5.10-5.11, we choose $f(x, y) = \sin(3.72\pi x + 0.35)\cos(2.74\pi y + .98)$ and show the relative errors of mean and STD of the MsDSM and DSM solution in L^2 norm and H^1 norm, respectively. Here $DSMf$ and $DSMc$ denote the DSM solution obtained on the fine and coarse grids, respectively. We conclude that the accuracy of the MsDSM is comparable with the one obtained by the DSM on a fine mesh grid. In addition, applying the DSM on a coarse mesh grid without any numerical upscaling will generate large errors in the numerical solution.

Table 5.10: Example 5.4 - L^2 and H^1 norm relative errors of the mean

	L^2 norm	H^1 norm
MsDSM	4.32×10^{-2}	1.45×10^{-1}
DSMc	7.16×10^{-3}	4.32×10^{-2}
DSMf	2.26×10^{-3}	1.06×10^{-2}

Table 5.11: Example 5.4 - L^2 and H^1 norm relative errors of the STD

	L^2 norm	H^1 norm
MsDSM	4.51×10^{-2}	1.56×10^{-1}
DSMc	1.15×10^{-2}	4.98×10^{-1}
DSMf	8.36×10^{-3}	3.01×10^{-1}

For the SCFEM solver, it will take 1132.94 seconds to solve equation (5.8) with one specific forcing term. Thus, in a multiquery problem, if we need to solve equation (5.8) with n different forcing terms, the total computational cost will be $t_{SCFEM} = 1132.94n$. The offline computation of the MsDSM and the DSM cost 3254.17 and 2898.32 seconds, respectively. In the online stage of the MsDSM, it takes 1.89 seconds to compute one query, thus the total computational cost will be $t_{MsDSM} = 3254.17 + 1.89n$. For the DSM solver on a fine grid, it takes 33.29 seconds to compute one query, thus the total computational cost will be $t_{MsDSM} = 2898.32 + 33.29n$.

It turns out that both the MsDSM and the DSM offer considerable computational savings over the SCFEM, if we need to solve the same SPDE with more than that of three different forcing functions. The offline cost of the MsDSM is more expensive

than the DSM, since we have to derive the effective equation. However, the online cost will be much cheaper than that of the DSM, because we solve the effective equation on a coarse grid.

Example 5.5. Finally, we consider the SPDE (5.8) on $D = [0,1] \times [0,1]$ with the coefficient in the same form as in the last example. However, here we choose $\epsilon_1 = 1/3$, $\epsilon_2 = 1/19$ and $\epsilon_3 = 1/65$. We choose a 1024×1024 fine mesh to well resolve the spatial dimension of the stochastic solution $u(x, y, \omega)$. We choose level six sparse grids in the discretization of stochastic dimension, which has 135 points. The reference solution is obtained by using higher-level sparse grids. In this example, due to memory overflow, the DSM easily breaks down. However, MsDSM still works owing to the upscaling in the physical dimension.

The MsDSM solver using 135 sparse grids in the computation produces $m = 8$ modes in the data-driven stochastic basis. In the online stage we use them to solve the effective equation of the multiscale SPDE (5.8). In Table 5.12, we choose $f(x, y) = \sin(1.37\pi x + 0.77)\cos(3.91\pi y + 0.11)$ and show the relative errors of mean and STD of the MsDSM solution in L^2 norm and H^1 norm, respectively.

Table 5.12: Example 5.5 - L^2 and H^1 norm relative errors of the solution

	L^2 norm	H^1 norm
mean	1.25×10^{-2}	3.24×10^{-2}
STD	1.52×10^{-2}	4.66×10^{-2}

For the SCFEM solver, it will take 18620.01 seconds to solve equation (5.8) with one specific forcing term $f(x, y, \theta)$. Thus in a multiple query problem, if we need to solve equation (5.8) with n different forcing term $f(x, y, \theta)$, the total computational cost will be $t_{SCFEM} = 18620.01n$. If we choose $N_c = 64$ in the MsDSM solver, the offline computation will cost 49258.59 seconds, which includes the computational time for deriving effective SPDE, calculating correction term and constructing DSM basis for the effective SPDE. In the online stage of the MsDSM, it takes 18.25 seconds to compute one query, thus the total computational cost will be $t_{MsDSM} = 49258.59 + 18.25n$. MsDSM offers considerable computational savings over the SCFEM, if we need to solve the same SPDE with more than three different forcing functions. We conjecture that the time model obtained in Section 5.2 may still be valid for the

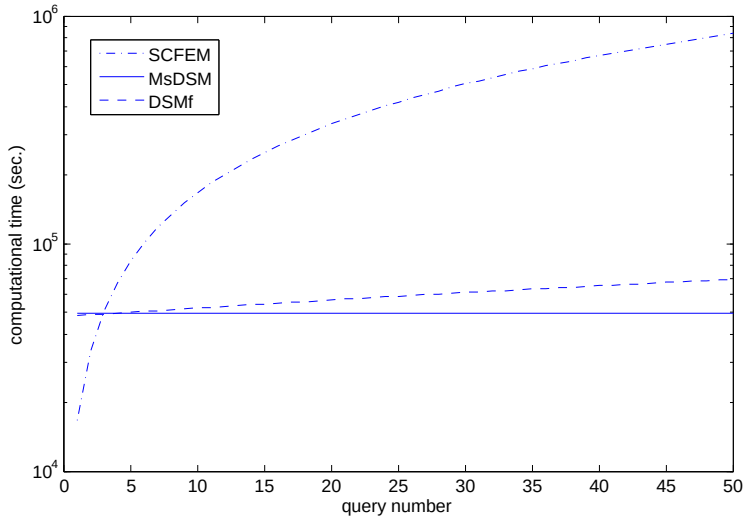


Figure 5.8: Example 5.5 - The computation time comparison

DSM. Actually, due to the fill-in, the real computation time and memory cost will be larger. The total computational cost for the DSM can be extrapolated as $t_{DSM} = 47700.90 + 438.62n$. We plot the total computational time in Figure 5.8. One can observe that the MsDSM offers huge savings over other methods in solving multiscale problems.

Chapter 6

Model reduction based multiscale multilevel Monte Carlo method for elliptic PDEs with random coefficients

6.1 Multilevel schemes for the effective stochastic equations

In this chapter, we also consider the following elliptic SPDE

$$\begin{cases} -\nabla \cdot (a(x,\omega)\nabla u(x,\omega)) = f(x), & x \in D, \omega \in \Omega, \\ u(x,\omega) = 0, & x \in \partial D, \omega \in \Omega. \end{cases} \tag{6.1}$$

We are interested in the expected value of some functional of the solutions, which could be the mean and high-order moments. In general, we could approximate the expectations by the standard Monte Carlo method (MC). For example, the mean of the solution $\mathbb{E}[u(x,\omega)]$ could be obtained by

$$Y = \frac{1}{N}\sum_{i=1}^{N} u_h(x,\omega_i), \tag{6.2}$$

where ω_i is the i-th example in N independently picked samples, and u_h is the numerical solution with mesh size h. Thus, we have

$$\mathbb{E}[Y] = \mathbb{E}[u_h(x,\omega)], \tag{6.3}$$

and

$$\mathrm{Var}[Y] = \frac{1}{N}\mathrm{Var}[u_h(x,\omega)]. \tag{6.4}$$

We define the mean square error of $e(Y)$ as

$$\begin{aligned} e(Y)^2 &= \int \mathbb{E}\left[(Y - \mathbb{E}[u(x,\omega)])^2\right] \mathrm{d}x \\ &= \int \mathbb{E}\left[(Y - \mathbb{E}[Y] + \mathbb{E}[Y] - \mathbb{E}[u(x,\omega)])^2\right] \mathrm{d}x \\ &= \int (\mathbb{E}[Y] - \mathbb{E}[u(x,\omega)])^2 \mathrm{d}x + \int \mathrm{Var}[Y] \mathrm{d}x \\ &= \int (\mathbb{E}[u_h(x,\omega)] - \mathbb{E}[u(x,\omega)])^2 \mathrm{d}x + \frac{1}{N}\int \mathrm{Var}[u_h(x,\omega)] \mathrm{d}x. \end{aligned} \tag{6.5}$$

The first term in the above equation gives the bias error introduced by the numerical discretization at mesh size h, while the second represents the sampling errors, and decay inversely with the number of samples. Thus we must choose the mesh size h to be fine enough to control the bias error, and many realizations are required to reduce sampling errors. For the multiscale stochastic problem, it is very time-consuming to obtain each realization. Therefore, the Monte Carol method is extremely expensive to obtain accurate results.

However, as we discussed before, we could construct an effective stochastic equation for each sample ω

$$\begin{cases} -\nabla \cdot (a^*(x,\omega)\nabla u^*(x,\omega)) = f(x), & x \in D, \omega \in \Omega, \\ u^*(x,\omega) = 0, & x \in \partial D, \omega \in \Omega. \end{cases} \tag{6.6}$$

The effective coefficient is given by

$$a^*(x,\omega) = a(x,\omega)(I + \frac{\partial \chi}{\partial x}(x,\omega)\frac{\partial x}{\partial g}(x,\omega)), \tag{6.7}$$

where $F = g + \chi$ and F is the associated harmonic coordinates.

We have already proven that u^* (or $u^* + \chi^T \frac{\partial x}{\partial g}\nabla u^*$) is a good approximation of u.

Thus we could use the following estimator Y^* instead of Y

$$Y^* = \frac{1}{N}\sum_{i=1}^{N} u_h^*(x,\omega_i). \tag{6.8}$$

Since u^* is smooth, we can pick the mesh size h much larger than the original mesh size, which would save a lot for each realization.

To further reduce the computational cost, we need to accelerate the Monte Carlo method. The multilevel Monte Carlo (MLMC) method [32] is a novel variance reduction technique for the standard Monte Carlo method. It exploits the linearity of expectation, by expressing the quantity of interest on the finest spatial grid in terms of the same quantity on a relatively coarse grid and correction terms. We will apply the multilevel Monte Carlo scheme to the effective equation (6.6).

We first divide the physical domain D into a number of nested coarse mesh grids, i.e., $D_{h_0} \subset \ldots \subset D_{h_{l-1}} \subset D_{h_l} \ldots \subset D_{h_L}$. Here $h_l = h_0 2^{-l}$ is the l-th level grid size ($l = 0, 1, \ldots, L$) and h_0 is the coarsest level mesh size. Denote $\mathbb{E}[u_l^*(x,\omega)] = \mathbb{E}[u_{h_l}^*(x,\omega)]$ to be the mean of the numerical solution on mesh size h_l. Linearity of the expectation operator implies that

$$\mathbb{E}[u_L^*(x,\omega)] = \mathbb{E}[u_0^*(x,\omega)] + \sum_{l=1}^{L} \mathbb{E}[u_l^*(x,\omega) - u_{l-1}^*(x,\omega)]. \tag{6.9}$$

Here $\mathbb{E}[u_L^*(x,\omega)]$ is the expectation of $u_h^*(x,\omega)$ as in equation (6.8). The key point is to avoid estimating $\mathbb{E}[u_L^*(x,\omega)]$ on the finest level; instead, we aim to estimate the expectation on the coarsest level, plus a sum of corrections adding the difference in expectation between simulations on consecutive levels. The multilevel idea is to independently estimate each of these expectations such that the overall variance is minimized for a fixed computational cost. Thus we have

$$Y^* = \sum_{l=0}^{L} Y_l^*, \tag{6.10}$$

where

$$Y_l^* = \frac{1}{N_l}\sum_{i=1}^{N_l}(u_l^*(x,\omega_i^{(l)}) - u_{l-1}^*(x,\omega_i^{(l)})), \quad k = 1,\ldots,L, \tag{6.11}$$

$$Y_0^* = \frac{1}{N_0}\sum_{i=1}^{N_0} u_0^*(x,\omega_i^{(0)}). \tag{6.12}$$

Here N_l is the number of Monte Carlo simulations at the l-th level. Y_0^* is the coarsest level estimator, while Y_l^* $(l = 1,\ldots,L)$ measures the fluctuations of l-th and $(l-1)$-th level. It is important to note that we have used the same random samples $\omega_i^{(l)}$ to estimate the quantity Y_l^*.

Simple calculations show that $e(Y^*)$ satisfies the following equation

$$\begin{aligned}
e(Y^*)^2 &= \int \mathbb{E}\left[(Y^* - \mathbb{E}[u(x,\omega)])^2\right]\mathrm{d}x \\
&= \int \mathbb{E}\left[(Y^* - \mathbb{E}[Y^*] + \mathbb{E}[Y^*] - \mathbb{E}[u(x,\omega)])^2\right]\mathrm{d}x \\
&= \int (\mathbb{E}[Y^*] - \mathbb{E}[u(x,\omega)])^2\mathrm{d}x + \int \mathrm{Var}[Y^*]\mathrm{d}x \\
&= \int (\mathbb{E}[u_L^*(x,\omega)] - \mathbb{E}[u(x,\omega)])^2\mathrm{d}x \\
&+ \sum_{l=1}^{L}\frac{1}{N_l}\int \mathrm{Var}[u_l^*(x,\omega) - u_{l-1}^*(x,\omega)]\mathrm{d}x + \frac{1}{N_0}\int \mathrm{Var}[u_0^*(x,\omega)]\mathrm{d}x.
\end{aligned} \tag{6.13}$$

The first term also gives the bias error introduced by the numerical discretization, and the second represents the sampling errors. As we have already seen, we could choose the mesh size h_L to be coarse and u_L^* is still a good approximation of u. If the variance $\mathrm{Var}[u_l^*(x,\omega) - u_{l-1}^*(x,\omega)]$ gets smaller and smaller and the mesh size h_l gets finer and finer, the samples needed for the Monte Carlo simulation are fewer and fewer, which is just the case for our effective equations. The reduction in cost associated with the multilevel Monte Carlo method over the Monte Carlo method is due to the fact that most of the uncertainty can be captured on the coarse grids ($h >> h_L$), so the number of realizations needed on the grid ($h = h_L$) is greatly reduced. We will show the computational cost reduction by several numerical examples.

6.2 Stiffness matrices assembling

Applying multilevel schemes on relatively coarse mesh grids could save us a lot of time. However, the implementation of such schemes is not so straightforward. As we mentioned, we must compute the harmonic coordinates to get the effective coefficients for one random realization. However, we cannot compute all the effective coefficients beforehand. First, for the Monte Carlo simulation, we do not know which random samples will be chosen and thus cannot compute the corresponding harmonic coordinates. Second, even if we use stochastic collocation points on a certain level, it would be too time-consuming to compute the harmonic coordinates for all the fixed points since we need to solve the physical PDE d times on a very fine mesh (d is the dimension of the physical space).

There is another difficulty in computing the numerical solutions on each mesh grid level. On mesh grid $h = h_l$, we need to discretize the effective coefficient $a_l^*(x, \omega)$ by some numerical method (such as standard finite element method) to obtain the stiffness matrix $A_l^*(x, \omega)$. It will take a lot of time if we do it for each random sample. Hence, to accelerate the online computations, we must come up with some efficient method to assemble the stiffness matrix $A_l^*(x, \omega)$ on mesh grid $h = h_l$.

We propose a Karhunen-Loève expansion based offline-online method to conquer the difficulties. During the offline stage, we choose some random samples ω from Monte Carlo simulation or stochastic collocation points, and compute the corresponding harmonic coordinates F. Then for each mesh grid h_l ($l = 0, \ldots, L$), we compute the effective coefficients $a_l^*(x, \omega)$ and the stiffness matrices $A_l^*(x, \omega)$. Note that for each mesh grid, h_l is relatively large, so the stiffness matrices $A_l^*(x, \omega)$ are relatively small in size and they possess the sparse structures. After that, we perform Karhunen-Loève expansion on $A_l^*(x, \omega)$ ($l = 0, \ldots, L$) and truncated it at m_l terms

$$A_l^*(x, \omega) \approx \sum_{i=0}^{m_l} \sqrt{\lambda_{l,i}} B_{l,i}(\omega) \Phi_{l,i}(x). \tag{6.14}$$

During the online stage, when we have a random sample $\tilde{\omega}$ (which might be different from any of the samples in the offline stage), we could get $B_{l,i}(\omega)$ by some

interpolation techniques. The stiffness matrix $A_l^*(x,\tilde{\omega})$ is easily computed by

$$A_l^*(x,\tilde{\omega}) \approx \sum_{i=0}^{m_l} \sqrt{\lambda_{l,i}} B_{l,i}(\tilde{\omega}) \Phi_{l,i}(x). \tag{6.15}$$

We will show by some numerical examples that the above estimation of the stiffness matrices is accurate provided that enough random samples are computed during the offline stage. For example, if we use the 12th-level stochastic collocation points for 3 random variables, 7th-level stochastic collocation points are needed in the offline stage. If we use 10000 Monte Carlo random samples for 15 random variables, 1000 Monte Carlo random samples are needed.

6.3 Complete algorithm

We provide the complete algorithm of the model reduction based multiscale mutilevel Monte Carol method as follows.

Algorithm 6.1. Offline Stage

1: Partition the domain D into a number of nested mesh grid blocks, i.e., $D_0 \subset \ldots \subset D_{l-1} \subset D_l \ldots \subset D_L$. The grid size of l-th level is $h_l = h_0 2^{-l}$, and h_0 is the coarsest level mesh size.
2: Pick Q random samples $\{\omega_j\}_{j=1}^{Q}$.
3: **for** $j = 1 \to Q$ **do**
4: For each ω_i, compute the corresponding harmonic coordinates F.
5: **for** $l = 0 \to L$ **do**
6: Decompose $F = g + \chi$ on mesh grid h_l.
7: Compute the effective coefficient $a_l^*(x,\omega_j)$ and the stiffness matrix $A_l^*(x,\omega_j)$.
8: **end for**
9: **end for**
10: Make Karhunen-Loève expansion on A_l^* $(l = 0,\ldots,L)$, truncate it as equation (6.14) and save the relevant data.

Algorithm 6.2. Online stage

1: **for** $l = 0 \to L$ **do**

Pick Q_l random samples $\{\omega_{j,l}\}_{j=1}^{Q_l}$.
for $j = 1 \to Q_l$ **do**
  Obtain $B_{l,i}(\omega_{j,l})$ by some interpolation method, $i = 1, \ldots, m_l$.
  Assemble the stiffness matrix A_l^* by equation (6.15).
  Solve for $u_l^*(x, \omega_{j,l})$.
end for
end for
Adopt the multilevel scheme formulation to calculate expected values of a functional of u^*.

6.4 Computational complexity analysis

We consider the standard Monte Carlo method as the benchmark in this section. Suppose we want to solve a two-dimensional multiscale problem, and the computational domain is $D = [0, 1] \times [0, 1]$. For such a problem, we need a fine mesh grid to resolve the small scales in physical space. Denote N_f to be the number of points in one direction, and for the multiscale problems, we usually take $N_f = 256$, $N_f = 384$ or $N_f = 512$. Let Q_{MC} be the number of random samples for Monte Carlo simulations. We know that the computational time of solving one linear systems with N unknowns is $O(N^{1.5})$. So the total time cost for the Monte Carlo method is approximately

$$t_{MC} \approx cQ_{MC}N_f^3, \tag{6.16}$$

where c is the constant associated with solving the linear systems.

6.4.1 Offline computational cost

For the offline stage, we have Q random samples as in Algorithm 6.1. For each sample, we need to solve the linear equation twice on the finest mesh grid N_f to get the harmonic coordinates. The time cost for this part is

$$t_1 \approx 2cQN_f^3. \tag{6.17}$$

In next steps, for each $l = 0, \ldots, L$, we will decompose the harmonic coordinates and compute the effective coefficients. This part only involves some matrix multiplications and so on, so the time cost is almost negligible. Then we need to generate the stiffness matrix on the coarse mesh grid D_l. We find that, when the mesh size is relatively large, it takes almost a constant time t_0 to generate the stiffness matrix for each D_l. The constant time t_0 is less than one-fourth of the time of solving for one solution on the finest mesh grid N_f. So the other significant time cost is

$$t_2 \approx (L+1)t_0 Q < \frac{L+1}{4} c Q N_f^3. \tag{6.18}$$

If we take $L = 3$ and $Q = \beta Q_{MC}$ ($\beta < 1$), the total offline computation time cost would be

$$t_{offline} \approx t_1 + t_2 < 3\beta c Q_{MC} N_f^3 = 3\beta c t_{MC}. \tag{6.19}$$

6.4.2 Online computational cost

For the online stage in Algorithm 6.2, the time cost in obtaining the value of $B_{l,i}(\omega_{j,l})$ and assembling the stiffness matrix is almost negligible compared with the time cost in solving for the solutions on the coarse mesh grid D_l. However, the coarse mesh size h_l is much larger than the finest mesh size h_f ($h_l < \frac{1}{10} h_f$), and the computational time of solving one linear system with N unknowns is $O(N^{1.5}) = O(h^{-3})$. So the online computational time cost is quite small compared with the offline stage or the standard Monte Carlo method. We will see from the numerical examples that it is less than 2% of the standard Monte Carlo method.

6.5 Numerical examples

In this section, we perform numerical experiments to test the performance and accuracy of the proposed MsMLMC method. We also demonstrate the computational efficiency of MsMLMC over the traditional method, such as the Monte Carlo or stochastic collocation finite element method in solving SPDEs with multiscale features.

Example 6.1. We consider the following stochastic elliptic equation with multiple

scales on $D = [0,1] \times [0,1]$. The multiscale coefficient is given by

$$a(x,y,\omega) = 0.1 + \frac{\xi_1(\omega)}{2 + P\sin(2\pi(x-y)/\epsilon_1)} + \frac{\xi_2(\omega)}{4 + P(\sin(2\pi x/\epsilon_2) + \sin(2\pi y/\epsilon_2))} + \frac{\xi_3(\omega)}{10(2 + P\sin(2\pi(x-0.5)/\epsilon_3))(2 + P\sin(2\pi(y-0.5)/\epsilon_3))},$$

where $P = 1.9$, $\epsilon_1 = \frac{1}{3}$, $\epsilon_2 = \frac{1}{27}$ and $\epsilon_3 = \frac{1}{51}$, and $\{\xi_i\}_{i=1}^3$ are independent uniform random variables in $[0,1]$.

In our computations, we use the standard finite element method. We choose a 512×512 fine mesh grid to well resolve the spatial scales of the stochastic solution $u(x,y,\omega)$. In previous sections, we mainly discuss the multilevel idea for variance reduction in the Monte Carlo method. Actually, the same idea can be applied in the stochastic collocation method. First, we conduct a convergence study, and find that the relative errors of mean and STD between the solutions obtained by level 11 sparse grids in the stochastic collocation method and higher level sparse grids are smaller than 0.1% both in L^2 and H^1 norm. Therefore, we choose the solution obtained by level 11 sparse grids with 1135 points as the reference solution.

To implement the multiscale multilevel stochastic collocation method (MsMLSC), the coarsest mesh grid of MsMLSC is chosen as 4×4, i.e., $h_0 = \frac{1}{4}$, and $h_l = \frac{h_0}{2^l}$, $l = 0, ..., L_0$ is the l-th level grid size with the coarsening factor 2. The coarsest grid expectation solution is obtained by level 11 sparse grids, and the difference of the expectation solutions between l and $l-1$ grids are obtained by level $11 - 2j$ sparse grid. In practical computation, one can relocate the sample number according to the variance decay property, which will be elaborated in the next example.

We choose $f(x,y) = \sin(0.42\pi x + 0.11)\cos(0.26\pi y + 0.43)$. In Table 6.1-6.2, we show the relative errors of mean and STD of the solution in L^2 norm and H^1 norm. The MsMLSC method gives very accurate results.

Table 6.1: Example 6.1 - L^2 norm relative errors of the solution

	mean	STD
$N_c = 4$	1.15×10^{-1}	1.46×10^{-1}
$N_c = 8$	3.17×10^{-2}	5.56×10^{-2}
$N_c = 16$	1.05×10^{-2}	4.24×10^{-1}
$N_c = 32$	5.11×10^{-3}	3.96×10^{-2}

Table 6.2: Example 6.1 - H^1 norm relative errors of the solution

	mean	STD
$N_c = 4$	3.95×10^{-1}	3.27×10^{-1}
$N_c = 8$	2.03×10^{-1}	1.99×10^{-1}
$N_c = 16$	9.72×10^{-2}	1.16×10^{-1}
$N_c = 32$	4.79×10^{-2}	9.22×10^{-2}

For the stochastic collocation method, it will take 6483.32 seconds to solve the original equation (6.1) with one specific forcing term. Thus in a multiquery problem, if we need to solve the equation with n different forcing terms, the total computational cost will be $t_{SC} = 6483.32n$. In our MsMLSC method, the offline computation will cost 2536.21 seconds, which includes the computational time of solving one cell problem, constructing the reduced basis, and computing the fixed data structure for the global stiffness matrix. In the online stage of the MsMLMC, it takes 73.55 seconds to compute each query, thus, the total computational cost will be $t_{MsMLSC} = 2536.21 + 73.55n$. We can see that the MsMLSC offers considerable computational savings over the stochastic collocation method if we need to solve the original SPDE with more than seven different forcing functions. Even for one forcing term, it is still superior.

Example 6.2. We consider the SPDE (6.1) with high-dimensional stochastic input and multiscale features on $D = [0, 1] \times [0, 1]$. The coefficient is given by

$$a(x, y, \omega) = \sum_{i=1}^{15} \xi_i(\omega) \frac{2 + P_i \sin(2\pi x/\epsilon_i)}{2 + Q_i \cos(2\pi y/\epsilon_i)},$$

where $\{\xi_i\}$ are independent uniform random variables in $[0, 1]$, P_i, $Q_i \in (1.8, 1.9)$ are randomly generated, and $(\epsilon_1, ..., \epsilon_{15}) = (\frac{1}{n_1}, ..., \frac{1}{n_{15}})$, where the integers $3 \leq n_j \leq 31$ are randomly generated.

The stochastic collocation method is computationally prohibitive due to the curse of dimensionality. We implement our method in the Monte Carlo method setting. We use the standard finite element method to discretize the spatial dimension. We choose a 384×384 fine mesh grid to well resolve the spatial dimension of the stochastic solution $u(x, y, \omega)$. Due to the tremendous computational cost, we use the Monte Carlo method with 10^4 samples to calculate the reference solution.

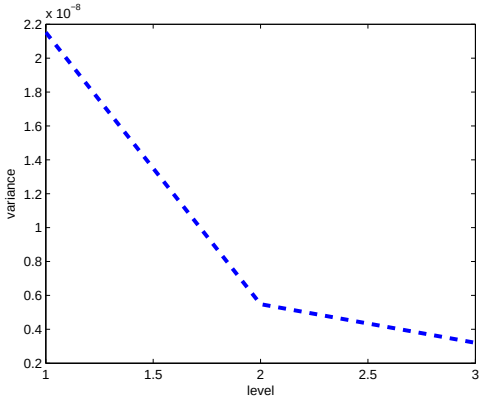


Figure 6.1: Example 6.2 - Variance decay on the coarse mesh grids

To implement the MsMLMC, the coarsest mesh grid is chosen as 4×4, i.e., $h_0 = \frac{1}{4}$, and $h_l = \frac{h_0}{2^l}$, $l = 0, ..., L_0$ is the l-th level grid size with the coarsening factor 2. The coarsest grid expectation solution is obtained by 10^4 Monte Carlo samples, and the difference of the expectation solutions between l and $l-1$ grids are obtained by $10^4 \times c_l$ Monte Carlo samples, where c_l is a scaling factor obtained by the variance decay property.

We choose $f(x, y) = \sin(0.28\pi x + 0.21)\cos(0.35\pi y + 0.03)$. In Table 6.3, we show the relative errors of mean and STD of the solution in L^2 norm and H^1 norm. The MsMLMC method gives very accurate results. Figure 6.1 shows the variance decay result for the solution.

Table 6.3: Example 6.2 - L^2 norm relative errors of the solution

	mean	STD
$N_c = 4$	8.23×10^{-1}	4.77×10^{-1}
$N_c = 8$	2.45×10^{-1}	2.84×10^{-1}
$N_c = 16$	6.77×10^{-2}	1.54×10^{-1}
$N_c = 32$	1.70×10^{-2}	7.32×10^{-2}

For the stochastic collocation method, it will take 33942.57 seconds to solve the original equation (6.1) with one specific forcing term. Thus in a multiquery problem, if we need to solve the equation with n different forcing terms, the total computational cost will be $t_{MC} = 33942.57n$. In our MsMLMC method, the offline computation will cost 13736.21 seconds, which includes the computational time of solving cell problem, constructing the reduced basis, and computing the fixed data structure

for the global stiffness matrix. In the online stage of the MsMLMC, it takes 136.16 seconds to compute each query, thus, the total computational cost will be $t_{MsMLMC} = 13736.21 + 136.16n$. We can see that the MsMLMC offers considerable computational savings over the stochastic collocation method if we need to solve the original SPDE with more than seven different forcing functions.

Example 6.3. In this example, the coefficient does not have the affine form and is given by

$$a(x, y, \omega) = \exp\left(\sum_{i=1}^{10} \xi_i(\omega) w_i \phi_i(x, y)\right),$$

where $\{\xi_i\}$ are independent uniform random variables in $[-\sqrt{3}, \sqrt{3}]$,
$\phi_i(x, y) = \frac{1}{10}(\sin(2\pi x/w_i) + \cos(2\pi y/w_i))(\sin(2\pi y/w_i) + \cos(2\pi x/w_i))$, and
$(w_1, ..., w_{10}) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{11}, \frac{1}{13}, \frac{1}{17}, \frac{1}{19}, \frac{1}{23}, \frac{1}{29}\right)$.

We implement our method in the Monte Carlo method setting. We use the standard finite element method to discretize the spatial dimension. We choose a 256×256 fine mesh grid to well resolve the spatial dimension of the stochastic solution $u(x, y, \omega)$. Due to the tremendous computational cost, we use the Monte Carlo method with 8000 samples to calculate the reference solution.

To implement the MsMLMC, the coarsest mesh grid is chosen as 4×4, i.e., $h_0 = \frac{1}{4}$, and $h_l = \frac{h_0}{2^l}$, $l = 0, ..., L_0$ is the l-th level grid size with the coarsening factor 2. The coarsest grid expectation solution is obtained by 8000 Monte Carlo samples, and the difference of the expectation solutions between l and $l - 1$ grids are obtained by $8000 \times c_l$ Monte Carlo samples, where c_l is a scaling factor obtained by the variance decay property.

We choose $f(x, y) = 1$. In Table 6.4, we show the relative errors of mean and STD of the solution in L^2 norm and H^1 norm. The MsMLMC method gives very accurate results. Figure 6.2 shows the variance decay result for the solution.

For the stochastic collocation method, it will take 54823.45 seconds to solve the original equation (6.1) with one specific forcing term. Thus in a multiquery problem, if we need to solve the equation with n different forcing terms, the total computational cost will be $t_{MC} = 54823.45n$. In our MsMLMC method, the offline computation

Table 6.4: Example 6.3 - L^2 norm relative errors of the solution

	mean	STD
$N_c = 4$	8.51×10^{-2}	7.69×10^{-1}
$N_c = 8$	2.29×10^{-2}	3.54×10^{-1}
$N_c = 16$	5.76×10^{-3}	1.48×10^{-1}
$N_c = 32$	2.89×10^{-3}	6.32×10^{-2}

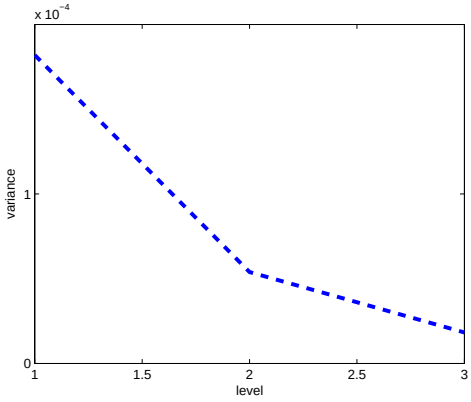


Figure 6.2: Example 6.3 - Variance decay on the coarse mesh grids

will cost 13736.21 seconds, which includes the computational time of solving cell problem, constructing the reduced basis, and computing the fixed data structure for the global stiffness matrix. In the online stage of the MsMLMC, it takes 179.35 seconds to compute each query, thus, the total computational cost will be $t_{MsMLMC} = 13736.21 + 179.35n$. We can see that the MsMLMC offers considerable computational savings over the stochastic collocation method if we need to solve the original SPDE with more than seven different forcing functions.

Chapter 7

Conclusions

Model reduction technique for PDEs. We have proposed a multiscale model reduction method for several standard types of elliptic, parabolic, hyperbolic, and convection-diffusion equations. A key ingredient of this method is to decompose the harmonic coordinate into a smooth part and a high oscillatory part of which the magnitude is small. The effective equation is derived by taking into the account of the interaction between the multiscale coefficient of the governing equation and the decomposition. One advantage of this approach is that we do not require scale separation or periodic structures. Another advantage is that our effective equation can be solved on a regular coarse mesh, and it is easy to implement. An efficient decomposition method has been proposed to decompose the harmonic coordinate into a smooth part plus a small remainder. Under some assumptions on the multiscale coefficient, we analyze the error between the effective solution and the original multiscale solution, and show that the error is small in the H^1 norm. Several numerical examples have been given to demonstrate the robustness and the accuracy of the proposed method.

Decomposition of harmonic coordinates. The decomposition method described above seems to work very well from our numerical experiments. One advantage of this approach is that it is very easy to implement. However, such decomposition may not work in all cases, and may not give the optimal result, especially when the harmonic coordinate, F, is not invertible. To overcome this difficulty, we are currently investigating an alternative approach based on optimization. More specifically, we would like to find a smooth g that lives in the linear finite element space

generated by the coarse mesh grid and minimizes the difference between g and F in some appropriate norm subject to the constraint that g is invertible. Formulating this problem as a convex optimization problem would be the key to make this method efficient. We will study this in the future.

Invertibility of harmonic coordinates. There are still some limitations of the model reduction method. One most challenging issue is how to perform the decomposition of the harmonic coordinate when the harmonic coordinate is not invertible due to irregular geometries or three dimensionality. Although our method does not require the harmonic coordinate, F, to be invertible, finding an optimal decomposition, $F = g + \chi$, such that g is smooth and invertible while keeping χ small becomes more challenging when F is not invertible in some local region. One way to overcome this difficulty is to apply our model reduction method locally instead of globally. By using a local mesh refinement, we can capture some nearly singular behavior of F by a locally well-resolved g and still keep the remainder χ small. Another way is to develop an optimization method to generate the optimal g iteratively as we mentioned. We can also use a residual error correction to further reduce the error if χ is not small in some localized region due to the degeneracy of F. These issues will be further investigated in our future work.

Finite difference discretization. The solution to the effective equation is smooth, while the coefficient is not smooth. For numerical implementations of our method on a coarse mesh grid, we need to take the average of the coefficient. We use the finite element method to take care of this work, since we are taking the average of the coefficient when we do integrations on the coarse mesh grid. To implement our method via other numerical methods, such as the finite difference method, we need to take the average before directly implementing the method. Recall that the smoothness of the solution comes from the fact that the effective equation has a non-divergence form. The coefficient is given by $a^* = a\frac{\partial F}{\partial x}\frac{\partial x}{\partial g}$, and $a\frac{\partial F}{\partial x}$ is divergence free. We must keep the divergence free condition when we take the average. Taking the average of a^* is equivalent to taking the average of $a\frac{\partial F}{\partial x}$, since the function g is smooth. Averaging $a\frac{\partial F}{\partial x}$ while keeping its divergence free property is not straightforward, and we are still investigating how to realize that efficiently, such as by projecting it onto

some divergence free basis functions.

Two model reduction methods for SPDEs. We combine the data-driven stochastic basis idea and multilevel Monte Carlo idea with the model reduction technique, and obtain the MsDSM and MsMLMC. The Karhunen-Loève expansion plays an important role. Both methods consist of offline and online stages. The offline computational cost for MsDSM is more expensive than MsMLMC, but the online computational cost for MsDSM is cheaper than MsMLMC. Also, the MsDSM provides the results with better accuracy than MsMLMC. It is important to point out that MsDSM involves the computation of global harmonic coordinates for all samples (sparse grids); thus, the memory consumption becomes a serious issue when the number of random variables is large. In this case, we adopt the MsMLMC to tackle these problems.

Improvement of MsMLMC. The MsMLMC is a robust method that can be used for many multiscale stochastic problems. We only need to compute the harmonic coordinates for a small amount of random samples, which makes the offline computational cost cheap. The interpolation technique used in the online stage makes the assembling of the stiffness matrices very efficient. However, the small amount of random samples and the interpolation technique makes MsMLMC less accurate than MsDSM. We are seeking other stiffness matrix assembling technique to overcome this difficulty.

Bibliography

[1] G. Alessandrini and V. Nesi. *Univalent σ-harmonic mappings*, Arch. Rational Mech. Anal. 158 (2001), 155-171.

[2] G. Allaire and R. Brizzi. *A multiscale finite element method for numerical homogenization.* SIAM MMS, 4(3) (2005), 790-812.

[3] A. Ancona. *Some results and examples about the behavior of harmonic functions and Green's funtions with respect to second order elliptic operators*, Nagoya Math. J. 165(2002), 123-158.

[4] T. Arbogast, G. Pencheva, M. F. Wheeler, and I. Yotov, *A multiscale mortar mixed finite element method*, SIAM MMS, 6(1)(2007), 319-346.

[5] B.V. Asokan and N. Zabaras. *A stochastic variational multiscale method for diffusion in heterogeneous random media*, J. Comput. Phys., 218:654-676, 2006.

[6] I. Babuška, G. Caloz, and E. Osborn. *Special finite element methods for a class of second order elliptic problems with rough coefficients.* SIAM J. Numer. Anal., 31 (1994), 945-981.

[7] I. Babuska, F. Nobile, and R. Tempone. *A stochastic collocation method for elliptic partial differential equations with random input data.* SIAM J. Numer. Anal., 45(3):1005-1034, 2007.

[8] I. Babuška and E. Osborn, *Generalized Finite Element Methods: Their Performance and Their Relation to Mixed Methods,* SIAM J. Numer. Anal., 20 (1983), pp. 510-536.

[9] A. Barth, C. Schwab and N. Zollinger. *MultiClevel Monte Carlo finite element method for elliptic PDEs with stochastic coefficients.* Report 2010-18, SAM Research Reports, ETH Zurich, 2010.

[10] A. Bensoussan, J.-L. Lions and G. Papanicolaou, *Asymptotic Analysis for Periodic Structure*, North-Holland, Amsterdam, 1978.

[11] H.J. Bungartz and M. Griebel. *Sparse grids*, Acta Numerica, 13:147-269, 2004.

[12] R.H. Cameron and W.T. Martin. *The orthogonal developement of non-linear functionals in series of Fourier-Hermite Functionals.* Annals of Mathmatics, 48(2):385-392, 1947.

[13] Y. Chen, L.J. Durlofsky, M. Gerritsen, and X.H. Wen, *A coupled local-global upscaling approach for simulating flow in highly heterogeneous formations*, Advances in Water Resources, 26 (2003), pp. 1041-1060.

[14] Z. Chen and T.Y. Hou, *A mixed multiscale finite element method for elliptic problems with oscillating coefficients*, Math. Comp., 72 (2002), 541-576.

[15] M.L. Cheng, T.Y. Hou, M. Yan and Z.W. Zhang. *A Data-driven Stochastic Method for elliptic PDEs with random coefficients.* Accepted by SIAM JUQ.

[16] M.L. Cheng, T.Y. Hou, Z.W. Zhang. *A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations I: Derivation and Algorithms.* Accepted by JCP, 2013. DOI:10.1016/j.jcp.2013.02.033.

[17] M.L. Cheng, T.Y. Hou, Z.W. Zhang. *A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations II: Adaptivity and Generalizations.* Accepted by JCP, 2013. DOI: 10.1016/j.jcp.2013.02.020.

[18] C. C. Chu, I. Graham, and T. Y. Hou, *A New multiscale finite element method for high-contrast elliptic interface problems,* Math. Comp., 79 (2010), 1915-1955.

[19] K.A. Cliffe, M.B. Giles, R. Scheichl and A.L. Teckentrup. *Multilevel Monte Carlo Methods and Applications to Elliptic PDEs with Random Coefficients,* Comput. Visual Sic., 14 (2011), 3-15.

[20] P. Dostert, Y. Efendiev, T. Y. Hou, and W. Luo, *Coarse gradient Langevin algorithms for dynamic data integration and uncertainty quantification.* J. Comput. Phys., 217:123-142, 2006.

[21] W. E and B. Engquist. *The heterogeneous multi-scale methods.* Comm Math. Sci., 1(1) (2003), 87-133.

[22] Y. Efendiev, J. Galvis, and T. Y. Hou, *Generalized multiscale finite element methods (GMsFEM)*, accepted by JCP, 2013.

[23] Y. Efendiev, J. Galvis, and X. H. Wu, *Multiscale finite element methods for high-contrast problems using local spectral basis functions*, J. Comp. Phys., 230 (2011), 937-955.

[24] Y. Efendiev, V. Ginting, T. Hou, and R. Ewing, *Accurate multiscale finite element methods for two-phase flow simulations*, J. Comp. Physics, 220 (1), pp. 155-174, 2006.

[25] Y. Efendiev and T. Hou, *Multiscale finite element methods. Theory and applications*, Springer, 2009.

[26] Y. Efendiev, T. Y. Hou, and W. Luo, *Preconditioning of Markov Chain Monte Carlo simulations using coarse-scale models.* SIAM J. Sci. Comput., 28(2):776-803, 2006.

[27] Y. Efendiev, T. Y. Hou, and X. H. Wu, *Convergence of a nonconforming multiscale finite element method*, SIAM J. Num. Anal., 37 (2000), 888-910.

[28] J. Galvis and Y. Efendiev, *Domain decomposition preconditioners for multiscale flows in high-contrast media: Reduced dimension coarse spaces*, SIAM MMS, 8 (2009), 1621-1644.

[29] B. Ganapathysubramanian and N. Zabaras, *Modelling diffusion in random heterogeneous media: Data-driven models, stochastic collocation and the variational multi-scale method*, J. Comput. Phys., 226:326-353, 2007.

[30] R.G. Ghanem and P.D. Spanos. *Stochastic Finite Elements: A Spectral Approach.* New York : Springer-Verlag, 1991.

[31] I. G. Graham, P. O. Lechner, and R. Scheichl, *Domain decomposition for multi-scale PDEs*, Numer. Math., 106(4):589-626, 2007.

[32] M.B. Giles. *Multilevel Monte Carlo Path Simulation.* Operations Research, 56(3):607-617, 2008.

[33] H. Hajibeygi, G. Bonfigli, M. A. Hesse, P. Jenny. *Iterative multiscale finite-volume method.* Journal of Computational Physics (2008), 227(19), 8604-8621.

[34] H.D. Han and Z.W. Zhang, *Multiscale tailored finite point method for second order elliptic equations with rough or highly oscillatory coefficients*, Comm. Math. Sci., 10(3) (2012), 945-976.

[35] S. Heinrich. *Multilevel Monte Carlo methods*, volume 2179 of Lecture Notes in Computer Science, 58-67, Springer, 2001.

[36] T.Y. Hou, W. Luo, B. Rozovskii, and H. Zhou. *Wiener Chaos expansions and numerical solutions of randomly forced equations of fluid mechanics.* J. Comput. Phys., 216(2):687-706, 2006.

[37] T.Y. Hou and X.H. Wu, *A multiscale finite element method for elliptic problems in composite materials and porous media*, J. Comput. Phys., 134 (1997), 169-189.

[38] T. Y. Hou, X. H. Wu, and Z. Cai, *Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients,* Math. Comput., 68 (1999), 913-943.

[39] T. Hughes, G. Feijoo, L. Mazzei, and J. Quincy, *The variational multiscale method - a paradigm for computational mechanics*, Comput. Methods Appl. Mech. Engrg, 166 (1998), 3-24.

[40] P. Jenny, S.H. Lee, and H. Tchelepi, *Multi-scale finite volume method for elliptic problems in subsurface flow simulation*, J. Comput. Phys., 187 (2003), 47-67.

[41] K. Karhunen. *Uber lineare Methoden in der Wahrscheinlichkeitsrechnung.* Ann. Acad. Sci. Fennicae. Ser. A. I. Math.-Phys. 37:1-79, 1947.

[42] O. Le Maitre. *Uncertainty propagation using Wiener-Haar expansions.* J. Comput. Phys., 197(1):28-57, 2004.

[43] M. Loeve. *Probability theory.* Vol. II, 4th ed. GTM. 46. Springer-Verlag. ISBN 0-387-90262-7.

[44] X. Ma and N. Zabaras. *An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations.* J. Comput. Phys., 228(8):3084-3113, 2009.

[45] A. J. Majda and M. Branicki. *Lessons in Uncertainty Quantification for Turbulent Dynamical System.* DCDS-A, 32:3133-3221, 2012.

[46] A. Maugeri, D. K. Palagachev, and L. G. Softova. *Elliptic and parabolic equations with discontinuous coefficients*, Mathematical Research, 109, Wiley-VCH, 2000.

[47] D. W. Mclaughlin, G. C. Papanicolaou and O. R. Pironneau. *Convetion of mircrostructure and related problems*, SIAM J. Appl. Math. 45(1985), 780-797.

[48] S. Moskow and M. Vogelius. *First-oder corrections to the homogenised eigenvalues of a periodic composite medium. A convergence proof*, Proc. Roy. Soc. Edinburgh. 127A(1997), 1263-1299.

[49] H.N. Najm. *Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics.* Ann. Rev. Fluid Mech., 41(1):35-52, 2009.

[50] B. K. Oksendal. *Stochastic Differential Equations: An Introduction with Applications.* Sixth edition. Berlin: Springer. 2003.

[51] H. Owhadi and L. Zhang, *Metric based up-scaling*, Comm. Pure Appl. Math., LX:675-723, 2007.

[52] H. Owhadi and L, Zhang. *Homogenization of parabolic equations with a continuum of space and time scales*, SIAM J. Numer. Anal. 46(2007), 1-36.

[53] C. Schwab, and R.A. Todor. *Karhunen-Loeve approximation of random fields by generalized fast multipole methods.* J. Comput. Phys., 217:100-122, 2006.

[54] G. Stefanou. *The stochastic finite element method: past, present and future*, ComputerMethods in AppliedMechanics and Engineering, 198(9-12):1031C1051, 2009.

[55] X. Wan and G.E. Karniadakis. *An adaptive multi-element generalized polynomial chaos method for stochastic differential equations.* J. Comput. Phys., 209(2):617-642, 2005.

[56] N. Wiener. *The homogeneous chaos.* Amer. J. Math., 60:897-936, 1938.

[57] D. Xiu and J.S. Hesthaven. *High-order collocation methods for differential equations with random inputs.* SIAM J. Sci. Comput., 27(3):1118-1139, 2005.

[58] D. Xiu and G.E. Karniadakis. *The Wiener-Askey polynomial chaos for stochastic differential equations.* SIAM J. Sci. Comput., 24(2):614-644, 2002.

[59] D. Xiu and G.E. Karniadakis. *Modeling uncertainty in flow simulations via generalized polynomial chaos.* J. Comput. Phys., 187(1):137-167, 2003.

[60] Z.W. Zhang, X. Hu, T.Y. Hou, G. Lin, and M. Yan, *An adaptive ANOVA-based data-driven stochastic method for elliptic PDE with random coefficients.* Submitted to Commun. in Comput. Phys.