Implicit Space-Time Domain Decomposition Methods for Stochastic Parabolic Partial Differential Equations

by

Cui Cong

B.A., Shandong University of Science and Technology, 2005

M.S., University of Colorado Boulder, 2011

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Mathematics

2013

This thesis entitled:
Implicit Space-Time Domain Decomposition Methods for Stochastic Parabolic Partial Differential Equations
written by Cui Cong
has been approved for the Department of Mathematics

Xiao-Chuan Cai

Karl Gustafson

Robert Goodrich

Congming Li

Robert Leben

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Cong, Cui (Ph.D., Mathematics)

Implicit Space-Time Domain Decomposition Methods for Stochastic Parabolic Partial Differential Equations

Thesis directed by Professor Xiao-Chuan Cai and Professor Karl Gustafson

In this thesis, we introduce and study parallel space-time domain decomposition methods for solving deterministic and stochastic parabolic equations. Traditional parallel algorithms solve parabolic problems time step by time step, *i.e.*, the simulation of the later time step is based on the solution of the earlier time step. Therefore, the parallelism is restricted to each time step, and the algorithms are purely sequential in time. Recently, there are several attempts to develop time-parallel algorithms, such as parareal, waveform relaxation, and space-time multigrid. In this thesis, we develop some overlapping Schwarz methods whose subdomains cover both space and time variables, and we show numerically that the methods work well for stochastic parabolic equations discretized with an implicit stochastic Galerkin method. The main components of the stochastic Galerkin method are Karhunen-Loève expansion and double orthogonal polynomials, which are used to decouple the stochastic parabolic problem into a sequence of deterministic parabolic equations. In order to solve the sequence of equations efficiently, one- and two-level Schwarz preconditioned recycling GMRES methods are carefully investigated such that some components of the methods are reused to maximize the benefit of the recycling Krylov subspace when a large number of linear systems are solved. The key elements of this approach include an ordering algorithm and two grouping algorithms. We present some experimental results obtained on a parallel computer with more than one thousand processors.

Dedication

To my husband, my daughter, my parents ...

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Xiao-Chuan Cai and Prof. Karl Gustafson for their continuous support of my Ph.D study and research. I deeply appreciate their helpful suggestions, patience, encouragement, and immense knowledge. Their guidance helped me throughout my research and writing this thesis.

My sincere thanks also go to Professors Robert Goodrich, Congming Li and Robert Leben for serving on my committee, reviewing my work, insightful comments, and hard questions.

I thank my wonderful labmates and colleagues: Yuqi Wu, Si Liu, Chao Yang, Chao Jin and Fande Kong for their helpful discussions.

Last but not least, I would like to thank my parents for giving birth to me and supporting me throughout my life, and my husband Hai Long, whose love and spiritual encouragement allowed me to finish this journey.

Contents

Chapter

1 Introduction 1

2 Space-time domain decomposition methods 7

3 Stochastic Galerkin method for parabolic problems 18

3.1 Weak form of the stochastic parabolic equation 18

3.2 Karhunen-Loève (KL) expansion 20

3.3 Double orthogonal basis 24

3.4 Discretization in ξ-space 25

3.5 Discretization in space and time 27

4 Recycling Krylov subspace method and grouping algorithms 29

4.1 Recycling Krylov subspace method 29

4.2 Ordering algorithm 31

4.3 Grouping algorithm for one-level RAS preconditioning 33

4.4 Grouping algorithm for two-level hybrid preconditioning 35

4.5 Numerical examples 37

5 Numerical experiments 53

5.1 Identifying the dimension of the recycling Krylov subsapce 53

5.2 Comparing several recycling strategies 56

5.3 Scalabilities study of the two-level hybrid preconditioning 67

6 Conclusions and future work 70

Appendix

Bibliography 72

Tables

Table

2.1 The extrema singular values and condition numbers for matrices A_s as s increases. 13

4.1 Reordering $C_{i_j,j}$ decreasingly by permutation, for all j, $0 \leq j \leq M$, $\{0, 1, \cdots, n_j\} \longrightarrow \{l_0^j, l_1^j, \cdots, l_{n_j}^j\}$. 32

5.1 Computing time (second) and average number of iterations (denoted by "aiter") for the one-level RAS preconditioned FGMRES without restart, $\delta = 8$, ILU($*$) and LU are subdomain solvers. 55

5.2 Computing time (second) and average number of iterations (denoted by "aiter") for the one-level RAS preconditioned FGMRES with restart (= 50), $\delta = 8$. 55

5.3 Computing time (second) and average number of iterations (denoted by "aiter") for the two-level hybrid preconditioned FGMRES without restart. $\delta = 8$, the coarse overlap $\delta_c = 0$. k_m is defined in (5.1). 56

5.4 Computing time (second) and average number of iterations (denoted by "aiter") of four schemes. 57

5.5 Average number of iterations for the two-level hybrid preconditioning with different mesh size, overlap size, and number of processors. The coarse mesh is 32×32. . . . 67

5.6 Computing time (second) per window size and average number of iterations (denoted by "aiter") for the two-level hybrid preconditioning with different window sizes. . . . 67

5.7 Computing time (second) and average numbers of iterations (denoted by "aiter") for the two-level hybrid preconditioning with different Δt. 68

Figures

Figure

2.1 The pattern of matrix A_s for a $8 \times 8 \times 4$ mesh, $s = 4$ is the number of time levels. The number of processors is 4. 16

4.1 The largest 4 eigenvalues of $C_a(x, x')$, $x, x' \in [0, 1]$. 39

4.2 The largest 11 eigenvalues of $C_a(x, x')$, $x, x' \in [0, 1]^2$. 39

4.3 The first eigenfunction of $C_a(x, x')$, $x, x' \in [0, 1]^2$. 40

4.4 The second eigenfunction of $C_a(x, x')$, $x, x' \in [0, 1]^2$. 40

4.5 The third eigenfunction of $C_a(x, x')$, $x, x' \in [0, 1]^2$. 41

4.6 The fourth eigenfunction of $C_a(x, x')$, $x, x' \in [0, 1]^2$. 41

4.7 The maxima and minima of the diffusion coefficients $a_{M,\mathbf{i}}(x)$ for all 9216 systems. . . 45

4.8 The maxima singular values of all 9216 systems without any preconditioning. 46

4.9 The minima singular values of all 9216 systems without any preconditioning. 47

4.10 The condition numbers of all 9216 systems without any preconditioning. 48

4.11 The maxima singular values of all 9216 systems with the two-level hybrid preconditioner. 50

4.12 The minima singular values of all 9216 systems with the two-level hybrid preconditioner. 51

4.13 The condition numbers of all 9216 systems with the two-level hybrid preconditioner. 52

5.1 One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 2 58
5.2 One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 3 59
5.3 One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 4 60
5.4 One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 4 in the bad group 61
5.5 Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 2 62
5.6 Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 3 63
5.7 Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 4 64
5.8 Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 4 in the bad group 65
5.9 Speedup of two-level hybrid preconditioner 69
5.10 Average numbers of iterations of two-level hybrid preconditioner 69

Chapter 1

Introduction

Many phenomena in science and engineering are modeled by differential equations, which may be elliptic, parabolic, hyperbolic, linear, nonlinear, deterministic, or stochastic, etc. In order to make the modeling more accurate, more and more researchers realize that a proper stochastic process needs to be involved into the equations in domain, boundary, initial condition or coefficient. The resulting equations are called stochastic differential equations. For example, we use a heat equation to describe the heat conduction process in physics. The conductivity coefficient is a deterministic function in terms of only space and time if the medium is homogenous. However, if the medium is inhomogenous, the deterministic coefficient would not be able to describe the physical phenomenon accurately, therefore, under this condition, a stochastic process is necessary to be included in the conductivity coefficient in the heat equation for accuracy.

In this thesis, we focus on and develop parallel implicit algorithms for solving a parabolic partial differential equation (PDE)

$$\begin{cases} \dfrac{\partial u}{\partial t} - \nabla(a\nabla u) = f(x) & \text{in} \quad D \times (0,T] \\ u(x,0) = u^0(x) & \text{in} \quad D \\ u(x,t) = 0 & \text{on} \quad \partial D \times [0,T], \end{cases} \tag{1.1}$$

where $x \in D \subset R^2$, and $f(x)$ and $u^0(x)$ are given. The diffusion coefficient a is either a standard function $a = a(x)$ or a stochastic process $a = a(x,\omega)$ with the random variable ω. In the latter case, the parabolic equation is called stochastic [37, 46]. Traditional parallel approaches for solving time dependent problems focus on the parallelization within each time step, and are purely sequential

between time steps. As the number of processors becomes much larger on recent, and future, supercomputers, a new generation of algorithms is being introduced that are parallel not only in space, but also in time. This higher degree of parallelism is desirable especially for the upcoming exascale computers with expected millions or more processors.

Generally speaking, "time" is a sequential concept, the solution $u(x, t^{k+1})$ can not be computed without knowing the solution $u(x, t^k)$ at the previous time step. However, since both $u(x, t^{k+1})$ and $u(x, t^k)$ are computed iteratively, their approximate solutions do not necessarily have the sequential dependency and can be obtained simultaneously. Based on this observation, several classes of algorithms have been developed.

Waveform relaxation [31, 44] is one of the time-parallel methods to solve systems of ordinary differential equations (ODEs) with initial condition. In this method, the matrix from the discretized system is separated into lower, diagonal and upper components. The decoupling of the matrix allows independent solving of each uncoupled system in parallel. For parabolic PDEs, a semi-discretization is applied to transform PDEs into ODEs, then the resulting systems can be solved by the waveform relaxation method. In order to accelerate the convergence, several variants of waveform relaxation are developed, for example, multigrid waveform relaxation method [25, 42] or Schwarz waveform relaxation method [17, 41].

The space-time multigrid method for parabolic PDEs [23, 24, 43] considers time as an additional dimension beside the spacial dimensions. It applies the multigrid operators of smoothing, coarsening, restriction and prolongation on the whole grid combining both temporal and spacial domains.

The parareal algorithm proposed by Lions, Maday and Turinici in [32] is an iterative method to solve evolution problems in a time-parallel manner. Since this method is attracting more and more attentions in the numerical mathematical world, we briefly introduce the general idea of the

basic parareal algorithm. For the general problem

$$\begin{cases} \dfrac{\partial y}{\partial t} + Ay &= 0 \\ y(0) &= y^0 \end{cases} \tag{1.2}$$

on the interval $(0, T)$, where A is an operator over the Hilbert space. By evenly dividing the time interval into L subintervals, we obtain the decomposition as follows

$$0 = T_0 < T_1 < \cdots < T_L = T,$$

with $T_i = i\Delta T$, and $\Delta T = T/L$.

We divide problem (1.2) as L subproblems with the similar form

$$\begin{cases} \dfrac{\partial y_l}{\partial t} + Ay_l &= 0, \quad t \in (T_l, T_{l+1}) \\ y(T_l^+) &= \lambda_l \end{cases} \tag{1.3}$$

for $l = 0, \cdots, L-1$. Then the collection of solutions $\{y_0, y_1, \cdots, y_{L-1}\}$ of problem (1.3) is connected with the solution y of the original problem (1.2) if only if

$$\lambda_l = y(T_l). \tag{1.4}$$

or, more intrinsically,

$$\lambda_l = y_{l-1}(T_l^-) \quad \text{with} \quad y_{-1}(T_0) = y^0.$$

Assuming that A is time-independent, we introduce the propagator $\mathcal{F}_{\Delta T}$ such that, for any given μ, $\mathcal{F}_{\Delta T}(\mu)$ is the solution at time ΔT of equation (1.2) with $y^0 = \mu$, or in a matrix form

$$\begin{pmatrix} I & & & & & \\ -\mathcal{F}_{\Delta T} & I & & & & \\ & \ddots & \ddots & & & \\ & & -\mathcal{F}_{\Delta T} & I & & \\ & & & \ddots & \ddots & \\ & & & & -\mathcal{F}_{\Delta T} & I \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_l \\ \vdots \\ \lambda_{L-1} \end{pmatrix} = \begin{pmatrix} y^0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{1.5}$$

or in the matrix notation

$$M\Lambda = F.$$

In order to accelerate the convergence, an iterative procedure is introduced to construct a sequence Λ^k that converges to the exact solution of equation (1.5). Using a coarse propagator $\mathcal{G}_{\Delta T}$ and an implicit Euler scheme

$$\frac{\mathcal{G}_{\Delta T}(\mu)}{\Delta t} + A\mathcal{G}_{\Delta T}(\mu) = 0,$$

then the corrector is defined as follows

$$\lambda_l^k = \mathcal{F}_{\Delta T}(\lambda_{l-1}^{k-1}) + \mathcal{G}_{\Delta T}(\lambda_{l-1}^{k}) - \mathcal{G}_{\Delta T}(\lambda_{l-1}^{k-1})$$

where the superscript k is the iteration index. Notice that $\mathcal{F}_{\Delta T}$ is a parallel propagator, and $\mathcal{G}_{\Delta T}$ is strictly serial. We introduce the matrix

$$\tilde{M} = \begin{pmatrix} I & & & & & \\ -\mathcal{G}_{\Delta T} & I & & & & \\ & \ddots & \ddots & & & \\ & & -\mathcal{G}_{\Delta T} & I & & \\ & & & \ddots & \ddots & \\ & & & & -\mathcal{G}_{\Delta T} & I \end{pmatrix}, \tag{1.6}$$

then the iterative procedure can be written in the matrix form as follows

$$\Lambda^{k+1} = \Lambda^k + \tilde{M}^{-1} Res^k,$$

where $Res^k = F - M\Lambda^k$. Refer to [34] for more details about the derivation.

More applications of parareal algorithm are reported in [4, 34, 35] and references therein. The advantage of the algorithm is that it successfully approximates the solutions later in time before accurately finding the solutions at earlier times. The algorithm has received great attention since it was proposed, and several variants are presented in different frameworks, for example, PITA (parallel implicit time integrator) [18], space-time multigrid and multiple shooting method [19].

Inspired by these time-parallel approaches, we propose an implicit space-time domain decomposition method for a parabolic PDE. To find the solution u at time steps $0 = t^0 < t^1 < \cdots < t^n = T$, $\Delta t_i = t^i - t^{i-1}$,we group equations for s ($s \leq n$) steps into a single system,

$$\begin{cases} u_h^1 + \Delta t_1 L_h(u_h^1) - u_h^0 &= \Delta t_1 f^1 \\ u_h^2 + \Delta t_2 L_h(u_h^2) - u_h^1 &= \Delta t_2 f^2 \\ & \vdots \\ u_h^s + \Delta t_s L_h(u_h^s) - u_h^{s-1} &= \Delta t_s f^s. \end{cases} \tag{1.7}$$

Here L_h is a discrete version of the elliptic part of equation (1.1), and u_h^k is a space-time approximation of $u(x,t)$. The coupled system becomes more ill-conditioned as s increases (we will theoretically prove this point in the later chapter), therefore, we usually select s to be much smaller than n. The algorithm needs to be used several times in order to cover all n time steps.

One of the main emphases of our work is to develop an overlapping Schwarz preconditioned recycling Krylov subspace method for solving the system (1.7). The subproblems are obtained by a partition of $D \times [t^1, t^s]$. The classical Schwarz theory [7, 8, 9] doesn't work for this problem, but our numerical experiments show that both the one- and two-level Schwarz algorithms which we will introduce later work very well.

The second focus of our work is to consider the case when the diffusion coefficient is a stochastic process that satisfies a certain probability distribution. The widely used methods for solving stochastic differential equations are Monte Carlo method and its variants [15, 45]. For Monte Carlo method, a sequence of realizations of random inputs are generated based on a given probability distribution, in which all the realizations are deterministic and independent of each other. The statistical information, such as mean and variance, of the problem is obtained from the solutions of the sequence of deterministic realizations. The main advantage of the Monte Carlo method is easy to implement, which simply requires repeating the deterministic simulations over and over again, but the converge rate of the solution statistics is relatively slow. For instance, the mean value of the solution typically converges as $1/\sqrt{K}$, where K is the number of realizations [28, 45]. Although some modified techniques are developed to accelerate the convergence, the expense on the

repetitive realizations is still a huge computational burden to limit their applicability. Alternatively, a recently proposed numerical approach has attracted great interests among the scientific computing community, which is called the stochastic Galerkin method. Using a stochastic Galerkin approach [20, 29, 30, 46], we convert the stochastic parabolic problem into a large number of deterministic equations that are similar to (1.7). After discretization, these linear systems have different matrices and right-hand sides. When designing methods for solving these systems, a key design point is "reuse of computation", which is a trivial issue for direct type methods, but a rather difficult task for methods that are "iterative" since most components are re-computed from iteration to iteration. Our approach starts with a careful analysis of the sequence of systems, orders them appropriately, and then puts them into separate groups. Within each group we construct a single Krylov subspace and a preconditioner that are effective for solving all systems in this group. To demonstrate the applicability of the method and its parallel performance, we implement the method on top of PETSc [5] and obtain some excellent results for a sequence with more than 9000 systems in which each system has several millions or even tens of millions degrees of freedom.

Chapter 2

Space-time domain decomposition methods

In this chapter, we consider the numerical solution of a deterministic parabolic equation

$$\begin{cases} \dfrac{\partial u}{\partial t} + Lu &= f(x) \quad \text{in} \quad D \times (0,T] \\ u(x,0) &= u^0(x) \quad \text{in} \quad D \\ u(x,t) &= 0 \qquad \text{on} \quad \partial D \times [0,T], \end{cases} \tag{2.1}$$

where $D \in \mathbb{R}^2$ is a polygonal domain with boundary ∂D, and L is an elliptic operator of the form

$$Lu = -\nabla \cdot (a(x) \nabla u).$$

Let $0 = t^0 < t^1 < \cdots < t^n = T$ and $\Delta t_k = t^k - t^{k-1}$. Suppose $u^k(x)$ is the solution at time t^k. We use a backward Euler scheme for the time discretization, then the problem becomes

$$\frac{u^{k+1}(x) - u^k(x)}{\Delta t_{k+1}} + Lu^{k+1} = f^{k+1}, \quad \text{for } k = 0, 1, \cdots, n-1. \tag{2.2}$$

Let L_h be the discretized operator in the spacial domain, and u_h^k the nodal solution at $t = t^k$, then we obtain the finite difference equation

$$u_h^{k+1} + \Delta t_{k+1} L_h \left(u_h^{k+1} \right) - u_h^k = \Delta t_{k+1} f^{k+1}. \tag{2.3}$$

We denote $\bar{L}_h$ as

$$\bar{L}_h \left(u_h^k \right) = u_h^k + \Delta t_k L_h \left(u_h^k \right) = \left(I + \Delta t_k L_h \right) \left(u_h^k \right),$$

and

$$U = \left(u_h^1, u_h^2, \cdots, u_h^s \right)^T$$

where $s(\leq n)$ is the window size; i.e., the number of time steps coupled into one system. Now the equation (2.3) can be rewritten in the matrix form

$$A_s U = B, \quad i.e.$$

$$\begin{pmatrix} \bar{L}_h & & & & & \\ -I & \bar{L}_h & & & & \\ & \ddots & \ddots & & & \\ & & -I & \bar{L}_h & & \\ & & & \ddots & \ddots & \\ & & & & -I & \bar{L}_h \end{pmatrix} \begin{pmatrix} u_h^1 \\ u_h^2 \\ \vdots \\ u_h^k \\ \vdots \\ u_h^s \end{pmatrix} = \begin{pmatrix} u_h^0 + \Delta t_1 f^1 \\ \Delta t_2 f^2 \\ \vdots \\ \Delta t_k f^k \\ \vdots \\ \Delta t_s f^s \end{pmatrix}. \tag{2.4}$$

By solving several linear systems of the form (2.4), all the approximations of $u(x,t)$ at $t^1, t^2, \cdots, t^n$ are obtained.

The matrix A_s has the same block matrix on the diagonal and $-I$ on the subdiagonal. Based on this special structure, we have the following theorem that relates the condition number of A_s and the window size s.

Theorem 1 The condition number of A_s is nondecreasing as s increases.

Theorem 1 may be proved by the Rayleigh-Ritz method similar to the considerations of [1], where a Schur complement argument is employed. One can also obtain the result directly by using the largest singular vector of A_s to approximate that of A_{s+1}. The proof of the latter method is given based on the following two lemmas.

Lemma 1 The largest singular value of A_{s+1} is no less than that of A_s.

Proof. Let

$$A_s = \begin{pmatrix} \bar{L}_h & & & & & & \\ -I & \bar{L}_h & & & & & \\ & \ddots & & & & & \\ & -I & \bar{L}_h & & & & \\ & & & \ddots & & & \\ & & & -I & \bar{L}_h & & \\ & & & & -I & \bar{L}_h \end{pmatrix}.$$

The largest singular value of a matrix A, is defined by

$$\sigma_1(A) := \sqrt{\max_{\|x\|=1} x^* A^* A x}.$$

Suppose the size of $\bar{L}_h$ is m, then the size of matrix A_{s+1} is $(s+1)m$.

$$A_{s+1} = \begin{pmatrix} A_s & 0 \\ -\bar{I} & \bar{L}_h \end{pmatrix}_{(s+1)m},$$

where $0 = 0_{sm\times 1}$ is a vertical zero vector of size $sm \times 1$, and $\bar{I} = \bar{I}_{1\times sm} = (0\ I) = \left(0_{1\times(s-1)m}\ I\right)$ is a horizontal vector of size $1 \times sm$.

We consider $x^* A_{s+1}^* A_{s+1} x$,

$$\begin{aligned} A_{s+1}^* A_{s+1} &= \begin{pmatrix} A_s^* & -\bar{I}^* \\ 0 & \bar{L}_h^* \end{pmatrix} \begin{pmatrix} A_s & 0 \\ -\bar{I} & \bar{L}_h \end{pmatrix} \\ &= \begin{pmatrix} A_s^* A_s + I_1 & -\bar{L}_{h1} \\ -\bar{L}_{h1}^* & \bar{L}_h^* \bar{L}_h \end{pmatrix}, \end{aligned}$$

where

$$I_1 = \begin{pmatrix} 0 & 0 \\ 0 & I_m \end{pmatrix}_{sm\times sm}, \quad \bar{L}_{h1} = \begin{pmatrix} 0 \\ \bar{L}_h \end{pmatrix}_{sm\times m}.$$

$$
\begin{aligned}
x^* A_{s+1}^* A_{s+1} x &= \begin{pmatrix} x_1^* & x_2^* \end{pmatrix} \begin{pmatrix} A_s^* & -\bar{I}^* \\ 0 & \bar{L}_h^* \end{pmatrix} \begin{pmatrix} A_s & 0 \\ -\bar{I} & \bar{L}_h \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
&= \begin{pmatrix} x_1^* & x_2^* \end{pmatrix} \begin{pmatrix} A_s^* A_s + I_1 & -\bar{L}_{h1} \\ -\bar{L}_{h1}^* & \bar{L}_h^* \bar{L}_h \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
&= x_1^* \left(A_s^* A_s + I_1 \right) x_1 - x_1^* \bar{L}_{h1} x_2 - x_2^* \bar{L}_{h1}^* x_1 + x_2^* \bar{L}_h^* \bar{L}_h x_2 \\
&= x_1^* A_s^* A_s x_1 + x_1^* I_1 x_1 - x_1^* \bar{L}_{h1} x_2 - x_2^* \bar{L}_{h1}^* x_1 + x_2^* \bar{L}_h^* \bar{L}_h x_2,
\end{aligned}
$$

where x_1, x_2 are compatible to the block matrix multiplication. Note that $x_1^* I_1 x_1$ is the sum of the squares of the last m elements in x_1.

We set

$$
x = \begin{pmatrix} x_1 \\ 0 \end{pmatrix}
$$

such that $\|x\| = \|x_1\| = 1$ and assume that x_1 is the vector to satisfy

$$
\sigma_1^2(A_s) = \max_{\|x\|=1} x^* A_s^* A_s x = x_1^* A_s^* A_s x_1.
$$

In this case,

$$
x^* A_{s+1}^* A_{s+1} x = x_1^* A_s^* A_s x_1 + x_1^* I_1 x_1.
$$

Hence,

$$
\sigma_1^2(A_{s+1}) \geq x^* A_{s+1}^* A_{s+1} x = x_1^* A_s^* A_s x_1 + x_1^* I_1 x_1 \geq \sigma_1^2(A_s),
$$

which yields:

$$
\sigma_1(A_{s+1}) \geq \sigma_1(A_s).
$$

Lemma 2 The smallest singular value of A_{s+1} is no greater than that of A_s.

Proof. The smallest singular value of a matrix A is defined by

$$\sigma_n(A) := \sqrt{\min_{\|x\|=1} x^* A^* A x}.$$

Note that A_{s+1} can also be written as

$$A_{s+1} = \begin{pmatrix} \bar{L}_h & 0 \\ -\tilde{I} & A_s \end{pmatrix},$$

where

$$\tilde{I} = \begin{pmatrix} I_m \\ 0 \end{pmatrix}_{sm \times m}.$$

Then

$$\begin{aligned} A_{s+1}^* A_{s+1} &= \begin{pmatrix} \bar{L}_h^* & -\tilde{I}^* \\ 0 & A_s^* \end{pmatrix} \begin{pmatrix} \bar{L}_h & 0 \\ -\tilde{I} & A_s \end{pmatrix} \\ &= \begin{pmatrix} \bar{L}_h^* \bar{L}_h + I_2 & -A_s(m) \\ -A_s(m)^* & A_s^* A_s \end{pmatrix}, \end{aligned}$$

where

$$I_2 = \begin{pmatrix} I_m & 0 \\ 0 & 0 \end{pmatrix}_{sm \times sm},$$

and $A_s(m)$ represents the matrix consisting of the first m rows of A_s.

$$\begin{aligned} x^* A_{s+1}^* A_{s+1} x &= \begin{pmatrix} x_1^* & x_2^* \end{pmatrix} \begin{pmatrix} \bar{L}_h^* \bar{L}_h + I_2 & -A_s(m) \\ -A_s(m)^* & A_s^* A_s \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ &= x_1^* \left(\bar{L}_h^* \bar{L}_h + I_2 \right) x_1 - x_1^* A_s(m) x_2 - x_2^* A_s(m) x_1 + x_2^* A_s^* A_s x_2. \end{aligned}$$

We set

$$x = \begin{pmatrix} 0 \\ x_2 \end{pmatrix}$$

such that $\|x\| = \|x_2\| = 1$ and assume that x_1 is the vector to satisfy

$$\sigma_n^2(A_s) = \max_{\|x\|=1} x^* A_s^* A_s x = x_2^* A_s^* A_s x_2.$$

In this case,

$$x^* A_{s+1}^* A_{s+1} x \quad = \quad x_1^* A_s^* A_s x_1.$$

Hence,

$$\sigma_n^2(A_{s+1}) \leq x^* A_{s+1}^* A_{s+1} x = x_1^* A_s^* A_s x_1 \leq \sigma_n^2(A_s).$$

Taking the square root on both sides, we obtain the desired result:

$$\sigma_n(A_{s+1}) \leq \sigma_n(A_s).$$

The condition number of A is defined by

$$\kappa(A) = \frac{\sigma_1(A)}{\sigma_n(A)},$$

then we obtain

$$\kappa(A_s) \leq \kappa(A_{s+1}),$$

which complete our proof of Theorem 1.

As an example for Theorem 1, we set $A = A_1 = \bar{L}_h$, the matrix takes the form

$$A_1 = \begin{pmatrix} 5.1 & 2 & 2 & & \\ 2 & 5.1 & 2 & 2 & \\ 2 & 2 & 5.1 & 2 & 2 \\ & 2 & 2 & 5.1 & 2 \\ & & 2 & 2 & 5.1 \end{pmatrix}.$$

From Table 2.1 of the experiment, we see that the maximum singular value of A_s increases with s, and they strictly increase in this example. The minimum singular value of A_s decreases with

Table 2.1: The extrema singular values and condition numbers for matrices A_s as s increases.

	σ_1	σ_n	$\kappa(A)_s$		σ_1	σ_n	$\kappa(A)_s$
A_1	3.3122	1.3653	2.4261	A_{41}	11.9683	0.8696	13.7625
A_2	11.4823	1.4298	8.0305	A_{42}	11.9684	0.8694	13.7667
A_3	11.6891	1.2330	9.4805	A_{43}	11.9685	0.8691	13.7706
A_4	11.7890	1.1252	10.4771	A_{44}	11.9686	0.8689	13.7742
A_5	11.8441	1.0592	11.1826	A_{45}	11.9687	0.8687	13.7776
A_6	11.8776	1.0155	11.6961	A_{46}	11.9688	0.8685	13.7809
A_7	11.8995	0.9851	12.0795	A_{47}	11.9689	0.8683	13.7839
A_8	11.9144	0.9630	12.3720	A_{48}	11.9690	0.8682	13.7867
A_9	11.9252	0.9465	12.5996	A_{49}	11.9691	0.8680	13.7894
A_{10}	11.9331	0.9337	12.7798	A_{50}	11.9691	0.8678	13.7919

s, also they strictly decrease in the example. Therefore, the condition number of A_s increases as the number of subblocks on the diagonal increases. Also note that the condition number increases very slowly in s.

As we know, a matrix with a low condition number is said to be well-conditioned, while a problem with a high condition number is said to be ill-conditioned. To solve this problem, we consider the preconditioning technique to reduce the condition number of a larger linear system.

According to Theorem 1, we realize that the window size cannot be too large, especially for certain sensitive systems that can be easily perturbed to be nearly singular. Because of this, additional technique should be considered to reduce the condition number, for example, Schwarz preconditioning method which we now introduce.

Before formally defining the Schwarz preconditioners in the space-time formulation, we firstly recall the traditional Schwarz preconditioners. For the one-level Schwarz preconditioner, we begin with a fine mesh $\mathcal{D}_h$ on the spacial domain D. We first decompose D into non-overlapping subregions $D_k, k = 1, 2, \cdots, N$, where the number of subdomain N is always the same as the number of processors np. To obtain an overlapping decomposition of the domain, we extend each subregion D_k to D_k^δ by including extra δ mesh layers from adjacent subregions. Assume we have m fine mesh points inside the whole spacial domain D and m_k fine mesh points inside the overlapped subdomain D_k^δ, then the elements of matrix R_k^δ of size $n_k \times n$ are defined as follow:

$$(R_k^\delta)_{pq} = \begin{cases} 1 & \text{if mesh point associated with } p, q \in D_k^\delta, \\ 0 & \text{otherwise.} \end{cases}$$

The matrix R_k^δ is used as a restriction operator by getting rid of all the components that are outside of the subregion D_k^δ. If we transpose the restriction matrix, the resulting matrix R_k^T works as an extension operator, which refills the components that do not belong to D_k^δ by 0. The one-level restricted additive Schwarz (RAS) preconditioner [10] for the matrix A can be written as

$$M_1^{-1} = (R_1^0)^T A_1^{-1} R_1^\delta + (R_2^0)^T A_2^{-1} R_2^\delta + \cdots + (R_N^0)^T A_N^{-1} R_N^\delta.$$

where $A_k, k = 1, 2, \cdots, N$, are subdomain matrices defined by $A_k = R_k^\delta A (R_k^\delta)^T$. The matrix vector

multiplication involving A_k^{-1} is usually computed by LU or ILU factorization.

For the traditional two-level additive Schwarz preconditioning, besides the one-level additive Schwarz preconditioner as a component, we also include a coarse level by defining a restriction operator I_h^H from the spacial fine mesh to the spacial coarse mesh, an interpolation operator I_H^h from the spacial coarse mesh to the spacial fine mesh. By constructing a coarse matrix $A_0 = I_h^H A_s I_H^h$ on the spacial coarse mesh, we obtain the two-level additive Schwarz preconditioner as

$$M_2^{-1} = I_H^h A_0^{-1} I_h^H + M_1^{-1}.$$

For the space-time formulation, we define a fine mesh $\mathcal{D}_h^s$ on the domain $D \times [0, T]$ by simply combining all the fine meshes in the domain $D \times [t^{l+1},\ t^{l+s}]$. Hence, the nonoverlapping and overlapping decomposition of $D \times [t^{l+1},\ t^{l+s}]$ are defined as $D_k \times [t^{l+1},\ t^{l+s}]$ and $D_k^\delta \times [t^{l+1},\ t^{l+s}]$, respectively. Suppose we have $\bar{n}$ fine mesh points inside $D \times [t^{l+1},\ t^{l+s}]$ and $\bar{n}_k$ fine mesh points inside $D_k^\delta \times [t^{l+1},\ t^{l+s}]$, then the elements of a matrix $\bar{R}_k^\delta$ of size $\bar{n}_k \times \bar{n}$ are defined as follows:

$$(\bar{R}_k^\delta)_{pq} = \begin{cases} 1 & \text{if mesh point associated with } p, q \in D_k^\delta \times [t^{l+1},\ t^{l+s}], \\ 0 & \text{otherwise.} \end{cases}$$

Analogously, the matrix $\bar{R}_k^\delta$ serves as a restriction operator by eliminating all the components that are outside of $D_k^\delta \times [t^{l+1}\ t^{l+s}]$. And the transpose of the matrix $\bar{R}_k^\delta$ refills the components that do not belong to $D_k^\delta \times [t^{l+1}\ t^{l+s}]$ by 0. Therefore, the one-level RAS preconditioner in the space-time formulation for the matrix A_s can be written as

$$\bar{M}_1^{-1} = (\bar{R}_1^0)^T A_1^{-1} \bar{R}_1^\delta + (\bar{R}_2^0)^T A_2^{-1} \bar{R}_2^\delta + \cdots + (\bar{R}_N^0)^T A_N^{-1} \bar{R}_N^\delta,$$

where $A_k, k = 1, 2, \cdots, N$, are subdomain matrices defined by $A_k = \bar{R}_k^\delta A_s (\bar{R}_k^\delta)^T$. The solution corresponding to the matrix A_k^{-1} is also computed by LU or ILU factorization based on the performance of the convergence.

Figure 2.1 shows the partition of a $8 \times 8 \times 4$ mesh, and the corresponding matrix pattern without overlapping. From the bottom figure of Figure 2.1, we notice that the discretized matrix is not symmetric because of the extra subdiagonal components.

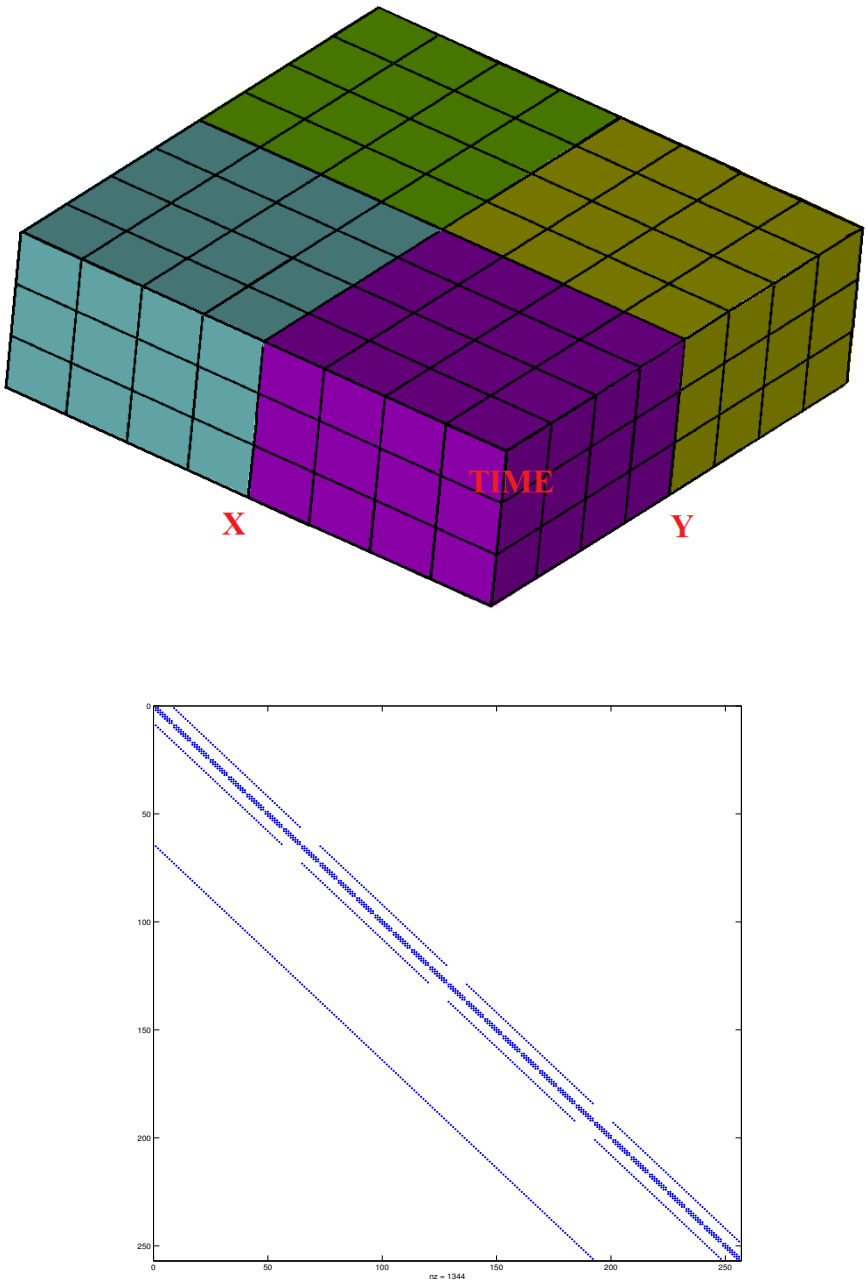

Figure 2.1: The pattern of matrix A_s for a $8 \times 8 \times 4$ mesh, $s = 4$ is the number of time levels. The number of processors is 4.

For the two-level additive Schwarz preconditioning in the space-time formulation, we define a restriction operator $\bar{I}_h^H$ from the fine mesh to the coarse mesh on $D_k^\delta \times [t^{l+1}\ t^{l+s}]$, and also define an interpolation operator $\bar{I}_H^h$ from the coarse mesh to the fine mesh on $D_k^\delta \times [t^{l+1}\ t^{l+s}]$. Accordingly, the coarse matrix $\bar{A}_0$ is constructed by

$$\bar{A}_0 = \bar{I}_H^h A_s \bar{I}_H^h,$$

and the two-level additive Schwarz preconditioner in the space-time formulation is

$$\bar{M}_2^{-1} = \bar{I}_H^h \bar{A}_0^{-1} \bar{I}_h^H + \bar{M}_1^{-1}.$$

Another widely used multilevel Schwarz method is the two-level hybrid preconditioner, which combines the coarse-level and fine-level in a multiplicative manner. More preciously, the space-time formulation of the two-level hybrid preconditioner is given as

$$\begin{aligned} z &:= \bar{M}_1^{-1} x, \\ z &:= z + \bar{M}_C^{-1}(x - A_s z), \\ z &:= z + \bar{M}_1^{-1}(x - A_s z), \end{aligned}$$

where $\bar{M}_C^{-1} = \bar{I}_H^h \bar{A}_0^{-1} \bar{I}_h^H$ is the coarse preconditioner constructed as the first term of $\bar{M}_2^{-1}$. On each subdomain, zero Dirichlet boundary conditions are used on the internal subdomain boundary $\partial(D_k^\delta \times [t^{l+1},\ t^{l+s}]) \cap D \times [t^{l+1},\ t^{l+s}]$, and the original boundary conditions are used on the physical boundary. Several inexact additive Schwarz preconditioners are available. In our numerical experiments, we employ the two-level hybrid preconditioner with an incomplete factorization for the subdomain matrices in the implementation.

Chapter 3

Stochastic Galerkin method for parabolic problems

In this chapter, we briefly introduce the main components of stochastic Galerkin method to show how to discretize a parabolic equation with a stochastic diffusion coefficient. Some related details are available in [2, 3, 16, 20].

3.1 Weak form of the stochastic parabolic equation

At the beginning, we introduce some notations and conceptions [6]. Let Ω be a given sample space, then a $\sigma-$algebra $\mathcal{A}$ on Ω is a family $\mathcal{A}$ of subsets of Ω with the following properties:

(1) $\phi \in \mathcal{A}$

(2) $A \in \mathcal{A} \Rightarrow A^C \in \mathcal{A}$, where $A^C = \Omega \backslash A$

(3) $A_1, A_2, \cdots \in \mathcal{A} \Rightarrow A := \bigcup A_i \in \mathcal{A}$.

The pair $(\Omega, \mathcal{A})$ is called a measurable space. A probability measure P on a measurable space $(\Omega, \mathcal{A})$ is a function $P : \mathcal{A} \longrightarrow [0, 1]$ such that

(1) $P(\phi) = 0$, $P(\Omega) = 1$

(2) If $A_1, A_2, \cdots \in \mathcal{A}$ and $\{A_i\}_{i=1}^{\infty}$ is disjoint, then

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i).$$

The triple $(\Omega, \mathcal{A}, P)$ is called a probability space. It is called a complete probability space if and only if

$$A \in \mathcal{A} \text{ with } P(A) = 0, \text{ and } B \subset A \Longrightarrow B \in \mathcal{A}.$$

We define a real-valued random function $\mathcal{F}$ on the jointly measurable spacial domain D and sample space Ω

$$\mathcal{F} : D \times \Omega \longrightarrow \mathbb{R}$$

such that for each fixed point $x \in D$, $\mathcal{F}(x, \cdot)$ is a random variable with respect to the probability space $(\Omega, \mathcal{A}, P)$, and for each sample $\omega \in \Omega$, $\mathcal{F}(\cdot, \omega)$ is a function on D. Thus, a random function is a stochastic process with the spatial coordinate $x \in D$ and sample point $\omega \in \Omega$.

Furthermore, the mean or expected value of a random varialbe ξ,

$$\xi : \Omega \longrightarrow \mathbb{R}$$

is denoted by

$$\langle \xi(\omega) \rangle := \int_{\Omega} \xi(\omega) dP(\omega) = \int_{\mathbb{R}} t\rho(t) dt$$

where ρ is the probability density function of $\xi(\omega)$.

The covariance of the random function f at $x_1, x_2 \in D$ is denoted by

$$Cov_f := \langle (f(x_1, \cdot) - \langle f(x_1, \cdot) \rangle)(f(x_2, \cdot) - \langle f(x_2, \cdot) \rangle) \rangle.$$

By definition, the covariance functions are bounded, symmetric, positive definite.

A random function $a(x, \omega)$ is jointly measurable defined from $D \times \Omega$ to $\mathbb{R}$. We define the stochastic Hilbert space

$$L^2(D \times \Omega) = \{u(x, \omega) | \quad \langle \| \, u(x, \omega) \, \|_{L^2(D)} \rangle < \infty\}.$$

Using the standard notation, the function space $H_0^1(D)$ is the subspace of the Sobolev space $H^1(D)$ consisting of functions which vanish on ∂D. Similarly, the stochastic Sobolev space $H_0^1(D \times \Omega) \times H([0, T])$ is defined by

$$H_0^1(D \times \Omega) \times H([0, T]) = \{u(x, t, \omega) | \quad \langle \| \, u(x, t, \omega) \, \|_{H_0^1(D \times \Omega) \times H([0,T])} \rangle < \infty\}.$$

The inner product over D is defined using the Lebesgue measure; i.e., for any two functions $h_1(x), h_2(x)$, their inner product is defined by

$$(h_1(x), h_2(x)) = \int_D h_1(x)h_2(x)dx.$$

All the notations and terminologies are listed above, now let's recall the classical deterministic parabolic PDE:

$$\begin{cases} \dfrac{\partial u(x,t)}{\partial t} - \nabla \cdot (a(x)\nabla u(x,t)) &= f(x) \quad \text{on} \quad D \times (0,T] \\ u(x,0) &= u^0(x) \quad \text{on} \quad D \\ u(x,t) &= 0 \qquad \text{on} \quad \partial D \times [0,T]. \end{cases}$$

Its weak form is to find $u(x,t) \in H_0^1(D) \times H^1([0,T])$ such that

$$\int_D u_t v(x)dx + \int_D a(x)\nabla u(x) \cdot \nabla v(x)dx = \int_D f(x)v(x)dx \tag{3.1}$$

for any $v(x) \in H_0^1(D)$.

Now we change the coefficient function $a(x)$ into the random function $a(x,\omega) \in L^2(D \times \Omega)$, so the stochastic parabolic PDE is changed correspondingly into:

$$\begin{cases} \dfrac{\partial u(x,t,\omega)}{\partial t} - \nabla \cdot (a(x,\omega)\nabla u(x,t,\omega)) &= f(x) \quad \text{on} \quad D \times (0,T] \times \Omega \\ u(x,0,\omega) &= u^0(x) \quad \text{on} \quad D \\ u(x,t,\omega) &= 0 \qquad \text{on} \quad \partial D \times [0,T]. \end{cases} \tag{3.2}$$

Analogously, the weak form of the stochastic parabolic PDE is to find $u(x,t,\omega) \in H_0^1(D \times \Omega) \times H^1([0,T])$ such that

$$\left\langle \int_D \frac{\partial u}{\partial t} v dx \right\rangle + \left\langle \int_D a(x,\omega)\nabla u(x,\omega) \cdot \nabla v dx \right\rangle = \left\langle \int_D f(x)v dx \right\rangle \tag{3.3}$$

for any $v \in H_0^1(D \times \Omega)$. Here $\langle \cdot \rangle$ denotes the expected value.

3.2 Karhunen-Loève (KL) expansion

The KL expansion [33] is one of the most widely used methods to represent the stochastic coefficient and for dimensional reduction. In this thesis, we employ the KL expansion by assuming

the mean function $a_0(x)$ and covariance function $C_a(x_1, x_2)$ of $a(x, \omega)$ are given as

$$\langle a(x,\omega)\rangle = a_0(x) = \int_\Omega a(x,\omega)dP(\omega),$$

and

$$C_a(x_1,x_2) = \int_\Omega (a(x_1,\omega) - a_0(x_1))(a(x_2,\omega) - a_0(x_2))dP(\omega).$$

On a probability space $(\Omega, \mathcal{A}, P)$, we denote $L^2(\Omega, \mathcal{A}, P)$ the collection of real-valued random variables $a(\omega)$ defined on $(\Omega, \mathcal{A}, P)$ such that the expectation of the square of the random variable is finite, i.e.

$$\langle a(\omega)^2\rangle < \infty.$$

The expectation $\langle xy\rangle$ defines an inner product $(\cdot,\cdot)$ on $L^2(\Omega, \mathcal{A}, P)$ associated with the L^2 norm

$$\|a(\omega)\|_{L^2} = \langle a(\omega)^2\rangle^{1/2}.$$

It has been shown that $L^2(\Omega, \mathcal{A}, P)$, equipped with the inner product $(\cdot,\cdot)$ and the L^2-norm is a Hilbert space. Thus, by the definition of the completeness, any random variable $a(\omega) \in L^2$ can be expressed as a summation:

$$a(\omega) = \sum_{i=0}^{\infty} c_i\zeta_i(\omega),$$

where $\{\zeta_i(\omega)\}_{i=0}^{\infty}$ is a basis of L^2 and $\{c_i\}_{i=0}^{\infty}$ denote the projections of $\xi(\omega)$ onto the basis $\{\zeta_i(\omega)\}_{i=0}^{\infty}$. If the stochastic process includes extra dimensions in space, the above expansion can be generated as follows:

$$\xi(x,\omega) = \sum_{i=0}^{\infty} c_i(x)\zeta_i(\omega).$$

Since the covariance function $C_a(x_1, x_2)$ of the stochastic process is bounded, symmetric, and positive definite, we can get the spectral decomposition of $C_a(x_1, x_2)$ [12]

$$C_a(x_1,x_2) = \sum_{n=1}^{\infty} \lambda_n k_n(x_1)k_n(x_2),$$

where the constants λ_n and function $k_n(x)$ are a eigenpair of the covariance function $C_a(x_1, x_2)$ such that

$$\int_D C_a(x_1,x_2)k_n(x_1)dx_1 = \lambda_n k_n(x_2).$$

For the same reason, we note that these eigenfunctions $\{k_n(x)\}_{n=1}^{\infty}$ form a complete orthonormal set in $L^2(D)$ in the sense

$$\int_D k_m(x)k_n(x)dx = \delta_{mn}$$

where δ_{mn} is the Kronecker delta. The eigenvalues $\{\lambda_n\}_{n=0}^{\infty}$ are all positive, which can be ordered non-increasingly, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq \cdots > 0$.

According to the KL expansion [20], $a(x,\omega)$ can be expressed by a series expansion:

$$a(x,\omega) = a_0(x) + \sum_{i=1}^{\infty} \sqrt{\lambda_i} k_i(x)\xi_i(\omega), \tag{3.4}$$

where $\{\xi_i(\omega)\}_{i=1}^{\infty}$, $\xi_i : \Omega \longrightarrow R$, are mutually uncorelated random variables in $L^2(\Omega)$ to be determined in the stochastic Galerkin method.

There are several important properties for the random variables $\xi_i(\omega)$ that we will derive below. Let the stochastic part of $a(x,\omega)$ be denoted by $a_\omega(x,\omega)$

$$a_\omega(x,\omega) = \sum_{i=1}^{\infty} \sqrt{\lambda_i} k_i(x)\xi_i(\omega), \tag{3.5}$$

then $a(x,\omega) = a_0(x) + a_\omega(x,\omega)$. It's easy to show that $\langle a_\omega(x,\omega)\rangle = 0$, and the covariance of $a_\omega(x,\omega)$ is the same as the covariance of $a(x,\omega)$,

$$\langle a_\omega(x,\omega)\rangle = \langle a(x,\omega) - a_0(x)\rangle = \langle a(x,\omega)\rangle - \langle a_0(x)\rangle = 0.$$

and

$$\begin{aligned} C_a(x_1,x_2) &= \int_\Omega (a(x_1,\omega) - a_0(x_1))(a(x_2,\omega) - a_0(x_2))dP(\omega) \\ &= \int_\Omega a_\omega(x_1,\omega)a_\omega(x_2,\omega))dP(\omega) \\ &= \langle a_\omega(x_1,\omega)a_\omega(x_2,\omega)\rangle. \end{aligned} \tag{3.6}$$

By equation (3.6) and (3.5), we have

$$\begin{aligned} C_a(x_1,x_2) &= \langle a_\omega(x_1,\omega)a_\omega(x_2,\omega))\rangle \\ &= \langle \sum_{i=1}^{\infty} \sqrt{\lambda_i} k_i(x_1)\xi_i(\omega) \cdot \sum_{j=1}^{\infty} \sqrt{\lambda_j} k_j(x_2)\xi_j(\omega)\rangle \\ &= \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \sqrt{\lambda_i\lambda_j} k_i(x_1)k_j(x_2)\langle \xi_i(\omega)\xi_j(\omega)\rangle. \end{aligned} \tag{3.7}$$

Using the orthogonality of the eigenfunctions, we multiply both sides of (3.7) by $k_m(x_2)$ for some fix number m and do integration with respect to x_2, we get

$$\begin{aligned}\int_D C_a(x_1,x_2)k_m(x_2)dx_2 &= \int_D \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\sqrt{\lambda_i\lambda_j}k_i(x_1)k_j(x_2)k_m(x_2)\langle\xi_i(\omega)\xi_j(\omega)\rangle dx_2 \\ &= \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\sqrt{\lambda_i\lambda_j}\langle\xi_i(\omega)\xi_j(\omega)\rangle k_i(x_1)\int_D k_j(x_2)k_m(x_2)dx_2 \\ &= \sum_{i=1}^{\infty}\sqrt{\lambda_i\lambda_m}\langle\xi_i(\omega)\xi_m(\omega)\rangle k_i(x_1). \qquad (3.8)\end{aligned}$$

We multiply $k_n(x_1)$ on the right-hand side of equation(3.8) and do integration with respect to x_1,

$$\begin{aligned}\sum_{i=1}^{\infty}\sqrt{\lambda_i\lambda_m}\langle\xi_i(\omega)\xi_m(\omega)\rangle\int_D k_i(x_1)k_n(x_1)dx_1 &= \sum_{i=1}^{\infty}\sqrt{\lambda_i\lambda_m}\langle\xi_i(\omega)\xi_m(\omega)\rangle\delta_{in} \\ &= \sqrt{\lambda_n\lambda_m}\langle\xi_n(\omega)\xi_m(\omega)\rangle. \qquad (3.9)\end{aligned}$$

By the definition of the covariance function, the left-hand side of (3.8) can be written as

$$\int_D C(x_1,x_2)k_m(x_2)dx_2 = \lambda_m k_m(x_1). \qquad (3.10)$$

Repeating the same work on the right-hand side of equation (3.10)

$$\int_D \lambda_m k_m(x_1)k_n(x_1)dx_1 = \lambda_m\delta_{mn}. \qquad (3.11)$$

Thus, the right-hand sides of equations (3.9) and (3.11) are equivalent. So

$$\sqrt{\lambda_n\lambda_m}\langle\xi_n(\omega)\xi_m(\omega)\rangle = \lambda_m\delta_{mn}. \qquad (3.12)$$

Rearrange the two sides of (3.12), we get the important property of the random functions

$$\langle\xi_n(\omega)\xi_m(\omega)\rangle = \sqrt{\frac{\lambda_m}{\lambda_n}}\delta_{mn} = \delta_{mn}. \qquad (3.13)$$

Therefore,

$$\langle\xi_n(\omega)\rangle = 0 \quad \text{and} \quad \langle\xi_n(\omega)\xi_m(\omega)\rangle = 0 \qquad \text{for } m,n\in\mathbb{Z}. \qquad (3.14)$$

[20] lists some additional properties and their proofs about the KL expansion, such as the Error Minimizing property and Uniqueness of the Expansion property.

For numerical simulation, we use a finite number of terms to approximate the series expansion of $a(x,\omega)$ by a truncated function $a_M(x,\omega)$,

$$a_M(x,\omega) = a_0(x) + \sum_{i=1}^{M} \sqrt{\lambda_i} k_i(x)\xi_i(\omega)$$

where the value of M depends on the decay of eigenvalues.

Let ρ_i be the probability density function of the random variable $\xi_i(\omega)$, then the joint probability density function of the joint random variable $\xi = (\xi_1, \xi_2, \cdots, \xi_M)$ can be denoted by $\rho = (\rho_1, \rho_2, \cdots, \rho_M)$,

$$\xi : \Omega^M \longrightarrow \mathbb{R}^M.$$

Suppose Γ_i is the image of $\xi_i \in R$, thus $\Gamma = (\Gamma_1, \Gamma_2, \cdots, \Gamma_M) \subseteq \mathbb{R}^M$ is the image of ξ, i.e.

$$\xi(\omega) = (\xi_1(\omega), \xi_2(\omega), \cdots, \xi_M(\omega)) \in \Gamma = (\Gamma_1, \Gamma_2, \cdots, \Gamma_M) \quad \text{for all } \omega \in \Omega.$$

Hence, for each $\omega \in \Omega$, there exists a unique $\xi \in \Gamma$ correspondingly. Then we can replace $a_M(x,\omega)$ by the approximation $a_M(x,\xi)$

$$a_M(x,\xi) = a_0(x) + \sum_{i=1}^{M} \sqrt{\lambda_i} k_i(x)\xi_i. \tag{3.15}$$

Now, the stochastic problem can be converted into a deterministic parabolic PDE with the solution $u_M(x,t,\xi) \in H_0^1(D) \times H^1([0,T]) \times L^2(\Gamma,\rho)$, namely,

$$\frac{\partial u_M(x,t,\xi)}{\partial t} - \nabla \cdot (a_M(x,\xi)\nabla u_M(x,t,\xi)) = f(x). \tag{3.16}$$

3.3 Double orthogonal basis

A double orthogonal basis [20] is introduced in order to decouple the equation (3.16) in the $\xi-$space under the assumption that the random variables $\{\xi_i(\omega)\}_{i=1}^M$ are all independent. The double orthogonal basis is constructed as follows. For some positive integer $n \in Z^+$, the space of single-variable polynomials of degree at most n is

$$P_n := \text{span}\{1, z, z^2, \cdots, z^n\}.$$

For $\mathbf{n}= (n_1, n_2, \cdots, n_M) \in (Z^+)^M$, we construct the multi-variable polynomial space $P_{\mathbf{n}}$ by the tensor product of M independent subspaces of single-variable polynomials P_{n_i}, for $i = 1, 2, \cdots, M$,

$$P_{\mathbf{n}} := P_{n_1} \otimes P_{n_2} \otimes \cdots \otimes P_{n_M} \in L^2(\Gamma, \rho).$$

On each independent subspace P_{n_j}, for $j = 1, 2, \cdots, M$, we define a double orthogonal polynomial basis $\{\phi_{k,j}(z)\}_{k=0}^{n_j}, j = 1, 2, \cdots, M$, by satisfying the following two conditions:

$$\begin{cases} \int_{\Gamma_j} \phi_{p,j}(z)\phi_{q,j}(z)\rho_j(z)dz &= \delta_{p,q} \qquad p, q = 0, 1, \cdots n_j, \\ \int_{\Gamma_j} t\phi_{p,j}(z)\phi_{q,j}(z)\rho_j(z)dz &= C_{p,j}\delta_{p,q} \quad p, q = 0, 1, \cdots n_j, \end{cases} \tag{3.17}$$

where $\{C_{p,j}\}_{p=0}^{n_j}$ are nonzero constants. In each subspace P_{n_j}, there are n_j+1 basis functions, which yield totally $N_{\mathbf{n}} := \prod_{j=1}^{M}(n_j + 1)$ basis functions in the approximation subspace $P_{\mathbf{n}} \in L^2(\Gamma, \rho)$. Defining an index for the $N_{\mathbf{n}}$ double orthogonal polynomials

$$\mathbf{I} = \{\{i_1, i_2, \cdots, i_M\} \mid i_j \leq n_j, \text{for } j = 1, 2, \cdots, n_M\},$$

the set of all the basis functions of $P_{\mathbf{n}}$ is

$$\left\{ \phi_{\mathbf{i}}(\xi) \mid \phi_{\mathbf{i}}(\xi) = \prod_{j=1}^{M} \phi_{i_j,j}(\xi), i_j \in \{0, 1, \cdots, n_j\}, \mathbf{i} \in \mathbf{I} \right\}. \tag{3.18}$$

3.4 Discretization in ξ-space

We use the double orthogonal basis to denote the solution

$$u_M(x, t, \xi) = \sum_{\mathbf{i}\in\mathbf{I}} u_{M,\mathbf{i}}(x, t)\phi_{\mathbf{i}}(\xi)$$

which satisfies

$$\left\langle \int_D \frac{\partial u}{\partial t} v dx \right\rangle + \left\langle \int_D a_M \nabla u \cdot \nabla v dx \right\rangle = \left\langle \int_D f(x) v dx \right\rangle, \tag{3.19}$$

for any $v = h(x)\phi_{\mathbf{j}}(\xi) \in H_0^1(D) \times P_{\mathbf{n}}(\Gamma)$, $\mathbf{j} \in \mathbf{I}$. The second term of the left-hand side in equation (3.19) can be rewritten as

$$\begin{aligned}
\left\langle \int_D a_M \nabla u \cdot \nabla v dx \right\rangle &= \left\langle \int_D a_M \nabla \left(\sum_{\mathbf{i}\in\mathbf{I}} u_{M,\mathbf{i}}(x,t)\phi_{\mathbf{i}}(\xi) \right) \cdot \nabla \left(h(x)\phi_{\mathbf{j}}(\xi) \right) \right\rangle \\
&= \sum_{\mathbf{i}\in\mathbf{I}} \left\langle \int_D a_M \nabla \left(u_{M,\mathbf{i}}(x,t)\phi_{\mathbf{i}}(\xi) \right) \cdot \nabla (h(x)\phi_{\mathbf{j}}(\xi)) \right\rangle \\
&= \sum_{\mathbf{i}\in\mathbf{I}} \left(a_0(x)\nabla u_{M,\mathbf{i}}(x,t), \nabla h(x) \right) \int_\Gamma \phi_{\mathbf{i}}(\xi)\phi_{\mathbf{j}}(\xi)\rho(\xi)d\xi \\
&\quad + \sum_{\mathbf{i}\in\mathbf{I}} \sum_{n=1}^{M} \sqrt{\lambda_n} \left(k_n(x)\nabla u_{M,\mathbf{i}}(x,t), \nabla h(x) \right) \int_\Gamma \phi_{\mathbf{i}}(\xi)\phi_{\mathbf{j}}(\xi)\xi_n\rho(\xi)d\xi.
\end{aligned}$$

Thus the representation in the double orthogonal basis yields

$$\begin{aligned}
&\left\langle \int_D a_M \nabla u \cdot \nabla v dx \right\rangle = \\
&\quad \sum_{\mathbf{i}\in\mathbf{I}} \left[\left(a_0(x)\nabla u_{M,\mathbf{i}}(x,t), \nabla h(x) \right) + \sum_{n=1}^{M} \sqrt{\lambda_n} \left(k_n(x)\nabla u_{M,\mathbf{i}}(x,t), \nabla h(x) \right) C_{i_n,j_n} \right] \delta_{\mathbf{ij}}.
\end{aligned}$$

And the first term is

$$\begin{aligned}
\left\langle \int_D \frac{\partial u(x,t,\xi)}{\partial t} v(x,\xi) dx \right\rangle &= \left\langle \int_D \left(\sum_{\mathbf{i}\in\mathbf{I}} \frac{\partial u_{M,\mathbf{i}}(x,t)}{\partial t} \phi_{\mathbf{i}}(\xi) \right) \left(h(x)\, \phi_{\mathbf{j}}(\xi) \right) dx \right\rangle \\
&= \sum_{\mathbf{i}\in\mathbf{I}} \left\langle \int_D \frac{\partial u_{M,\mathbf{i}}(x,t)}{\partial t} h(x)\, \phi_{\mathbf{i}}(\xi)\, \phi_{\mathbf{j}}(\xi)\, dx \right\rangle \\
&= \sum_{\mathbf{i}\in\mathbf{I}} \int_D \frac{\partial u_{M,\mathbf{i}}(x,t)}{\partial t} h(x)\, dx \int_\Gamma \phi_{\mathbf{i}}(\xi)\, \phi_{\mathbf{j}} \rho(\xi) d\xi \\
&= \sum_{\mathbf{i}\in\mathbf{I}} \int_D \frac{\partial u_{M,\mathbf{i}}(x,t)}{\partial t} h(x)\, dx \delta_{\mathbf{ij}}.
\end{aligned}$$

The right-hand side of equation (3.19) is

$$\begin{aligned}
\left\langle \int_D f(x)v(x,\xi)dx \right\rangle &= \int_D f(x)h(x)dx \left\langle \phi_{\mathbf{j}}(\xi) \right\rangle \\
&= \int_D f(x)h(x)dx \prod_{j=1}^{M} \int_{\Gamma_j} \phi_{i_j,j}(\xi_j)\rho_j(\xi_j)d\xi_j.
\end{aligned}$$

We summarize the results in the following theorem.

Theorem 2 For any $M \in Z^+$, there exists a multi-variable polynomial space $P_{\mathbf{n}} = P_{n_1} \otimes P_{n_2} \otimes \cdots \otimes P_{n_M} \in L^2(\Gamma, \rho)$, on which the stochastic equation (3.19), whose initial and boundary conditions

are the same as (1.1), can be decoupled into $N_{\mathbf{n}} = \prod_{j=1}^{M}(n_j+1)$ deterministic equations

$$\frac{\partial u_{M,\mathbf{i}}(x,t)}{\partial t} - \nabla \cdot \left(a_{M,\mathbf{i}}(x)\,\nabla u_{M,\mathbf{i}}(x,t)\right) = f_{\mathbf{i}}(x), \tag{3.20}$$

where

$$\begin{cases} a_{M,\mathbf{i}}(x) & := & a_0(x) + \sum_{j=1}^{M}\sqrt{\lambda_j}k_j(x)C_{i_j,j} \\ f_{\mathbf{i}}(x) & := & f(x)\cdot\prod_{j=1}^{M}\int_{\Gamma_j}\phi_{i_j,j}(z)\rho_j(z)dz \\ \mathbf{i} & \in & \mathbf{I} = \{\{i_1,i_2,\cdots,i_M\} \mid i_j \le n_j, \text{for} j = 1,2,\cdots,n_M\}. \end{cases} \tag{3.21}$$

The solution to the equation (3.19) is

$$u_M(x,t,\xi) = \sum_{\mathbf{i}\in\mathbf{I}} u_{M,\mathbf{i}}(x,t)\,\phi_{\mathbf{i}}(\xi). \tag{3.22}$$

3.5 Discretization in space and time

In this section, we discretize the equation (3.2) using implicit Euler formula for temporal direction and 5-point scheme for the spacial direction.

Let Δx and Δy be the mesh size in the x- and y-dimension in domain D. Suppose $u_{i,j}^k$ is the solution at the mesh point $x_i = i\cdot\Delta x, y_j = j\cdot\Delta y$, and $T_k = k\cdot\Delta t$. We know that

$$\begin{aligned} \nabla\cdot(a\nabla u) &= \frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(a\frac{\partial u}{\partial y}\right) \\ \frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right) &\approx \frac{\left(a\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2},j} - \left(a\frac{\partial u}{\partial x}\right)_{i-\frac{1}{2},j}}{\Delta x} \\ \left(a\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2},j} &\approx a_{i+\frac{1}{2},j}\frac{u_{i+1,j}-u_{i,j}}{\Delta x} \\ \left(a\frac{\partial u}{\partial x}\right)_{i-\frac{1}{2},j} &\approx a_{i-\frac{1}{2},j}\frac{u_{i,j}-u_{i-1,j}}{\Delta x}, \end{aligned}$$

then

$$\frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right) \approx \frac{1}{(\Delta x)^2}\left[a_{i+\frac{1}{2},j}u_{i+1,j} + a_{i-\frac{1}{2},j}u_{i-1,j} - \left(a_{i+\frac{1}{2},j} + a_{i-\frac{1}{2},j}\right)u_{i,j}\right],$$

and

$$\frac{\partial}{\partial y}\left(a\frac{\partial u}{\partial y}\right) \approx \frac{1}{(\Delta y)^2}\left[a_{i,j+\frac{1}{2}}u_{i,j+1} + a_{i,j-\frac{1}{2}}u_{i,j-1} - \left(a_{i,j+\frac{1}{2}} + a_{i,j-\frac{1}{2}}\right)u_{i,j}\right].$$

Hence

$$\begin{aligned}\nabla\cdot(a\nabla u) &\approx \frac{1}{(\Delta x)^2}\left[a_{i+\frac{1}{2},j}u_{i+1,j}+a_{i-\frac{1}{2},j}u_{i-1,j}-\left(a_{i+\frac{1}{2},j}+a_{i-\frac{1}{2},j}\right)u_{i,j}\right]\\ &\quad+\frac{1}{(\Delta y)^2}\left[a_{i,j+\frac{1}{2}}u_{i,j+1}+a_{i,j-\frac{1}{2}}u_{i,j-1}-\left(a_{i,j+\frac{1}{2}}+a_{i,j-\frac{1}{2}}\right)u_{i,j}\right].\end{aligned}$$

On the other hand,

$$\frac{\partial u}{\partial t}=\frac{u_{i,j}^{k+1}-u_{i,j}^{k}}{\Delta t}$$

then the implicit Euler formula yields

$$\begin{aligned}\frac{u_{i,j}^{k+1}-u_{i,j}^{k}}{\Delta t} &\approx \frac{1}{(\Delta x)^2}\left[a_{i+\frac{1}{2},j}u_{i+1,j}+a_{i-\frac{1}{2},j}u_{i-1,j}-\left(a_{i+\frac{1}{2},j}+a_{i-\frac{1}{2},j}\right)u_{i,j}\right]\\ &\quad+\frac{1}{(\Delta y)^2}\left[a_{i,j+\frac{1}{2}}u_{i,j+1}+a_{i,j-\frac{1}{2}}u_{i,j-1}-\left(a_{i,j+\frac{1}{2}}+a_{i,j-\frac{1}{2}}\right)u_{i,j}\right]\\ &\quad+f_{i,j}^{k+1}\cdot\int_{\Gamma_j}\phi_{i_j,j}(z)\rho_j(z)dz.\end{aligned}$$

Let's rewrite the above equation into the space-time formulation,

$$\begin{aligned}&\left[1+\gamma_x\left(a_{i+\frac{1}{2},j}+a_{i-\frac{1}{2},j}\right)+\gamma_y\left(a_{i,j+\frac{1}{2}}+a_{i,j-\frac{1}{2}}\right)\right]u_{i,j}^{k+1}\\ &-\gamma_x a_{i+\frac{1}{2},j}u_{i+1,j}^{k+1}-\gamma_x a_{i-\frac{1}{2},j}u_{i-1,j}^{k+1}-\gamma_y a_{i,j+\frac{1}{2}}u_{i,j+1}^{k+1}-\gamma_y a_{i,j-\frac{1}{2}}u_{i,j-1}^{k+1}-u_{i,j}^{k}\\ &=f_{i,j}^{k+1}\int_{\Gamma_j}\phi_{i_j,j}(z)\rho_j(z)dz\cdot\Delta t,\end{aligned}\tag{3.23}$$

where $\gamma_x=\Delta t/\Delta x^2,\gamma_y=\Delta t/\Delta y^2$. If $\Delta x=\Delta y$, then $\gamma_x=\gamma_y=\gamma$, we get the discretized formula

$$\begin{aligned}&\left(\frac{1}{\gamma}+a_{i+\frac{1}{2},j}+a_{i-\frac{1}{2},j}+a_{i,j+\frac{1}{2}}+a_{i,j-\frac{1}{2}}\right)u_{i,j}^{k+1}\\ &-a_{i+\frac{1}{2},j}u_{i+1,j}^{k+1}-a_{i-\frac{1}{2},j}u_{i-1,j}^{k+1}-a_{i,j+\frac{1}{2}}u_{i,j+1}^{k+1}-a_{i,j-\frac{1}{2}}u_{i,j-1}^{k+1}-\frac{1}{\gamma}u_{i,j}^{k}\\ &=\Delta x^2 f_{i,j}^{k+1}\int_{\Gamma_j}\phi_{i_j,j}(z)\rho_j(z)dz,\end{aligned}\tag{3.24}$$

Chapter 4

Recycling Krylov subspace method and grouping algorithms

Based on Theorem 2, the stochastic parabolic equation is decoupled into a sequence of deterministic parabolic equations. These deterministic problems are related and, with a proper grouping, the change within a group is small from one to the next. We expect to significantly reduce the number of iterations and the total computing time by (1) reusing the preconditioner, and (2) recycling selected vectors from the Krylov subspace generated in the previous linear system in the same group. In this section, we first propose an ordering algorithm according to the change of the parameter $C_{i_j,j}$ computed in each subspace of $P_{\mathbf{n}}$. Based on the ordering algorithm, we introduce a grouping algorithm for the one-level RAS preconditioning, and a grouping algorithm for the two-level hybrid preconditioning. Finally, we show an example to demonstrate how to utilize the grouping algorithms.

4.1 Recycling Krylov subspace method

In the past several decades, some variants of Krylov subspace methods were developed for solving large sparse systems, [11, 13, 14, 22, 36, 39, 40]. We choose to use the so-called GCRO-DR version of the recycling GMRES introduced in [38] since both the matrix and the right-hand side change in the sequence of systems. When solving a single linear system of equations, GCRO-DR is algebraically equivalent to the GMRES-DR method [36, 38], which, in each cycle, carries forward k harmonic Ritz vectors associated with k harmonic Ritz values computed at the end of the previous cycle and restarts with those vectors. When solving multiple systems, after the first system in

the group is solved, a subspace spanned by selected harmonic Ritz vectors is retained and used as the initial subspace for the other systems. More detailed description about GCRO-DR is available in [38]. To make the GCRO-DR algorithm more flexible, [28] generalizes it to a flexible version, namely, FGMRES. The framework of GCRO-DR is briefly introduced as follows.

Suppose we retain k recycled vectors in $\bar{Y}_k$ after solving the previous $(i-1)^{th}$ system, i.e.

$$\bar{Y}_k = [\bar{y}_1, \bar{y}_2, \cdots, \bar{y}_k].$$

Then GCRO-DR constructs the matrices $C_k, U_k \in R^{n\times k}$ as in the following equations, which take the approximate invariant subspace from the $(i-1)^{th}$ system.

$$C_k = Q, \quad U_k = \bar{Y}R^{-1},$$

where Q, R are obtained using reduced QR decomposition of $A_i\bar{Y}_k$ such that

$$A_iU_k = C_k, \quad C_k^HC_k = I_k.$$

Let x_0 and r_0 be the initial guess and initial residual of the i^{th} system. We write the solution over the range(U_k) by $x = x_0 + U_kC_k^Hr_0$ and set $r = r_0 - C_kC_k^Hr_0$. Suppose m is the maximum number of iterations before restarting another FGMRES cycle. A Krylov subspace $\mathcal{K}_{m-k+1}$ is produced with $(I - C_kC_k^H)A_i$ by the Arnoldi process, and the corresponding Arnoldi relation is

$$(I - C_kC_k^H)A_iV_{m-k} = V_{m-k+1}\bar{H}_{m-k}.$$

Note that the matrix V_{m-k} is produced with preconditioning in FGMRES. Since $V_{m-k+1} \perp C_k$, we have

$$A_i[U_k\ V_{m-k}] = [C_k\ V_{m-k+1}] \begin{bmatrix} I_k & B_{m-k} \\ 0 & \bar{H}_{m-k} \end{bmatrix}, \tag{4.1}$$

where $B_{m-k} = C_k^HA_iV_{m-k}$. In order to avoid the ill-conditioning in equation (4.1), we normalize the columns of U_k as $\tilde{U}_k := U_kD_k$. Defining

$$\hat{V}_m = [\tilde{U}_k\ \ V_{m-k}], \quad \hat{W}_{m+1} = [C_k\ \ V_{m-k+1}], \quad \bar{G}_m = \begin{bmatrix} D_k & B_{m-k} \\ 0 & \bar{H}_{m-k} \end{bmatrix},$$

we rewrite equation (4.1) as

$$A_i \hat{V}_m = \hat{W}_{m+1} \bar{G}_m.$$

Finally, we update the solution and evaluate the residual of the i^{th} linear system

$$\begin{aligned} x &= x + \hat{V}_m t \\ r &= r - A_i \hat{V}_m t = r - \hat{W}_{m+1} \bar{G}_m t, \end{aligned}$$

where t minimizes $\|e_{k+1}\|r\| - \hat{G}_m t\|$. More details can be found in [28, 38].

4.2 Ordering algorithm

In order to maximize the benefit of the recycling strategy, we arrange the systems following a decreasing order of the perturbation. In this way, we can determine when to restart a new Krylov subspace and a new preconditioner when the cumulative perturbation has grown too large using the current Krylov subspace and preconditioner. Theorem 2 shows that the perturbation among the decoupled systems is originated from the diffusion coefficient $a_{M,\mathbf{i}}(x)$, which consists of the mean function, eigenvalues, eigenfunctions and the constants $C_{i_j,j}$. The mean function and covariance function are stationary in the KL expansion. The eigenvalues and the corresponding normalized eigenfunctions are determined by the covariance function and they appear in all systems. Thus, we see that the constants $C_{i_j,j}$ play the dominant role in the variability of the systems.

In principle, the decreasing ordering of the perturbation can be realized when

$$\sum_{j=1}^{M} \sqrt{\lambda_j} k_j(x) C_{i_j,j} \tag{4.2}$$

is ordered, in some sense, from large to small. However, it is very expensive to obtain such ordering directly. Consequently, an alternative approach is proposed here to produce a pseudo-decreasing order of the summation (4.2) by reordering $C_{i_j,j}$ decreasingly in each subspace of the approximated random space, as shown in Table 4.1. Given $\mathbf{n} = \{n_1, n_2, \cdots, n_M\}$, the sequence of systems is indexed using a multi-index $\mathbf{i} = \{i_1, i_2, \cdots, i_M\} \in \mathbf{I}$, $0 \leq i_j \leq n_j$, $1 \leq j \leq M$. We reorder

Table 4.1: Reordering $C_{i_j,j}$ decreasingly by permutation, for all j, $0 \leq j \leq M$, $\{0, 1, \cdots, n_j\} \longrightarrow \{l_0^j, l_1^j, \cdots, l_{n_j}^j\}$.

λ_1:	$C_{0,1}$	$C_{1,1}$	$\cdots$	$C_{n_1,1}$	$\longrightarrow$	$C_{l_0^1,1}$	$C_{l_1^1,1}$	$\cdots$	$C_{l_{n_1}^1,1}$
λ_2:	$C_{0,2}$	$C_{1,2}$	$\cdots$	$C_{n_2,2}$	$\longrightarrow$	$C_{l_0^2,2}$	$C_{l_1^2,2}$	$\cdots$	$C_{l_{n_2}^2,2}$
$\vdots$		$\vdots$			$\vdots$		$\vdots$		
λ_M:	$C_{0,M}$	$C_{1,M}$	$\cdots$	$C_{n_M,M}$	$\longrightarrow$	$C_{l_0^M,M}$	$C_{l_1^M,M}$	$\cdots$	$C_{l_{n_M}^M,M}$

$\{0, 1, \cdots, n_j\}$ by a certain permutation:

$$\{0, 1, \cdots, n_j\} \longrightarrow \{l_0^j, l_1^j, \cdots, l_{n_j}^j\},$$

such that $C_{i_j,j}$ decreases in the subspace corresponding to λ_j, for all $j = 1, 2, \cdots, M$. We call this a "pseudo-decreasing" ordering because (4.2) is not strictly decreasing, and in fact it is easy to construct a counterexample by choosing appropriate values for each product component. But our experiments indicate that our ordering algorithm adequately determines a sufficiently decreasing ordering. Based on the above discussion, we summarize our ordering in the following algorithm:

Ordering Algorithm:

$k = 1$;

for $i_1 = l_0^1, l_1^1, \cdots, l_{n_1}^1$

  for $i_2 = l_0^2, l_1^2, \cdots, l_{n_2}^2$

    $\cdots$

      for $i_M = l_0^M, l_1^M, \cdots, l_{n_M}^M$

        label the k^{th} system corresponding to $\mathbf{i} = \{i_1, i_2, \cdots, i_M\}$;

        $k = k + 1$;

end all for

4.3 Grouping algorithm for one-level RAS preconditioning

The ordering algorithm generates a sequence of systems with the property that the perturbation is relatively small among systems that are nearby in the sequence. If two systems are not nearby in the sequence, then the perturbation can be quite large, which means a Krylov subspace generated by one system can be reused for some nearby systems, but not for others. In this section, we mainly discuss how to restart the recycling for the one-level RAS preconditioning.

Although there is no theory to describe how the recycled one-level RAS preconditioner im-

pacts the convergence of the sequence of systems, experiments show that it is good enough to consider a division corresponding to the constants $C_{0,1}$, $C_{1,1}$, $\cdots$, $C_{n_1,1}$ associated with the largest eigenvalue λ_1. This gives (n_1+1) groups, denoted as G_0, G_1, $\cdots$, G_{n_1}. We recycle a selected Krylov subspace and the preconditioner of the first linear system in each group.

In most cases, the perturbation within the group is quite minor so that the selected Krylov subspace and preconditioner can be reused to solve all the other systems. However, in the last group G_{n_1}, for some special cases, some of the systems are close to being singular. In this situation, recycling the Krylov subspace and preconditioner for all the systems is not a wise strategy any more. Especially for the space-time method, as shown in Theorem 1, the more time steps are coupled into one system, the worse the condition number. Therefore, it is then more important to remove the sensitive systems from the group if we want to couple more time steps into one system. One approach to address this issue is to extract those sensitive systems from the group, and choose a smaller window size s to make them better conditioned, then solve them separately. In order to single out the sensitive systems, we set up a cutoff parameter $\gamma(\geq 0)$ as a criterion to evaluate the minimum of $a_{M,\mathbf{i}}(x)$ on a coarse mesh $\mathcal{V}_C^H$ of the spacial domain D, i.e.,

$$\min_{x\in\mathcal{V}_C^H}\{a_{M,\mathbf{i}}(x)\} < \gamma. \tag{4.3}$$

When the inequality (4.3) holds, we remove its corresponding system from the group and label it as nearly singular. All these nearly singular systems are collected together to form another group, called the "bad" group, G_b.

According to the above analysis, we summarize our grouping algorithm for the one-level RAS preconditioning as follows: For a given $\mathbf{n} = (n_1, n_2, \cdots, n_M)$, there are $(n_1+1)\times(n_2+1)\times\cdots\times(n_M+1)$ systems to be solved. Assume the systems are ordered by the ordering algorithm proposed in the previous section.

Grouping algorithm for one-level RAS preconditioning:

- Divide all the systems evenly into (n_1+1) groups, **i.e.**, $G_0, G_1, \cdots, G_{n_1}$, with each group

containing $(n_2+1)\times\cdots\times(n_M+1)$ systems.

- Choose an appropriate cutoff parameter $\gamma \geq 0$, extract all the systems in the last group G_{n_1} that satisfy the inequality (4.3), and collect them in the bad group G_b. The modified last group is then denoted as $\overline{G}_{n_1} = G_{n_1} \setminus G_b$.

- For the regular groups, take a relatively large number of time steps coupled into one system. In each group, construct a Krylov subspace and a preconditioner from the first system, then recycle them when solving the other systems within the same group.

- For the bad group G_b, take a relatively smaller number of time steps to make all the systems in G_b solvable. Construct a Krylov subspace and a preconditioner from one of the systems, then recycle the Krylov subspace and the symbolic factorization of the submatrix solver for the other systems in G_b.

Notice that the choice of the value γ depends on the specific problem. For some problems with $0 < a_1 \leq a(x,\omega) \leq a_2 < \infty$, where a_1 is not too close to zero, we don't need to set up such a cutoff parameter γ, since there is no sensitive system in this case.

4.4 Grouping algorithm for two-level hybrid preconditioning

The grouping algorithm for one-level RAS doesn't work well when the two-level hybrid preconditioner is employed. This is because the additional coarse level correction on the hybrid preconditioner greatly improves the condition numbers of the systems, then the numbers of iterations are drastically reduced. However, the smaller Krylov subspace generated in the reduced iterations yields fewer subsequent systems fitting the recycled Krylov subspace and preconditioner. Therefore, we need another grouping algorithm for the two-level hybrid preconditioning. The algebraic average of the eigenvalues provides a clue as to how to further divide the groups of systems.

We first evenly divide all the systems into (n_1+1) groups, denoted as G_0, G_1, $\cdots$, G_{n_1}, as

in the section 4.2. Then we average the M eigenvalues, denoted as $\overline{\lambda}$, as following

$$\lambda_p > \overline{\lambda} = \frac{1}{M}\sum_{i=1}^{M}\lambda_i > \lambda_{p+1}, \quad 2 \le p \le M,$$

where the index p works as a divider. We divide each of the groups G_0, G_1, $\cdots$, G_{n_1-1} uniformly into $(n_2+1)\times(n_3+1)\times\cdots\times(n_p+1)$ subgroups with $(n_{p+1}+1)\times(n_{p+2}+1)\times\cdots\times(n_M+1)$ systems in each subgroup.

Since the last group G_{n_1} contains the systems that may be close to be singular, the systems in this group can be sensitive. So we divide the group G_{n_1} uniformly into $(n_2+1)\times(n_3+1)\times\cdots\times(n_p+1)\times(n_{p+1}+1)$ subgroups with $(n_{p+2}+1)\times(n_{p+3}+1)\times\cdots\times(n_M+1)$ systems in each subgroup. As in the grouping algorithm for one-level RAS preconditioning, the sensitive systems are gathered into the last group G_{n_1}, which is extracted from some of the subgroups in G_{n_1} by the cutoff parameter $\gamma(\ge 0)$. All the sensitive systems satisfying (4.3) are put in the bad group G_b.

For the regular subgroups in G_0, G_1, $\cdots$, G_{n_1-1} as well as all the subgroups in G_{n_1} after the sensitive systems are extracted, we construct a Krylov subspace and a preconditioner from the first system, then recycle them when solving the other systems within the same subgroup. For the bad group G_b, we use the same strategy as the one-level RAS preconditioning.

The grouping algorithm for two-level hybrid preconditioning is thus organized as follows: All the assumptions are the same as the grouping algorithm for the one-level RAS preconditioning, and all the decoupled systems are already ordered by the ordering algorithm.

Grouping algorithm for two-level hybrid preconditioning:

- Follow step 1 of the grouping algorithm for the one-level RAS preconditioning, we have (n_1+1) groups, $G_0, G_1, \cdots, G_{n_1}$.
- Average the M eigenvalues to obtain $\overline{\lambda}$, and then compare it with all the eigenvalues to find the index p, such that $\lambda_p > \overline{\lambda} > \lambda_{p+1}, 2 \le p \le M$.
- In groups $G_0, G_1, \cdots, G_{n_1-1}$, each group is uniformly divided into $(n_2+1)\times(n_3+1)\times\cdots\times$

(n_p+1) subgroups with $(n_{p+1}+1)\times(n_{p+2}+1)\times\cdots\times(n_M+1)$ systems in each subgroup. For the last group G_{n_1}, we divide it uniformly into $(n_2+1)\times(n_3+1)\times\cdots\times(n_p+1)\times(n_{p+1}+1)$ subgroups with $(n_{p+2}+1)\times(n_{p+3}+1)\times\cdots\times(n_M+1)$ systems in each subgroup.

- Choose an appropriate cutoff parameter $\gamma \geq 0$, extract all the systems from the subgroups in G_{n_1} that satisfy (4.3), and collect them in the bad group G_b.
- For the regular subgroups and the subgroups from which sensitive systems are extracted, we construct a Krylov subspace and a preconditioner from the first system, then recycle them within the same subgroup.
- Then follow step 4 of the grouping algorithm for the one-level RAS preconditioning.

4.5 Numerical examples

We illustrate our grouping algorithms by the following example [30] with the mean and covariance functions:

$$a_0(x) = 3 + sin(\pi x_1) \qquad C_a(x,x') = e^{-|x-x'|^2}, \quad x \in [0,1]^2. \tag{4.4}$$

The series expansion of $a(x,\xi)$ is truncated at $M = 11$ by the decay of the eigenvalues. The selection algorithm proposed in [16] gives the dimensions in each subspaces: $\mathbf{n} = (3,2,2,1,1,1,1,1,1,1,1)$, which means we obtain 9216 linear systems to be solved. Let the random variable ξ_j in the KL expansion be uniformly distributed in $\Gamma_j = [-\sqrt{3},\sqrt{3}]$, $j = 1,2,\cdots,M$. For the one-level RAS preconditioner, the total 9216 systems are evenly divided into $n_1+1 = 3+1 = 4$ groups with 2304 systems in each group.

For the squared-exponential kernel, the eigenvalues and eigenfunctions can be analytically computed by Zhu et al. in [47]. Alternatively, in our experiments, we calculate the eigenvalues and eigenfunctions in terms of the matrix associated with the covariance function $C_a(x,x')$.

Firstly, we consider the one-dimensional case:

$$C_a(x,x') = e^{-|x-x'|^2}, \quad x,x' \in [0,1].$$

Suppose we evenly divide the sample interval $[0,1]$ into n subdivisions, then choose the middle points of each subdivision as the grid points. Now the discretized matrix $\bar{C}$ associated with the covariance function has the entries as follows:

$$\bar{C}_{i,j} = C_a(x_i, x_j) = e^{-|x_i - x_j|^2} \cdot \frac{1}{n},$$

where $x_k = \frac{1}{n}(k - 0.5)$, for $k = 1, 2, \cdots, n$. Using MATLAB, we get the eigenvalues $\lambda_{1,i}$ and its corresponding eigenfunctions $v_{1,i}$, for $i = 1, 2, \cdots, n$ of the matrix $\bar{C}$. Similarly, we obtain the eigenpairs $\lambda_{2,j}$ and $v_{2,j}$ of $C_a(y, y') = e^{-|y-y'|^2}$, $y, y' \in [0,1]$, $j = 1, 2, \cdots, n$.

For the two-dimensional case:

$$C_a(x, x') = e^{-|x-x'|^2}, \quad x, x' \in [0,1]^2,$$

the eigenvalues and eigenfunctions are

$$\lambda_k = \lambda_{1,i}\lambda_{2,j}, \quad v_k = v_{1,i}v_{2,j}.$$

Since the largest 4 eigenvalues of $C_a(x, x')$ and $C_a(y, y')$, $x, x', y, y' \in [0,1]$, shown in Fig.4.1, are $\lambda_{1,1} = \lambda_{2,1} = 0.8648$, $\lambda_{1,2} = \lambda_{2,2} = 0.1262$, $\lambda_{1,3} = \lambda_{2,3} = 0.0086$, $\lambda_{1,4} = \lambda_{2,4} = 0.0004$, the largest 11 eigenvalues for $C_a(x, x') = e^{-|x-x'|^2}, x \in [0,1]^2$ are $\lambda_1 = 0.7480, \lambda_2 = \lambda_3 = 0.1092, \lambda_4 = 0.0159, \lambda_5 = \lambda_6 = 0.0074, \lambda_7 = \lambda_8 = 0.00108, \lambda_9 = \lambda_{10} = 0.00032, \lambda_{11} = 0.00007$ as plotted in Fig.4.2. The four eigenfunctions corresponding to the largest four eigenvalues are plotted in Fig.4.3, Fig.4.4, Fig.4.5, and Fig.4.6.

Next, we calculate the non-zero constant parameters $C_{p,j}$ in equation (3.17). Since we set $\dim P_{n_1} = 3$, then there are 4 double orthogonal basis functions in P_{n_1}. Suppose those double orthogonal basis have the following form:

$$\phi_i = c_{i,0} + c_{i,1}x + c_{i,2}x^2 + c_{i,3}x^3 \text{ for } i = 0, 1, 2, 3. \tag{4.5}$$

where $c_{i,j}$ are the coefficients to be determined. Plugging all the basis functions into equation (3.17), we have

$$\begin{cases} \int_{\Gamma_1} \phi_p(t)\phi_q(t)\rho(t)dt &= \sum_{k=0}^{3}\sum_{l=0}^{3} c_{p,k}b_{k,l}c_{q,l} = \delta_{p,q}, & p, q = 0, 1, 2, 3, \\ \int_{\Gamma_1} t\phi_p(t)\phi_q(t)\rho(t)dt &= \sum_{k=0}^{3}\sum_{l=0}^{3} c_{p,k}a_{k,l}c_{q,l} = C_{p,1}\delta_{p,q} & p, q = 0, 1, 2, 3 \end{cases} \tag{4.6}$$

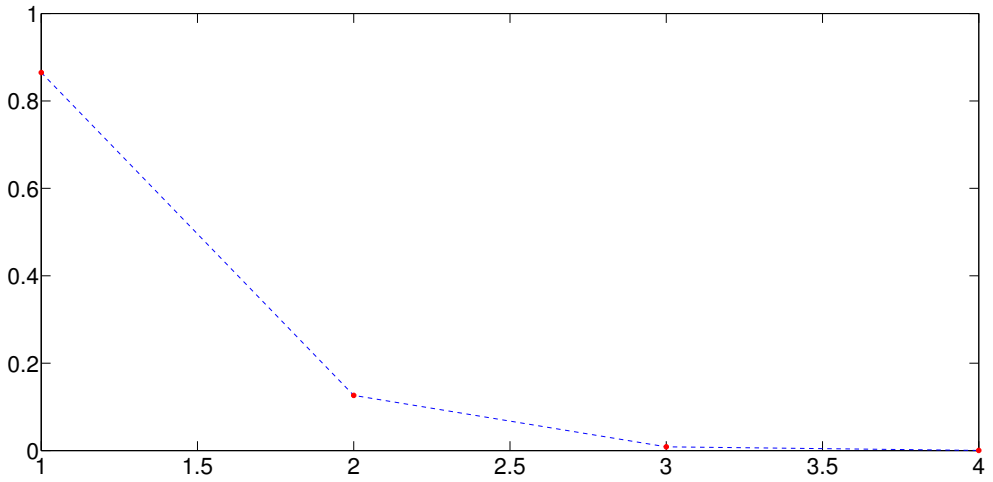

Figure 4.1: The largest 4 eigenvalues of $C_a(x, x')$, $x, x' \in [0, 1]$.

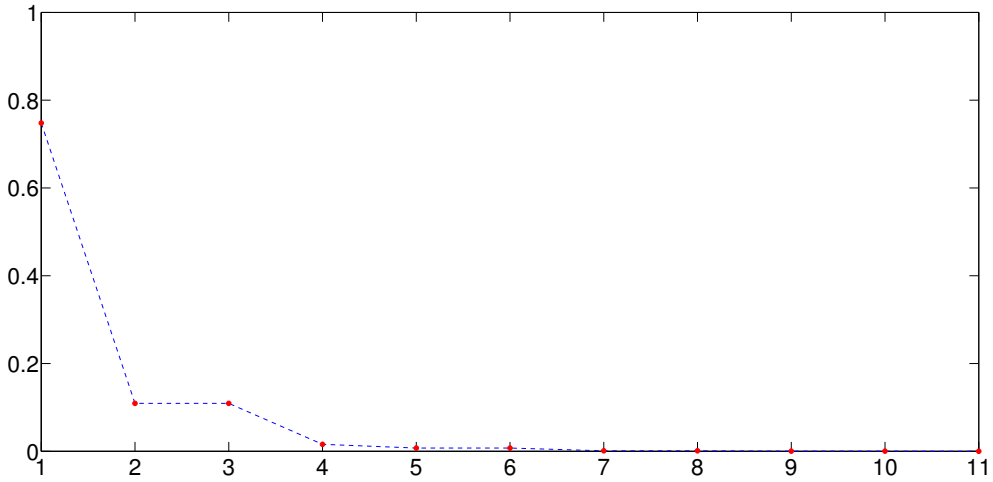

Figure 4.2: The largest 11 eigenvalues of $C_a(x, x')$, $x, x' \in [0, 1]^2$.

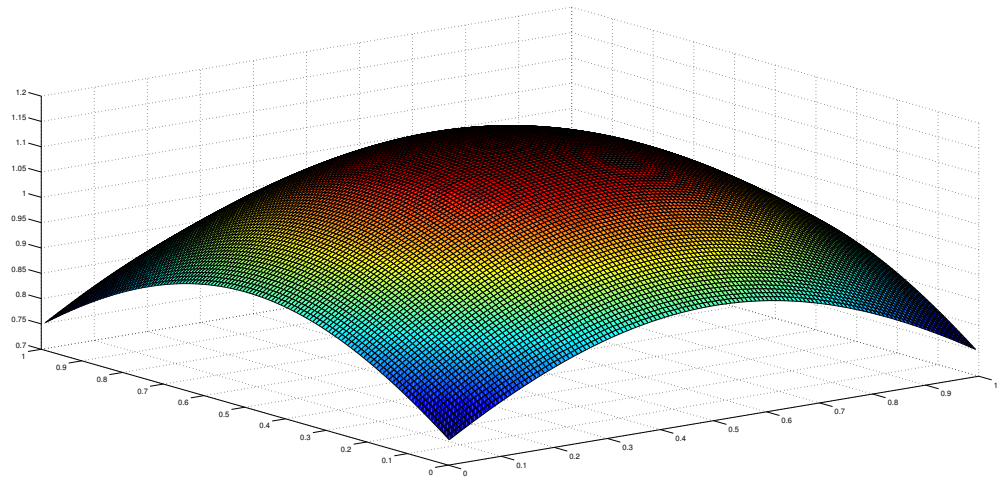

Figure 4.3: The first eigenfunction of $C_a(x, x')$, $x, x' \in [0, 1]^2$.

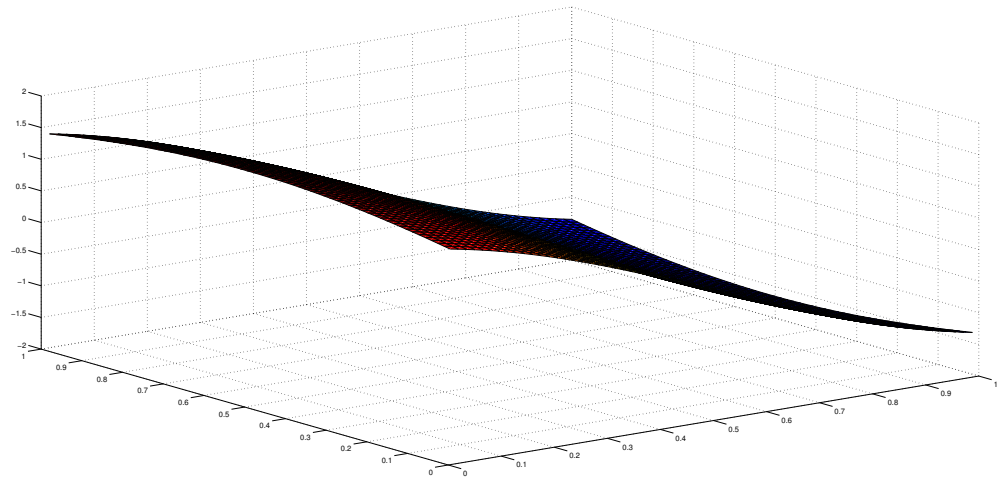

Figure 4.4: The second eigenfunction of $C_a(x, x')$, $x, x' \in [0, 1]^2$.

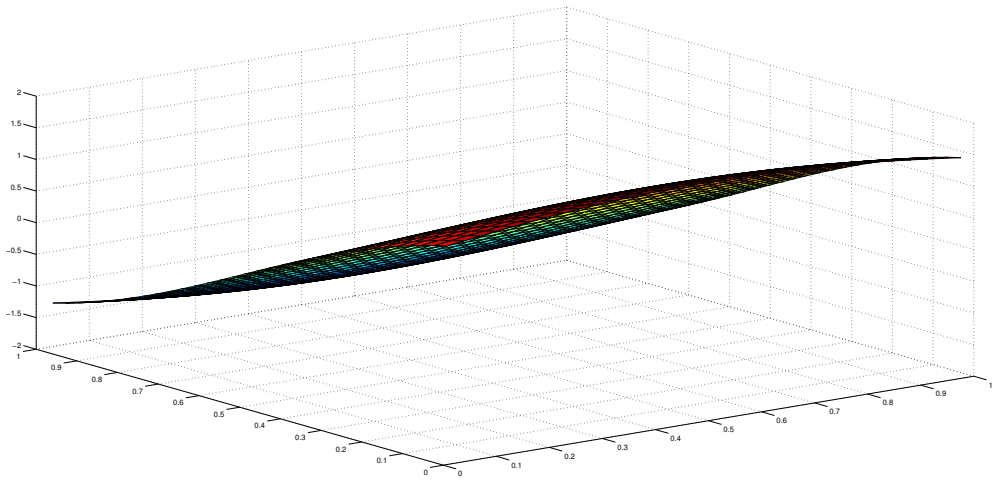

Figure 4.5: The third eigenfunction of $C_a(x, x')$, $x, x' \in [0, 1]^2$.

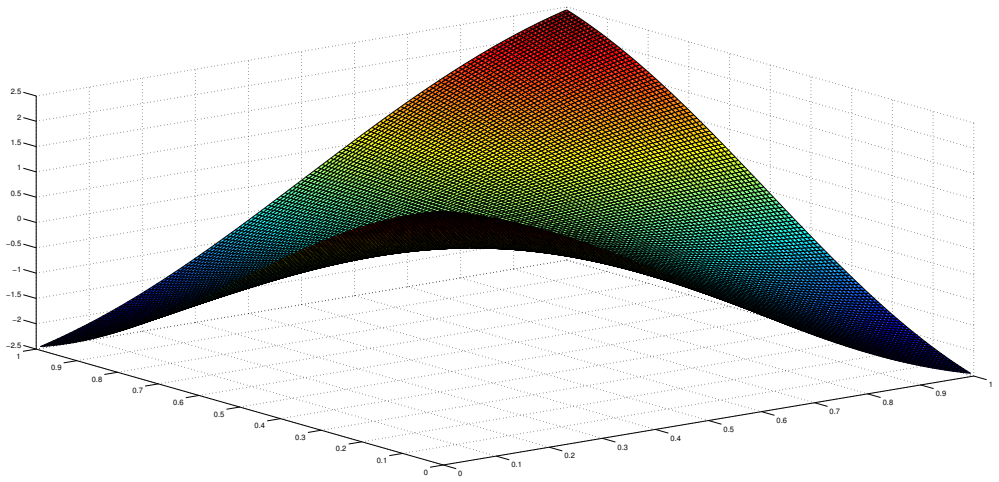

Figure 4.6: The fourth eigenfunction of $C_a(x, x')$, $x, x' \in [0, 1]^2$.

where

$$b_{k,l} = \frac{1}{2\sqrt{3}} \int_{-\sqrt{3}}^{\sqrt{3}} t^k t^l dt \quad \text{and} \quad a_{k,l} = \frac{1}{2\sqrt{3}} \int_{-\sqrt{3}}^{\sqrt{3}} t t^k t^l dt.$$

Rewrite it into the matrix form,

$$XBX^T = I, \quad \text{and} \quad XAX^T = \text{Diag}\{C_{0,1}, C_{1,1}, C_{2,1}, C_{3,1}\}, \tag{4.7}$$

where

$$X = (c_{k,l}), \quad k, l = 0, 1, 2, 3$$

$$A = (a_{k,l}), \quad k, l = 0, 1, 2, 3$$

$$B = (b_{k,l}), \quad k, l = 0, 1, 2, 3.$$

The algorithm 8.7.1 in [21] is employed to calculate the matrix X and $\text{Diag}\{C_{0,1}, C_{1,1}, C_{2,1}, C_{3,1}\}$:

Given matrices $A = A^T \in \mathbb{R}^{n\times n}$ and $B = B^T \in \mathbb{R}^{n\times n}$ with B positive definite, the following algorithm computes a nonsingular X such that

$$XBX^T = I_n \quad XAX^T = \text{Diag}(a_1, a_2, \cdots, a_n).$$

- Compute the Cholesky factorization $B = GG^T$, where G is a lower triangular matrix.
- Compute $C = G^{-1}AG^{-T}$.
- Use the symmetric QR algorithm to compute the Schur decomposition $Q^TCQ = \text{diag}(a_1, \cdots, a_n)$.
- Set $X = Q^TG^{-1}$.

Therefore, we get

$$X = \begin{pmatrix} -0.2214 & 0.1484 & 0.6384 & -0.4280 \\ -1.0373 & 1.7615 & 0.4663 & -0.7918 \\ 0.2214 & 0.1484 & -0.6384 & -0.4280 \\ -1.0373 & -1.7615 & 0.4663 & 0.7918 \end{pmatrix} \tag{4.8}$$

$$\begin{cases} C_{0,1} = 1.49153184392336179975 \\ C_{1,1} = 0.58886444109925895063 \\ C_{2,1} = -1.49153184392336513042 \\ C_{3,1} = -0.58886444109925928370. \end{cases}$$

Similarly, in P_{n_2} and P_{n_3}, the computed X and the diagonals are:

$$X = \begin{pmatrix} 0.0000 & -0.7071 & 0.5270 \\ -1.5000 & 0.0000 & 0.8333 \\ 0.0000 & 0.7071 & 0.5270 \end{pmatrix}$$

$$\begin{cases} C_{0,2} = C_{0,3} = -1.34164078649987383862 \\ C_{1,2} = C_{1,3} = 0.00000000000000008500 \\ C_{2,2} = C_{2,3} = 1.34164078649987406067. \end{cases}$$

For the space P_{n_j}, $j = 4, \cdots, 11$, the computed values are

$$X = \begin{pmatrix} -0.7071 & 0.7071 \\ 0.7071 & 0.7071 \end{pmatrix}$$

$$\begin{cases} C_{0,4} = \cdots = C_{0,11} = -1.00000000000000000000 \\ C_{1,4} = \cdots = C_{0,11} = 1.00000000000000000000. \end{cases}$$

In the following paragraphs, we provide some analysis to reveal the relationship between the eigenvalues λ_i and the constant parameters $C_{p,j}$ in the double orthogonal basis. Based on the results of our experiments, we notice that the largest eigenvalue $\lambda_1 = 0.7480$ plays an important role in the perturbation of the diffusion coefficient. $n_1 = 3$ means the first subspace associated with the largest eigenvalue has four important constants $C_{0,1} = 1.49$, $C_{1,1} = 0.59$, $C_{2,1} = -1.49$, $C_{3,1} = -0.59$. By our ordering algorithm and grouping algorithm for the one-level RAS preconditioning, we divide all the systems uniformly into 4 groups: G_0, G_1, G_2, G_3, with 2304 systems in each one. The systems in

G_0, G_1, G_2, G_3 correspond to $C_{0,1}, C_{1,1}, C_{3,1}, C_{2,1}$ respectively, since $C_{i_j,j}$ are ordered decreasingly. Fig.4.7 illustrates the maximum and minimum of the diffusion coefficient $a_{M,\mathbf{i}}(x)$, $1 \leq \mathbf{i} \leq 9216$. The four groups are plotted in Fig.4.7 with different colors. For the two-level hybrid preconditioning, the average of M eigenvalues is $\lambda_3 > \overline{\lambda} = 0.0909 > \lambda_4$, so we divide groups G_0, G_1, G_2 uniformly into $(n_2+1) \times (n_3+1) = (2+1) \times (2+1) = 9$ subgroups with 256 systems in every subgroup. For the last group G_3, it is divided into $(n_2+1) \times (n_3+1) \times (n_4+1) = (2+1) \times (2+1) \times (1+1) = 18$ subgroups with 128 systems in each one.

In Fig.4.7, we can easily notice that there are four points at which the diffusion coefficients curve crosses the x-axis, each of those represents four systems. We set the cutoff parameter $\gamma = 0$ to separate those sensitive systems and form the bad group. For the bad group, we need to find a proper window size s to make sure all these sensitive systems are solvable. In our experiments, we couple 2 time steps for the bad group, and up to 64 time steps for the other regular groups. Since the extrema of the diffusion coefficients are relatively close for the systems in the bad group, and the matrix pattern is exactly the same, we construct the Krylov subspace and the symbolic factorization of subdomain matrices from one of the systems and recycle them throughout the bad group.

We also plot the extrema singular values and condition numbers for the 9216 systems without preconditioning in Fig.4.8, Fig.4.9 and Fig.4.10. All the systems are coupled by 16 time steps for the regular groups, but only one time step for the bad group. Fig.4.8 describes the maximum singular values of all the systems without any preconditioning. Note that the graph is similar, to certain extent, to the maximum of diffusion coefficients in Fig.4.7. In Fig.4.8 and Fig.4.9, it's easy to observe that there are again four points with minimum singular values and condition numbers that are dramatically distinct from the others. These correspond to the situation identified in Fig.4.7 and constitute the bad group.

For comparison, we also plot the extrema singular values and condition numbers for all the systems with the two-level hybrid preconditioner in Fig.4.11, Fig.4.12 and Fig.4.13. After applying the preconditioner on the systems, the maximum singular values reduced by about 50% and the

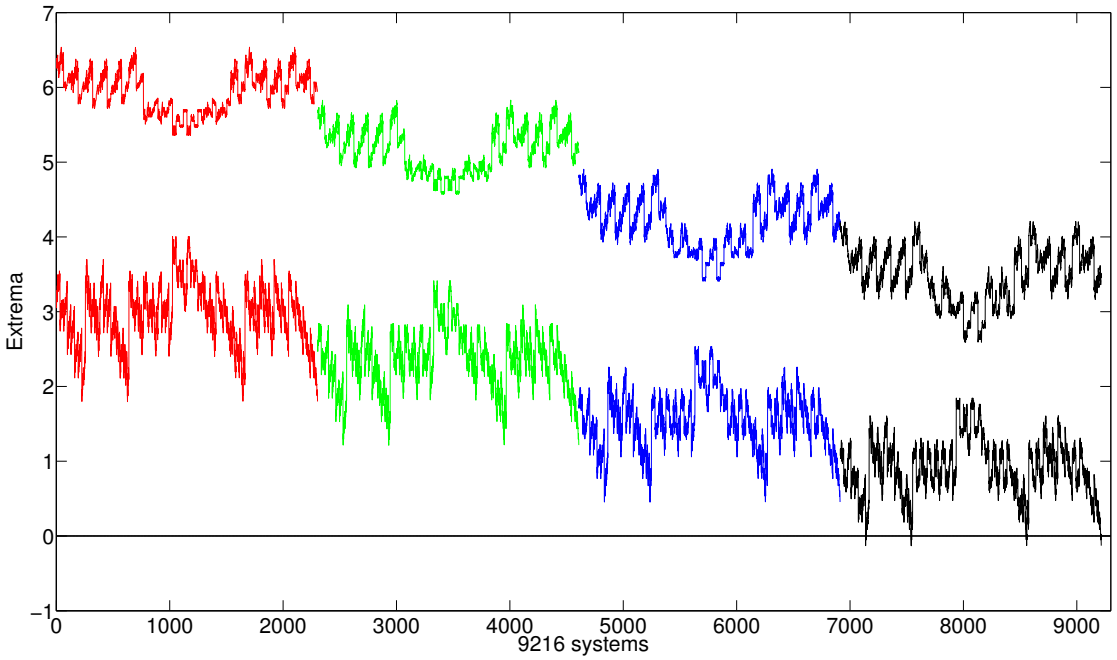

Figure 4.7: The maxima and minima of the diffusion coefficients $a_{M,\mathbf{i}}(x)$ for all 9216 systems.

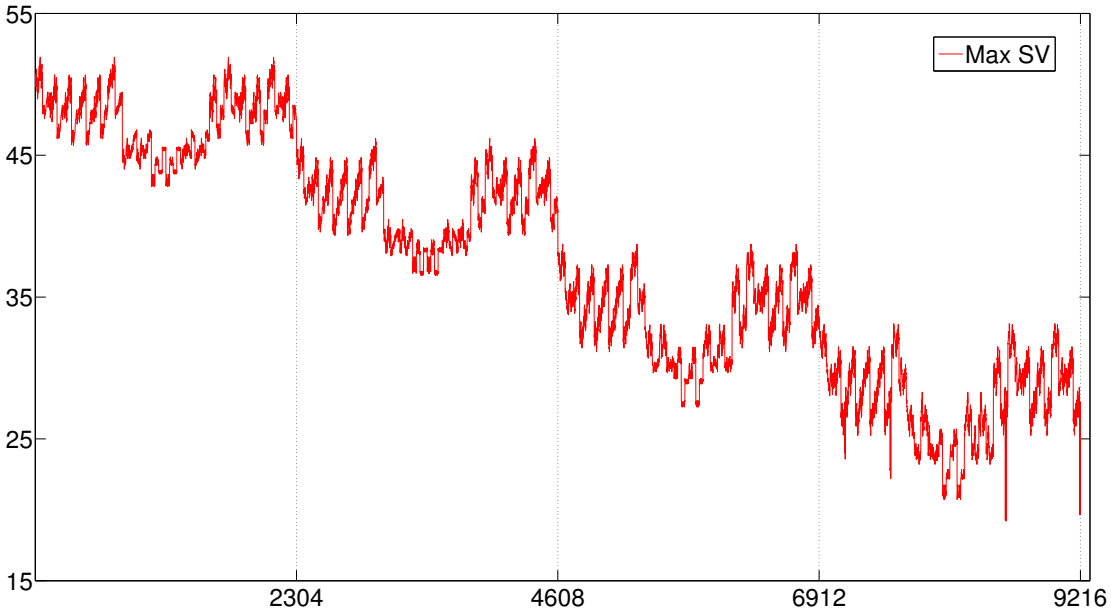

Figure 4.8: The maxima singular values of all 9216 systems without any preconditioning.

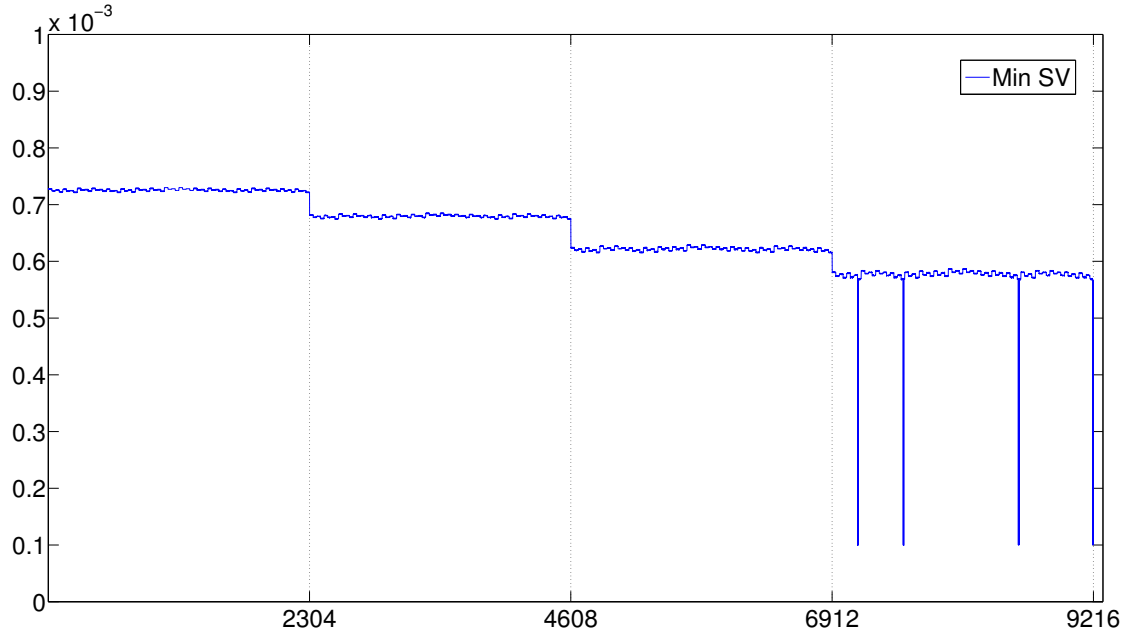

Figure 4.9: The minima singular values of all 9216 systems without any preconditioning.

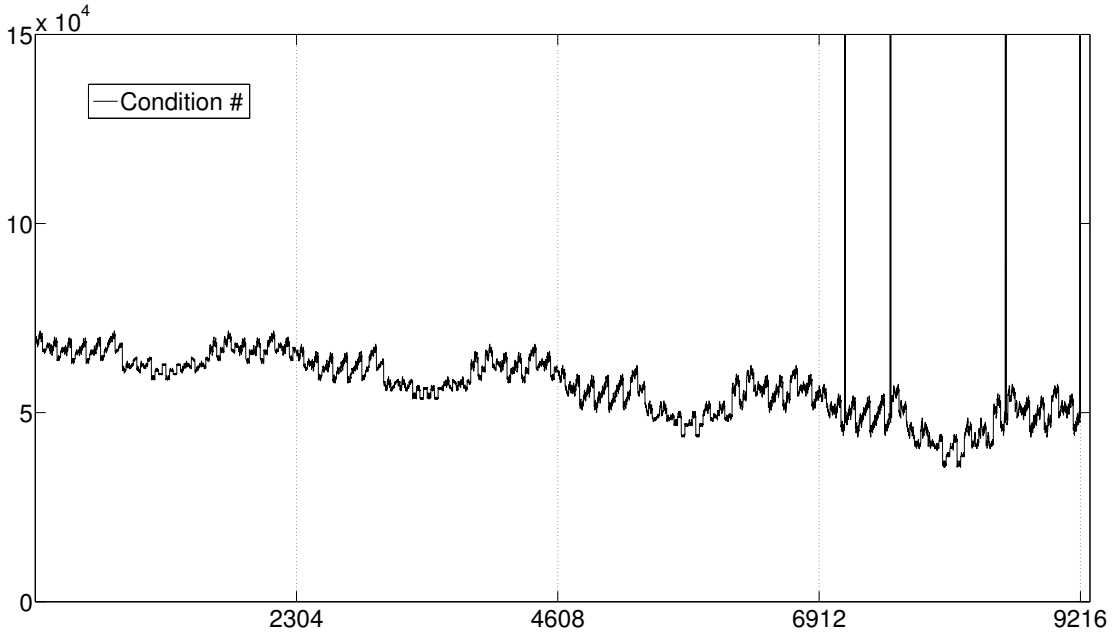

Figure 4.10: The condition numbers of all 9216 systems without any preconditioning.

minimum singular values increase by almost 1000 times. Hence, the resulting condition numbers are reduced to about one thousandth of the condition numbers without any preconditioning. Because of the improved condition numbers, the systems in the bad group may include more time steps together. In Fig.4.11 and Fig.4.12, we couple 2 or 4 time steps for the systems in the bad group, such that all the maximum singular values in the bad group are larger than the regular systems and all the minimum singular values in the bad group are smaller than the regular systems, which can be seen in the figures at the distinctive four bunches of cusps.

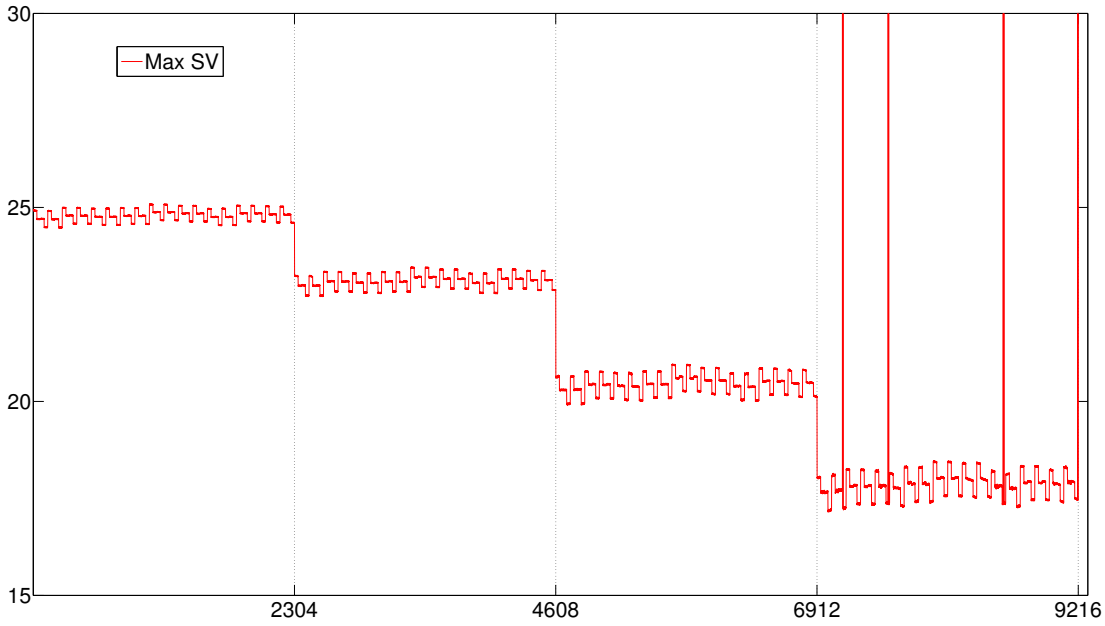

Figure 4.11: The maxima singular values of all 9216 systems with the two-level hybrid preconditioner.

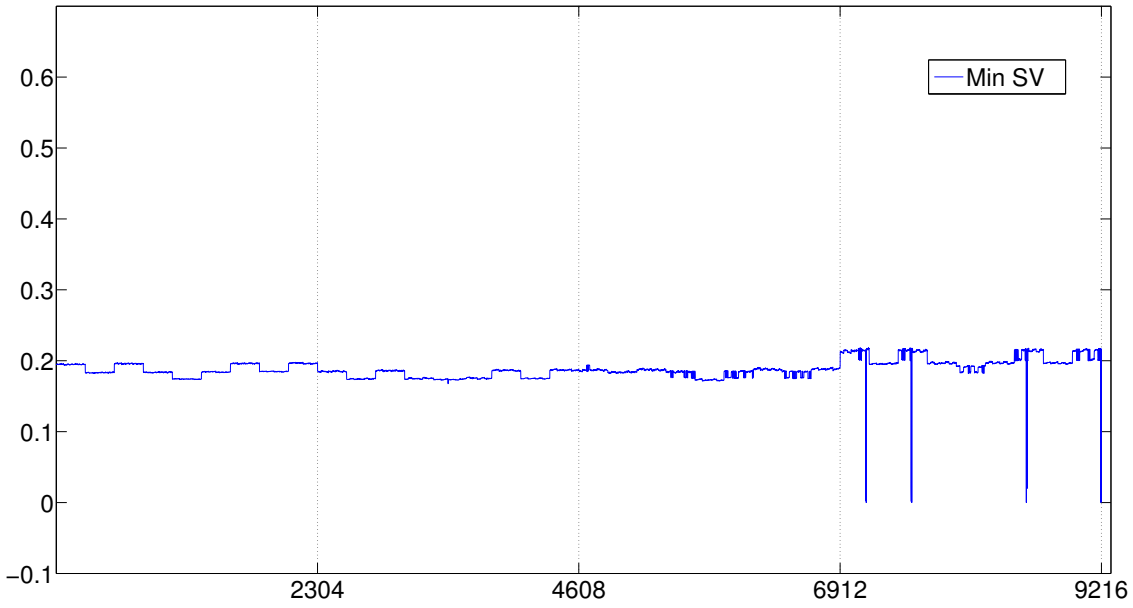

Figure 4.12: The minima singular values of all 9216 systems with the two-level hybrid preconditioner.

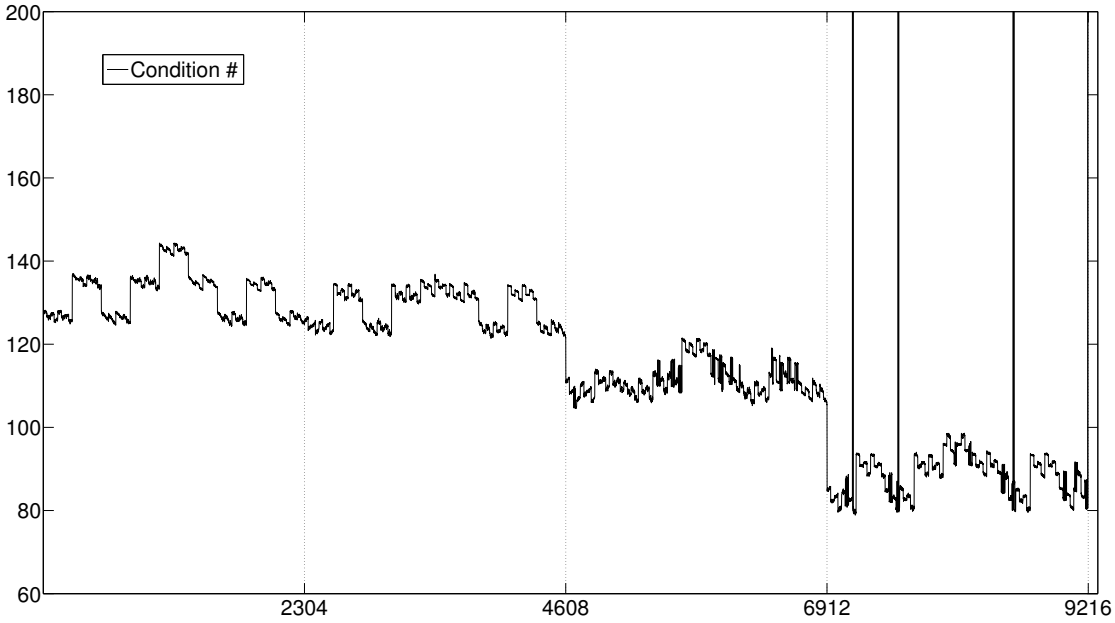

Figure 4.13: The condition numbers of all 9216 systems with the two-level hybrid preconditioner.

Chapter 5

Numerical experiments

In this chapter, we report some numerical experiments to illustrate the performance of the Schwarz preconditioned recycling FGMRES of [38] used together with our ordering and grouping algorithms. The software is developed using PETSc [5] and tested on the parallel supercomputer Janus at University of Colorado Boulder. We consider the stochastic parabolic equation (3.2) on a two-dimensional domain $D = [0, 1]^2$, $f(x) = 1$, $u^0(x) = 0$, the mean and covariance functions are given in equation (4.4), and all other parameters are the same as in Section 4.4. All the tests use $\Delta t = 0.001$, except the ones in Table 5.7, where we show the tests with different time step sizes. The stopping criteria for FGMRES, i.e., the relative tolerance and the absolute tolerance, are both 10^{-6}.

5.1 Identifying the dimension of the recycling Krylov subsapce

In this section, we numerically investigate the suitable dimension of the recycling Krylov subspace for the one-level and two-level preconditionings respectively.

We firstly test the one-level RAS preconditioning on a 497×497 mesh. The overlap between adjacent subdomains is $\delta = 8$. For the regular groups, we set the window size to $s = 16$. For the bad group, we set $s = 2$, which is the largest window size such that the algorithm does not stagnate. In this test, the number of processors is $np = 1024$. In each group G_i, we solve the first system by the non-recycled and non-restarted FGMRES method, and record the number of iterations as k_i.

It is not feasible to test all possible dimensions for the recycling subspace, we therefore take

$$k_m = \min_i k_i, \tag{5.1}$$

then set the dimension k_r as a percentage of k_m as shown in Table 5.1.

In Table 5.1, we report the computing time (the unit is second) and the average number of iterations (denoted by "aiter" in the tables) for various dimensions of the recycling subspaces. k_m is defined as in (5.1). The total number of processors is 1024, and the number of unknowns of the systems in the regular groups is 3952144. "best" represents the dimension of the recycling Krylov subspace corresponding to the minimal computing time, which is marked in red. "total" represents the total dimension, i.e., the summation of "best" and "aiter" corresponding to the minimal computing time. From this table, we notice that when ILU(0) is used as the subdomain solver in the one-level RAS preconditioner, the best computing time is obtained when the recycling subspace is $k_r = 30\%k_m$, which is approximately 65. As the fill-in of ILU increases, the percentage becomes larger in order to reduce the computing time. Since k_m is relatively large for the one-level RAS preconditioning, the computing cost of Arnoldi process cannot be ignored when we calculate the total number of iterations (refer to the last row in Table 5.1), which means the total number of iterations consists of the actual number of iterations and the size of the Krylov subspace we kept for recycling. Therefore, the total number of iterations decreases when a more accurate subdomain solver is applied. We notice that the computing time also decreases except ILU(3) where the computing time increases slightly in column 4. This is because the total numbers of iterations with ILU(2) and ILU(3) are pretty close, but ILU(3) is more expensive.

We also test the restarted FGMRES for the one-level RAS preconditioning in Table 5.2. As the subdomain solver becomes more accurate, the computing time and the number of iterations both decrease. But the best result for the restarted FGMRES is worse than the best one in Table 5.1. Therefore, we conclude that the one-level RAS preconditioner with LU subdomain solver is the best choice.

Next, we investigate the choice of recycling dimension for the two-level hybrid precondition-

Table 5.1: Computing time (second) and average number of iterations (denoted by "aiter") for the one-level RAS preconditioned FGMRES without restart, $\delta = 8$, ILU($*$) and LU are subdomain solvers.

	ILU(0)		ILU(1)		ILU(2)		ILU(3)		LU	
	$k_m = 216$		$k_m = 134$		$k_m = 108$		$k_m = 82$		$k_m = 40$	
$k_r =$	time	aiter	time	aiter	time	aiter	time	aiter	time	aiter
$100\%k_m$	90565	3.80	35279	3.97	23829	3.99	15603	4.22	9866	7.17
$90\%k_m$	73206	4.32	29227	4.49	20079	4.84	13343	4.95	9156	7.15
$80\%k_m$	60058	4.94	24376	5.09	16805	5.54	11301	6.13	9162	7.64
$70\%k_m$	45446	5.72	19196	6.18	13646	6.78	10405	10.12	9590	8.86
$60\%k_m$	35317	7.02	14618	7.45	12767	8.84	10830	15.87	11375	9.35
$50\%k_m$	26388	8.54	11447	9.51	10164	14.55	11153	21.51	12136	11.44
$40\%k_m$	18460	11.71	10865	19.96	11907	29.90	14158	31.47	11972	12.93
$30\%k_m$	13382	16.44	12947	42.66	13983	44.07	15095	41.47	15430	17.45
$20\%k_m$	17015	58.10	15772	66.81	16334	61.71	20334	55.98	18329	21.31
$10\%k_m$	30433	125.7	20628	98.92	20536	85.86	20943	67.76	32814	39.01
best	65		54		54		57		36	
total	81.44		73.96		68.55		67.12		43.15	

Table 5.2: Computing time (second) and average number of iterations (denoted by "aiter") for the one-level RAS preconditioned FGMRES with restart ($= 50$), $\delta = 8$.

	ILU(0)		ILU(1)		ILU(2)		ILU(3)		LU	
k_r	time	aiter	time	aiter	time	aiter	time	aiter	time	aiter
10	22715	195.19	17870	117.04	19315	93.73	19124	66.28	18554	19.27
20	22671	155.77	16631	91.08	16606	71.49	17245	52.48	11334	11.74
30	23787	120.58	16411	68.41	15020	50.79	14645	36.40	9743	8.51
40	29794	85.13	18380	43.70	18563	39.89	19230	30.09	11367	7.00

Table 5.3: Computing time (second) and average number of iterations (denoted by "aiter") for the two-level hybrid preconditioned FGMRES without restart. $\delta = 8$, the coarse overlap $\delta_c = 0$. k_m is defined in (5.1).

	ILU(0)		ILU(1)		ILU(2)		ILU(3)		LU	
	$k_m = 13$		$k_m = 8$		$k_m = 7$		$k_m = 5$		$k_m = 4$	
	time	aiter	time	aiter	time	aiter	time	aiter	time	aiter
$k_r = k_m$	1278	4.38	2043	5.91	2608	5.39	3563	4.88	14268	5.16
$k_r = k_m - 1$	1326	4.42	2090	6.01	2576	5.45	3509	4.88	11084	5.18
$k_r = k_m - 2$	1353	5.04	2374	6.15	2583	5.49	3606	4.92	12564	5.06
$k_r = k_m - 3$	2083	7.31	2112	6.18	2600	5.56	4889	5.02	11191	5.17

ing in Table 5.3. The coarse mesh is 32×32 and the fine mesh is 497×497. The computing time increases when a more accurate subdomain solver is employed. However, the average numbers of iterations oscillate slightly between 4 and 6. There are two main factors that impact the number of iterations. (1) the dimension of the recycling Krylov subspace. A larger recycling Krylov subspace usually yields to a faster convergence. If the recycling Krylov subspace is too small, such as the last column in Table 5.3, more iterations are necessary. (2) the subdomain solver. A more accurate subdomain solver implies fewer number of iterations. These two opposite factors working together yield the oscillation of the iterations in Table 5.3. For the computing time, the subdomain solver is dominant when the numbers of iterations are close. Hence, the expensive subdomain solver yields larger computing time. Compared with the one-level RAS preconditioner, the two-level hybrid preconditioner results in a better convergence with ILU(0) and the largest recycling Krylov subspace.

5.2 Comparing several recycling strategies

In this section, we compare four recycling strategies. The parameters correspond to the optimal combinations of one- and two-level preconditionings obtained in the last section. The number of systems in the regular groups is 9200, the other 16 sensitive systems are contained in the bad group G_b. The four schemes to be compared are listed below, in which we mainly make comparisons for the systems in the regular groups. For the systems in the bad group G_b, we only compare the

Table 5.4: Computing time (second) and average number of iterations (denoted by "aiter") of four schemes.

	Scheme 1		Scheme 2		Scheme 3		Scheme 4	
preconditioning	time	aiter	time	aiter	time	aiter	time	aiter
one-level RAS	43733	48.61	30472	8.44	13537	11.90	9371	7.15
two-level hybrid	4817	15.86	3817	8.06	4495	15.50	1257	4.38

number of iterations for Scheme 1 and Scheme 4.

1. Solve all the systems separately without any recycling.

2. Recycle the Krylov subspace and the symbolic factorizations of subdomain matrices, both constructed from the first system, throughout all the other 9199 systems.

3. Keep the Krylov subspace and the preconditioner, both constructed from the first system, and then recycle them throughout all the other 9199 systems.

4. Apply the one- and two-level grouping algorithm.

First, we look at the performance of the one-level RAS preconditioning for the four schemes. A comparison of the four schemes for the regular groups is shown in Fig.5.1, 5.2 and 5.3. The computing time and average numbers of iterations are reported in Table 5.4. For the bad group, we only show the numbers of iterations in Fig.5.4.

Scheme 1 is the most time-consuming approach, since all the 9200 systems are solved independently without any recycling. For Scheme 2, all the matrices in the sequence of linear systems share the same nonzero pattern, so we recycle the symbolic factorization of the subdomain matrices. At the same time, we also recycle the Krylov subspace constructed from the first system throughout all the other 9199 systems. For this test, we keep the harmonic Ritz vectors associated with the smallest $k = 36$ harmonic Ritz values in the first system for recycling. The numbers of iterations of Scheme 2 shown in Fig.5.1 are reduced drastically by more than 80% compared with Scheme 1, and the computing time is saved by about 30%.

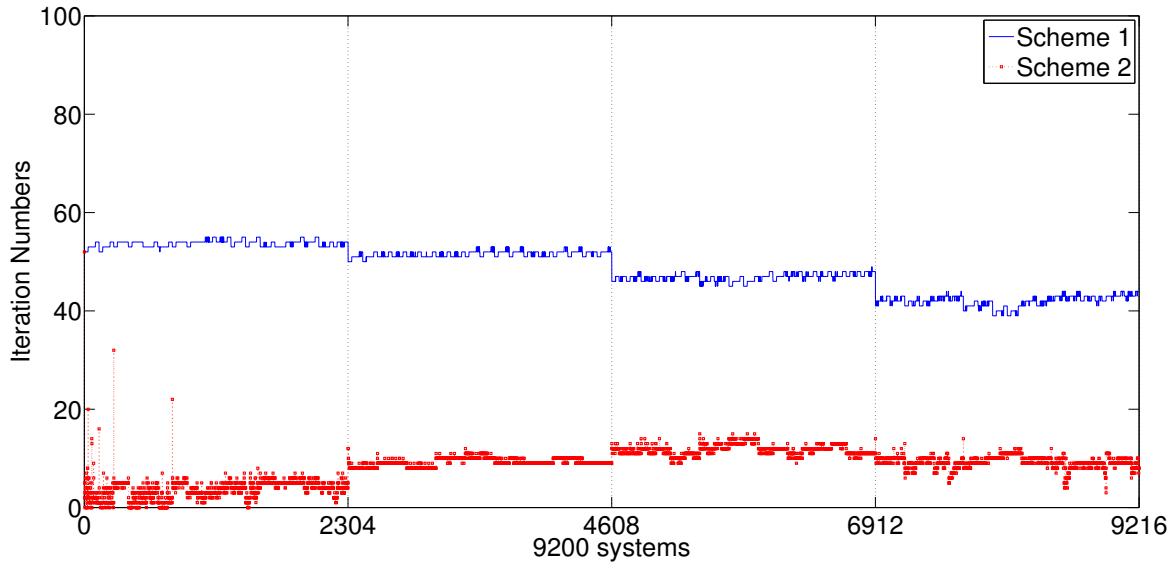

Figure 5.1: One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 2

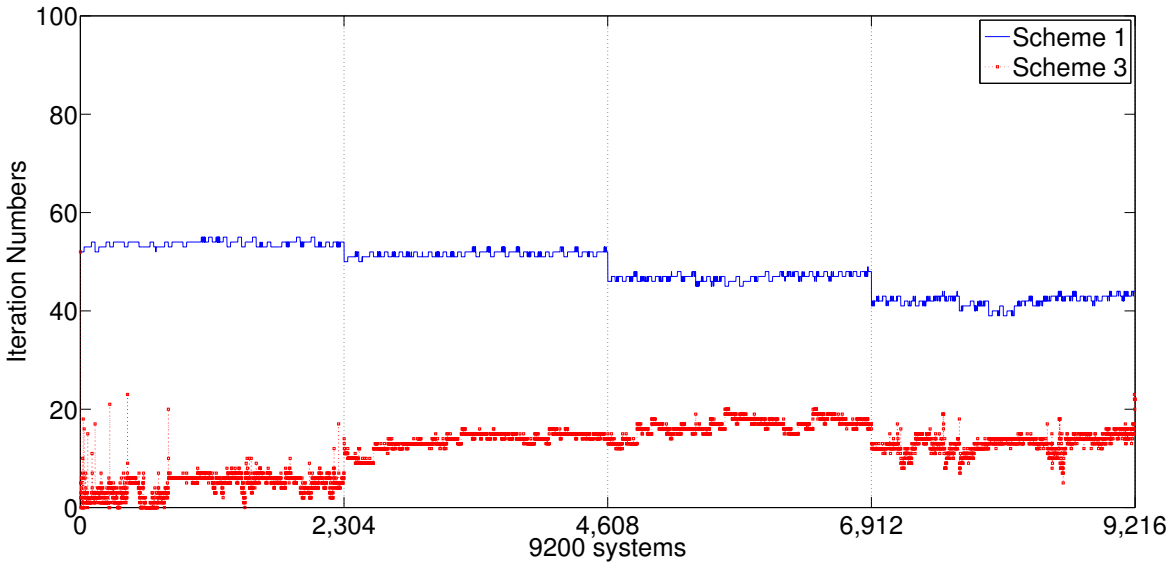

Figure 5.2: One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 3

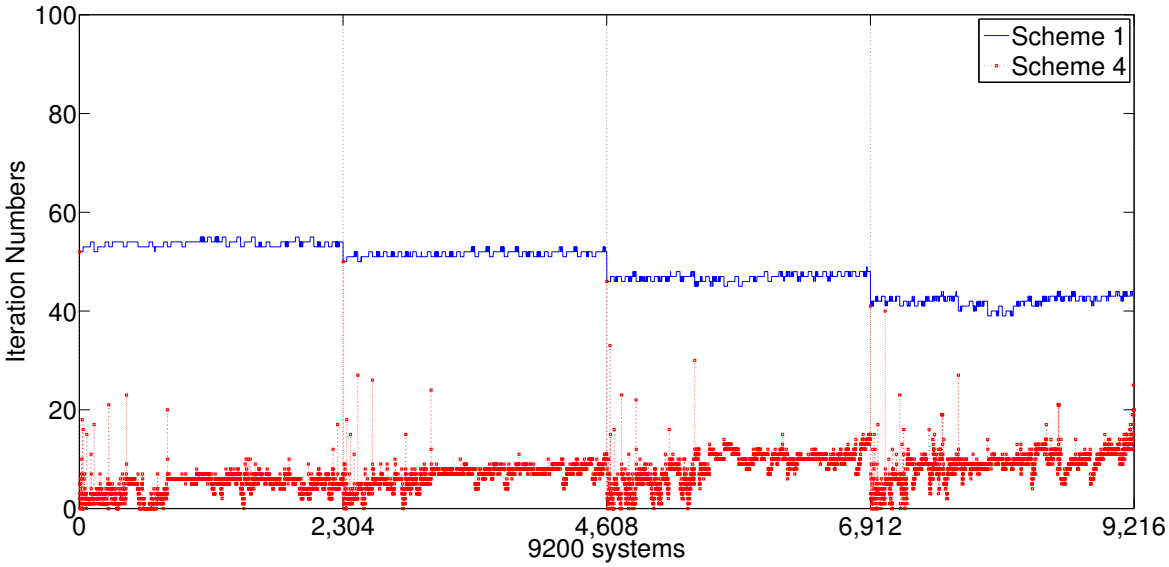

Figure 5.3: One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 4

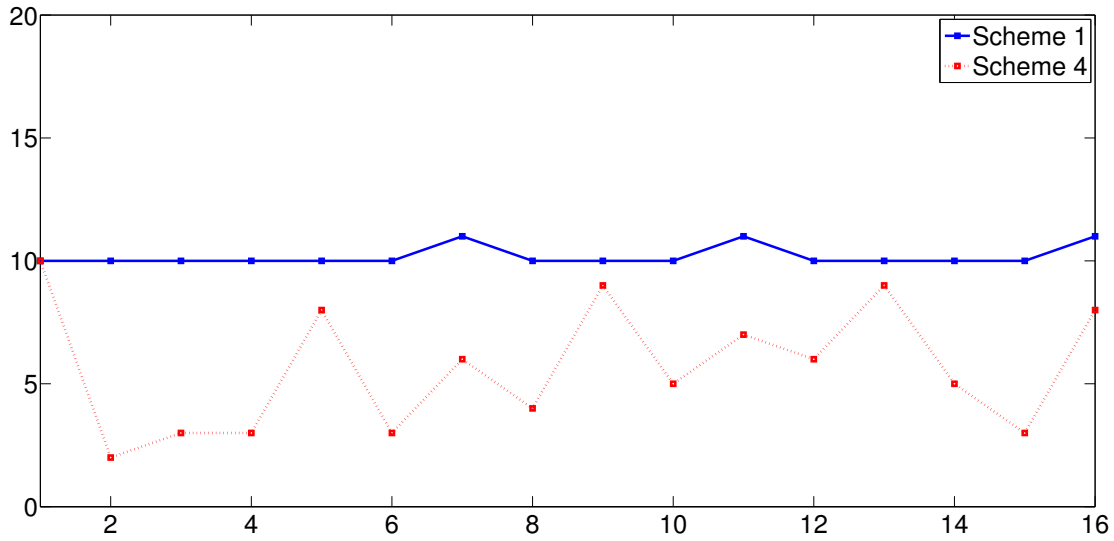

Figure 5.4: One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 4 in the bad group

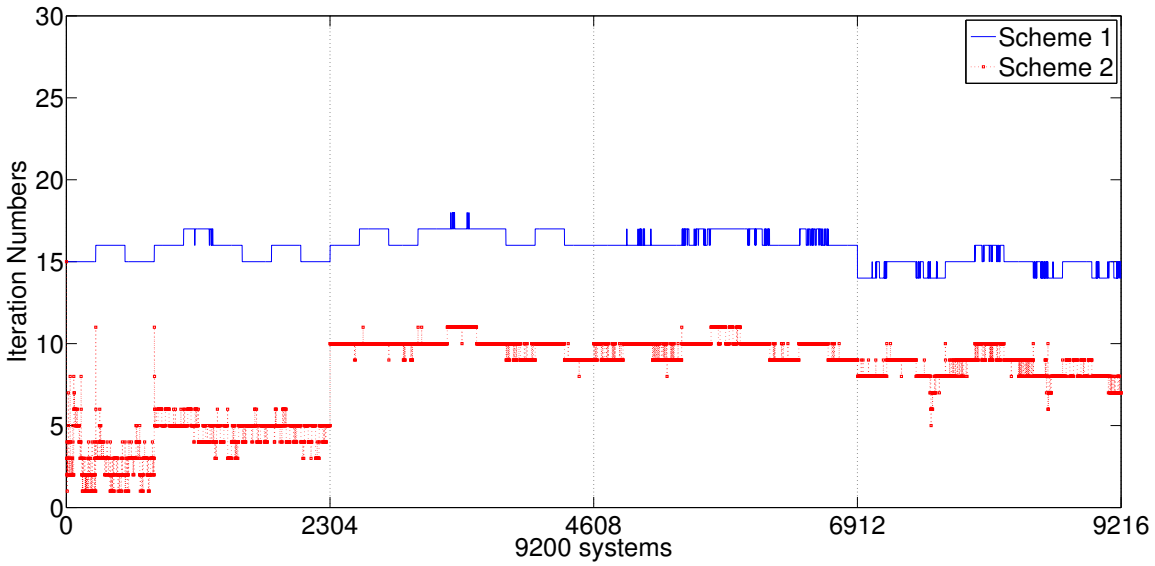

Figure 5.5: Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 2

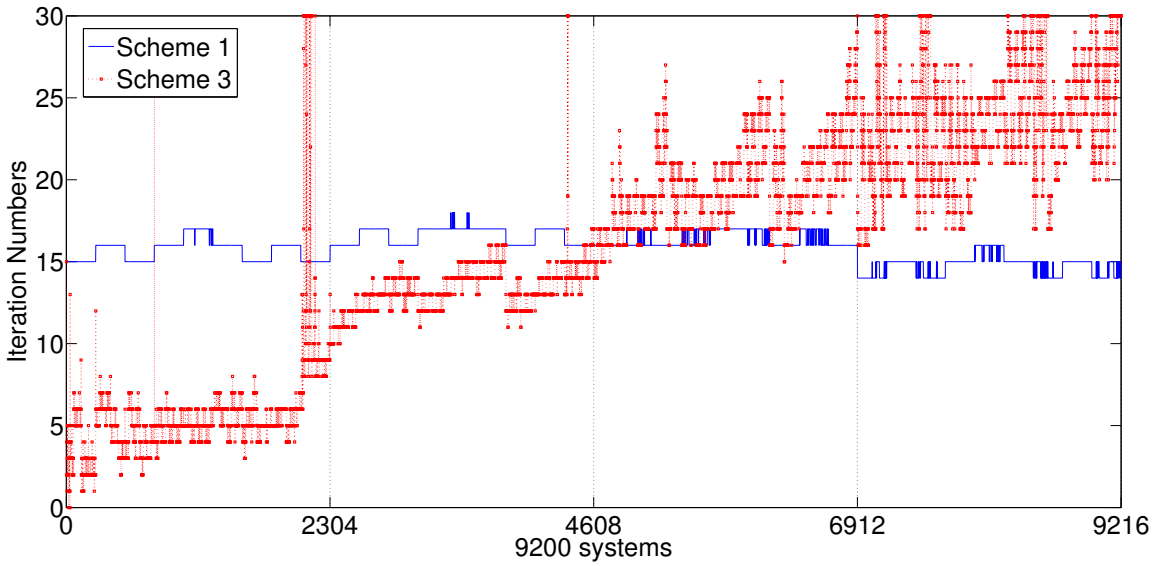

Figure 5.6: Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 3

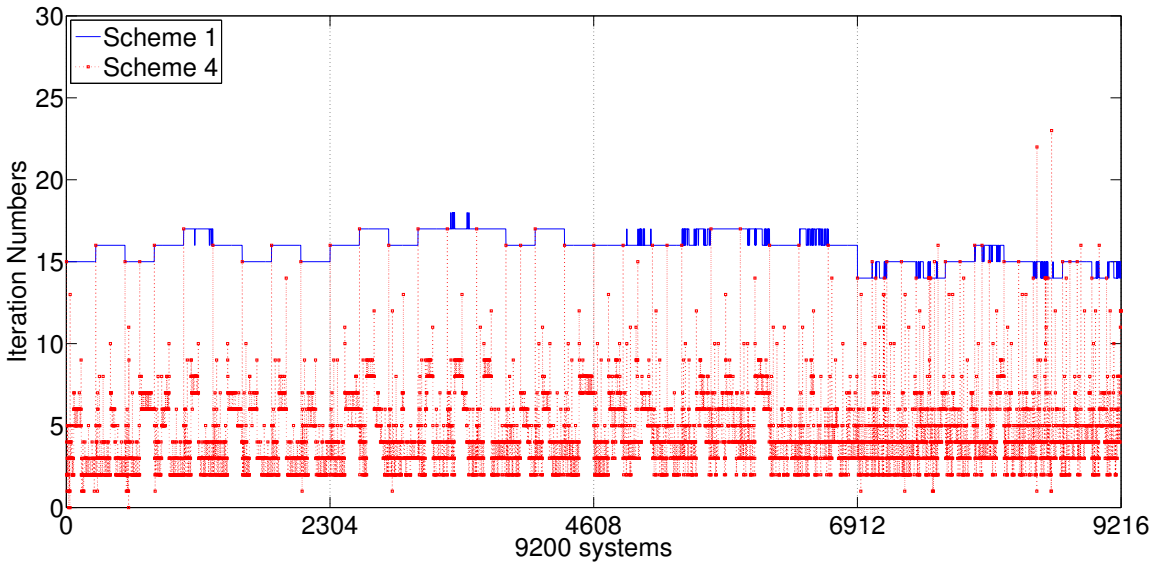

Figure 5.7: Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 4

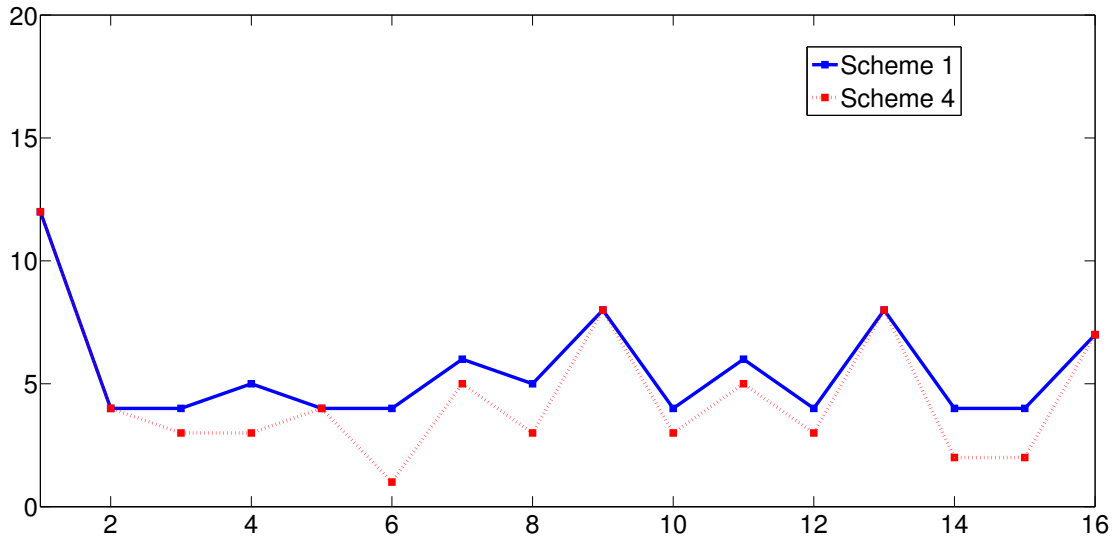

Figure 5.8: Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 4 in the bad group

Scheme 3 recycles both the Krylov subspace and the preconditioner obtained from the first system throughout the other 9199 systems, which means that we construct the Krylov subspace and the preconditioner only once. From Table 5.4, we can see that the numbers of iterations of Scheme 3 are slightly worse than that of Scheme 2 because of the recycled preconditioner, but the computing time is reduced by around 70% due to the time saved from recomputing the preconditioners.

Scheme 4 takes advantage of both Scheme 2 and Scheme 3. It reconstructs a new Krylov subspace and preconditioner for recycling before the cumulative perturbation grows too large, hence the average number of iterations of Scheme 4 is greatly reduced by the one-level grouping algorithm. Some of the systems require zero iteration, which means that the recycled Krylov subspace already contains the solutions to these systems. The computing time is the smallest among the four schemes.

Next, we analyze the performance of the two-level hybrid preconditioning. The parameters are chosen from the best timing results of the previous section. We present results of the four schemes in Fig.5.5, 5.6 and 5.7. The computing time and the average numbers of iterations for the two-level hybrid preconditioning are also recorded in Table 5.4. Notice that the computing time for Scheme 4 in Table 5.4 is slightly different from the optimal results in Table 5.1 and 5.3, even though they represent different runs of the same tests. This is because the network of the supercomputer is shared by all the users, which leads to the slight instability for the computing time. The comparison of four schemes with two-level hybrid preconditioner shows similar results to that with the one-level RAS preconditioner. Scheme 1 is the most expensive scheme. Scheme 2 (Fig.5.5) needs fewer number of iterations and smaller computing time by recycling Krylov subspace through all the systems. Scheme 3 (Fig.5.6) is worse than Scheme 2, since the cumulative perturbation at some point is too large, which yields a blow up at one point. Scheme 4 (Fig.5.7) does the recycling in each subgroup, so it has the best performance in terms of the numbers of iterations and the computing time. Fig.5.8 shows the numbers of iterations of the systems in the bad group.

Table 5.5: Average number of iterations for the two-level hybrid preconditioning with different mesh size, overlap size, and number of processors. The coarse mesh is 32×32.

mesh-window size	overlap	number of processors			
		128	256	512	1024
$249 \times 249 \times 8$	4	4.68	4.72	4.70	4.69
$373 \times 373 \times 16$	6	5.83	5.87	5.90	5.96
$497 \times 497 \times 32$	8	5.62	5.49	5.58	5.54

5.3 Scalabilities study of the two-level hybrid preconditioning

In this section, we test the dependency of the numbers of iterations on the mesh size, the number of processors, the overlap, the window size and the time step size using the grouping algorithm for the two-level hybrid preconditioning. The speedup is also presented to show its parallel scalability.

We first check how the average numbers of iterations behave with respect to the change of the mesh size, the overlap, and the number of processors. Table 5.5 shows that the average numbers of iterations are quite stable when the overlap is proportional to the diameter of subdomain, **i.e.**, the condition numbers are nearly independent of the mesh size and the number of processors.

Next, we check the performance of the algorithm with respect to the change of the window size. Table 5.6 shows that the computing time per window size is minimized when $s = 8$, then increases thereafter. The average numbers of iterations also have the same tendency as the computing time for each window size.

Table 5.7 shows some results with different Δt. We notice that the computing time and

Table 5.6: Computing time (second) per window size and average number of iterations (denoted by "aiter") for the two-level hybrid preconditioning with different window sizes.

497×497	window size				
	4	8	16	32	64
aiter	4.44	3.64	4.38	5.54	6.65
time/window size	79	64	80	126	167

Table 5.7: Computing time (second) and average numbers of iterations (denoted by "aiter") for the two-level hybrid preconditioning with different Δt.

Δt	0.0001	0.001	0.01	0.1
time	1307	1278	1556	1499
aiter	5.49	4.38	5.20	5.19

the average numbers of iterations do not change too much, even with large Δt. This shows the robustness of the algorithm.

We next present the speedup results obtained using two meshes in Fig.5.9. One has a fine mesh-window size $497 \times 497 \times 16$ and a coarse mesh-window size $32 \times 32 \times 16$, the other has the mesh-window size $497 \times 497 \times 32$ and a coarse mesh-window size $32 \times 32 \times 32$. From the left figure, we see that, for the smaller system, the speedup is close to be linear. For the larger system, the speedup is superlinear. We present the average numbers of iterations of two meshes in Fig.5.10, where the average number of iterations corresponding to the larger mesh is slightly more than that of the smaller mesh. The average number of iterations corresponding to the smaller mesh remains between 4 and 5 as the number of processors increases. Similarly, for the larger case, the average number of iterations increases slowly as the number of processors increases to 1024.

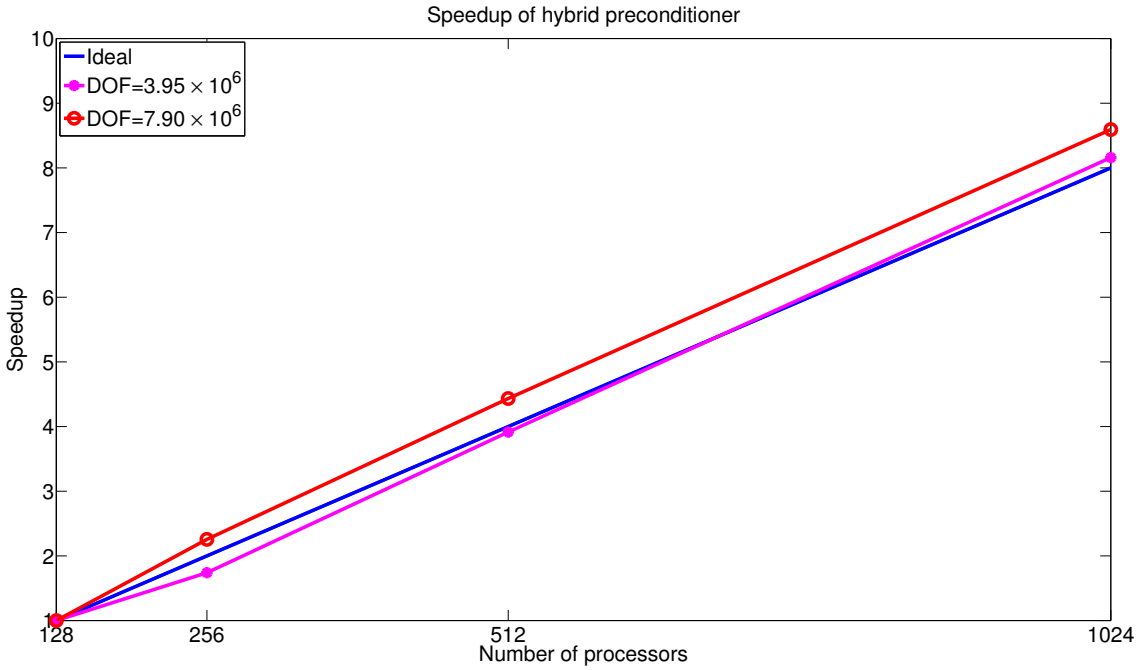

Figure 5.9: Speedup of two-level hybrid preconditioner

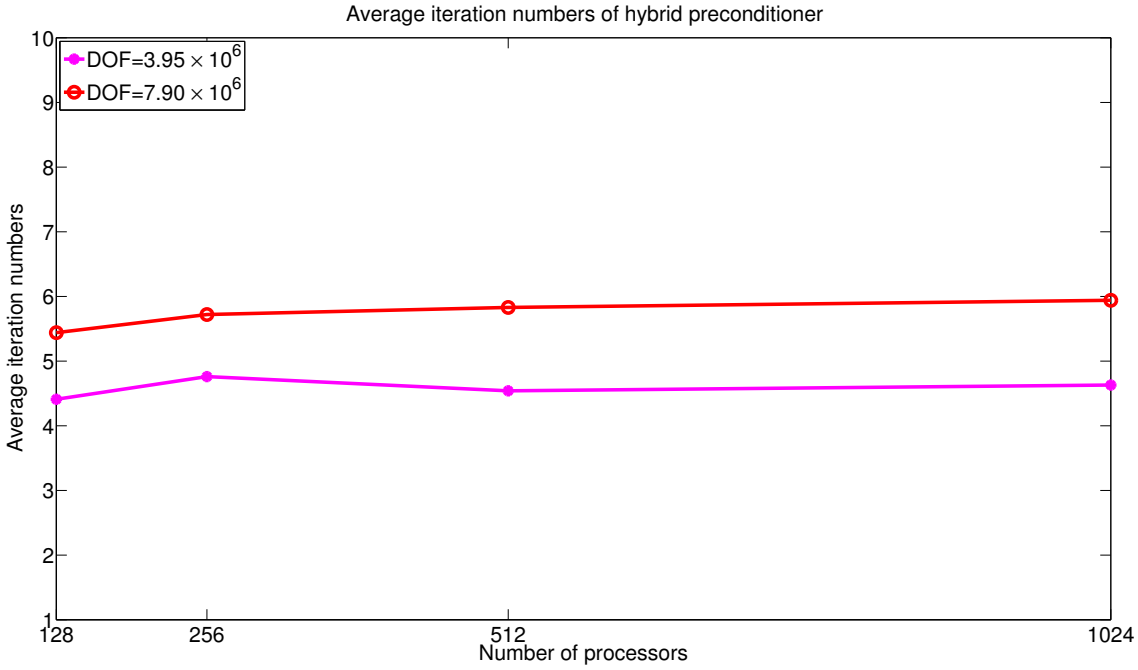

Figure 5.10: Average numbers of iterations of two-level hybrid preconditioner

Chapter 6

Conclusions and future work

In this chapter, we make some conclusions based on the research and experiments. We also try to outline some potential research directions on numerical simulation of stochastic differential equations.

In this thesis, we introduced and studied some implicit space-time domain decomposition preconditioned recycling Krylov subspace methods for stochastic parabolic differential equations. Using a stochastic Galerkin method, we decoupled the stochastic parabolic equation into a sequence of uncoupled deterministic parabolic equations. In order to accelerate the convergence of the preconditioned GMRES solver for the sequence of systems, an ordering algorithm and two grouping algorithms were proposed to take advantages of of the recycling Krylov subspace method and preconditioners. By using the grouping algorithms in some cases, the total computing time was reduced by almost 80%. Based on the experiments obtained on a supercomputer with over one thousand processors, we concluded that the two-level hybrid preconditioning technique with the corresponding grouping algorithm was the best choice in terms of the total computing time. In this thesis, we only considered domain decomposition methods, but multigrid methods [26, 27] may also work for the space-time discretized problems with some varieties of the proposed ordering and grouping algorithms.

We proved a theorem indicating that the window size cannot be too large, and in other words, the window size needs relatively small subdomain size compared with the spacial direction.

We believe that our approaches can be extended to solve the nonlinear system of stochas-

tic parabolic PDEs, such as Navier-Stokes equations in fluid dynamics, and also the hyperbolic equations. Increasing the number of levels in the preconditioner may also be necessary when the number of processors is much larger.

Bibliography

[1] O. Axelsson, Iterative Solution Methods, Cambridge University, New York, 1994.

[2] I. Babuska and P. Chatzipantelidis, On solving elliptic stochastic partial differential equations, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 4093-4122.

[3] I. Babuska, R. Tempone, and G. Zouraris, Galerkin finite element approximations of stochastic elliptic partial differential equations, SIAM J. Numer. Anal., 42 (2004), pp. 800-825.

[4] G. Bal and Y. Maday, A "parareal" time discretization for non-linear PDE's with application to the pricing of an American put, Recent developments in domain decomposition methods, Lect. Notes Comput. Sci. Engrg., Springer, Berlin, 23 (2001), pp. 189-202.

[5] S. Balay, K. Buschelman, W. D. Gropp, D, Kaushik, M. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, PETSc Users Manual, Argonne National Laboratory, 2013.

[6] B. Øksendal, Stochastic Differential Equations: An Introduction with Applications, Springer 2010.

[7] X.-C. Cai, Additive Schwarz algorithm for parabolic convection-diffusion equations, Numer. Math., 60 (1990), pp. 41-62.

[8] X.-C. Cai, Multiplicative Schwarz methods for parabolic problems, SIAM J. Sci. Comput., 15 (1994), pp. 587-603.

[9] X.-C. Cai, Some Domain Decomposition Algorithms for Nonselfadjoint Elliptic and Parabolic Partial Differential Equations, Ph.D. Thesis, Courant Institute, 1989.

[10] X.-C. Cai and M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput., 21 (1999), pp. 792-797.

[11] A. Chapman and Y. Saad, Deflated and argumented Krylov subspace techniques, Numer. Linear Algebra Appl., 4 (1997), pp. 43-66.

[12] Courant and Hilbert, Methods of Mathematical Physics, Interscience, New York, 1953.

[13] C. Farhat, L. Crivelli, and F. X. Roux, Extending substructure based on iterative solvers to multiple load and repeated analyses, Comput. Methods Appl. Mech. Engrg., 117 (1994), pp. 195-209.

[14] P. F. Fischer, Projection techniques for iterative solution of Ax=b with successive right-hand sides, Comput. Methods Appl. Mech. Engrg., 163 (1998), pp. 193-204.

[15] G. S. Fishman, Monte Carlo: Concepts, Algorithms, and Applications. Springer-Verlag, New York, 1996.

[16] P. Frauenfelder, C. Schwab, and R. A. Todor, Finite elements for elliptic problems with stochastic coefficients, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 205-228.

[17] M. J. Gander, L. Halpern, and F. Nataf, Optimal convergence for overlapping and non-overlapping Schwarz waveform relaxation, Eleventh International Conference on Domain Decomposition Methods, 1999.

[18] M. J. Gander and Petcu, Analysis of a Krylov subspace enhanced parareal algorithm for linear problems, Paris-Sud Working Group on Modeling and Scientific Computing 2007-2008 (E. Cances et al., eds.), ESAIM Proc., no. 25, EDP Sci., Les Ulis, 2008, pp. 114-129

[19] M. J. Gander and S. Vandewalle, Analysis of the parareal time-parallel time-integration method, SIAM J. Sci. Comput., 29 (2007), pp. 556-578

[20] R. G. Ghanem and P. D. Spanos, Stochastic Finite Elements: A Spectral Approach, Revised Edition, Dover, 2003.

[21] G. Golub and C. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[22] K. Guruprasad, D. E. Keyes, and J. H. Kane, GMRES for sequentially multiple nearby systems, Technical Report (1995), Old Dominium University.

[23] G. Horton, The time-parallel multigrid method, Comm. Appl. Numer. Methods, 8 (1992), pp. 585-595.

[24] G. Horton and S. Vandewalle, A space-time multigrid method for parabolic partial differential equations, SIAM J. Sci. Comput., 16 (1995), pp. 848-864.

[25] G. Horton, S. Vandewalle, and P. Worley, An algorithm with polylog parallel complexity for solving parabolic partial differential equations, SIAM J. Sci. Comput., 16 (1995), pp. 531-541.

[26] H. Elman and D. Furnival, Solving the stochastic steady-state diffusion problem using multigrid, IMA J. Number. Anal., 27 (2007), pp. 675-688.

[27] H. Elman, C. Miller, E. Phipps, and R. Tuminaro, Assessment of collocation and Galerkin approaches to linear diffusion equations with random data, Intel. J. Uncertainty Quantification, 1 (2011), pp. 19-33.

[28] C. Jin, Parallel Domain Decomposition Methods for Stochastic Partial Differential Equations and Analysis of Nonlinear Integral Equations Ph.D. Thesis, University of Colorado at Boulder, 2007.

[29] C. Jin and X.-C. Cai, A preconditioned recycling GMRES solver for stochastic Helmholtz problems, Commun. Comput. Phys., 6 (2009), pp. 342-353.

[30] C. Jin, X.-C. Cai, and C. Li, Parallel domain decomposition methods for stochastic elliptic equations, SIAM J. Sci. Comput., 29 (2007), pp. 2096-2114.

[31] E. Lelarasmee, A. Ruehli, and A. Sangiovanni-Vincentelli, The waveform relaxation method for time-domain analysis of large scale integrated circuits, IEEE Trans. Computer-Aided Design, 1 (1982), pp. 131-145.

[32] J.-L. Lions, Y. Maday, and G. Turinici, E*á*solution d'EDP par un sch*é*ma en temps "parar*é*al", C. R. Acad. Sci. Paris S*é*r. I Math. 332 (2001), pp. 661-668.

[33] M. Loève, Probability Theory, Vol. I, II, Springer, New York, 1978.

[34] Y. Maday and G. Turinici, A parareal in time procedure for the control of partial differential equations, C. R. Acad. Sci. Paris, S*é*r. I Math., 335 (2002), pp. 387-392.

[35] Y. Maday and G. Turinici, The parareal in time iterative solver: A further direction to parallel implementation, Proceedings of the 15th International Domain Decomposition Conference. Lect. Notes Comput. Sci. Engrg., Springer, Berlin, 40 (2005), pp. 441-448.

[36] R. B. Morgan, GMRES with deflated restarting, SIAM J. Sci. Comput., 24 (2002), pp. 20-37.

[37] F. Nobile and R. Tempone, Analysis and implementation issues for the numerical approximation of parabolic equations with random coefficients, Intl J. for Numer. Methods in Engrg., 80 (2009), pp. 979-1006.

[38] M. L. Parks, E. d. Sturler, G. Mackey, D. Johnson, and S. Maiti, Recycling Krylov subspaces for sequences of linear systems, SIAM J. Sci. Comput., 28 (2006), pp. 1651-1674.

[39] V. Simoncini and E. Gallopoulos, An iterative method for nonsysmmetric systems with multiple right-hand sides, SIAM J. Sci. Comput., 16 (1996), pp. 917-933.

[40] E. de Sturler, Nested Krylov methods based on GCR, J. Comput. Appl. Math., 67 (1996), pp. 15-41.

[41] M.-B. Tran, Parallel Schwarz waveform relaxation algorithm for an n-dimensional semilinear heat equation, arXiv preprint arXiv:1006.1323 (2010).

[42] S. Vandewalle, Parallel Multigrid Waveform Relaxation for Parabolic Problems, B.G. Teubner Verlag, Stuttgart, 1993.

[43] T. Weinzierl and T Koppl, A geometric space-time multigrid algorithm for the heat equation, Numer. Math. Theor. Meth. Appl., 5 (2012), pp. 110-130.

[44] J. White, A. Sangiovanni-Vincentelli, F. Odeh, and A. Ruehli, Waveform relaxation: theory and practice, Trans. Soc. Comput. Simul., 2 (1985), pp. 95-133.

[45] D. Xiu, Numerical Methods for Stochastic Computations-A Spectral Method Approach, Princeton University Press, 2010.

[46] D. Xiu and G. E. Karniadakis, A new stochastic approach to transient heat conduction modeling with uncertainty, Internat. J. Heat Mass Trans., 46 (2003), pp. 4681-4693.

[47] H. Zhu, C. Williams, R. Rohwer, and M. Morciniec, Gaussian Regression and Optimal Finite Dimensional Linear Models, Neural Networks and Machine Learning, Springer-Verlag, Berlin (1998).