# IMPLICIT SPACE-TIME DOMAIN DECOMPOSITION METHODS FOR STOCHASTIC PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS*

CUI CONG†, XIAO-CHUAN CAI‡, AND KARL GUSTAFSON§

**Abstract.** 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. The parallelism is restricted to each time step, and the algorithms are purely sequential in time. In this paper, we develop some overlapping Schwarz methods whose subdomains cover both space and time variables, and we show that the methods work well for stochastic parabolic equations discretized with an implicit stochastic Galerkin method. One- and two-level Schwarz preconditioned recycling GMRES methods are carefully investigated such that many components of the methods are reused 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.

**Key words.** Stochastic parabolic equations, implicit space-time domain decomposition methods, recycling Krylov subspace method, parallel computing

**AMS subject classifications.** 35R60, 60H15, 60H35, 65C30, 47B80, 65N55, 65M55, 65M60

## 1. Introduction.

In this paper, we 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 random function $a = a(x,\omega)$. In the latter case, the parabolic equation is stochastic [33, 41]. 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.

*This research is supported in part by NSF grants DMS-0913089 and CCF-1216314.

†Department of Mathematics, University of Colorado Boulder, Boulder, CO 80309 (cui.cong@colorado.edu)

‡Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309 (cai@cs.colorado.edu)

§Department of Mathematics, University of Colorado Boulder, Boulder, CO 80309 (karl.gustafson@colorado.edu)

Waveform relaxation [27, 40] 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 [22, 38] or Schwarz waveform relaxation method [15, 37]. The space-time multigrid method for parabolic PDEs [20, 21, 39] 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 [28] is an iterative method to solve evolution problems in a time-parallel manner. More applications of this method are reported in [4, 30, 31] and references therein. The advantage of this algorithm is that it approximates the solutions later in time before accurately approximating the solutions at earlier times [17]. This algorithm has received great attention since it was proposed, and several variants are presented in different frameworks, for example, PITA (parallel implicit time integrator) [11], space-time multigrid and multiple shooting method [17].

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

$$\left\{\begin{array}{lcl} u_h^1 + \Delta t_1 L_h(u_h^1) - u_h^0 & = & \Delta t_1 f^1 \\ u_h^2 + \Delta t_2 L_h(u_h^2) - u_h^1 & = & \Delta t_2 f^2 \\ & \vdots & \\ u_h^s + \Delta t_s L_h(u_h^s) - u_h^{s-1} & = & \Delta t_s f^s. \end{array}\right. \tag{1.2}$$

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, therefore, we usually select $s$ to be much smaller than $n$. For the rest of the paper, $s$ will be referred to as the window size. The algorithm needs to be used multiple times in order to cover all $n$ time steps.

One of the main emphases of the paper is to develop an overlapping Schwarz preconditioned recycling Krylov subspace method for solving (1.2). The subproblems are obtained by a partition of $D \times [t^1, t^s]$. The classical Schwarz theory [6, 7, 8] doesn't work for this coupled space-time problem, but our numerical experiments show that both the one- and two-level algorithms which we will introduce later work well. The second focus of the paper is to consider the case when the diffusion coefficient is a random function. Using a stochastic Galerkin approach [18, 25, 26, 41], we convert the stochastic parabolic problem into a large number of deterministic equations that are similar to (1.2). 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.

We structure the rest of the paper as follows. Section 2 describes some space-time domain decomposition methods for a deterministic parabolic problem. In Section 3, we describe the stochastic Galerkin method for a parabolic equation including the weak formulation, the Karhunen-Loève expansion, and the double orthogonal basis. Based on the method, we show how to decouple the stochastic parabolic equation into a sequence of deterministic parabolic equations. In Section 4, a recycling Krylov subspace method is briefly reviewed. Then an ordering algorithm and two grouping algorithms are proposed to maximize the benefit of the recycling Krylov subspace. Finally, experimental results and remarks are reported in Section 5 and 6, respectively.

**2. Space-time domain decomposition methods.** We consider the numerical solution of a deterministic parabolic equation

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

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

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

Let $0 = t^0 < t^1 < t^2 < \cdots < t^n = T$ 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 $s(< n)$ 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 2.1. *The condition number of $A_s$ is nondecreasing as $s$ increases.*

Theorem 2.1 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 $A_{s+1}$. We omit the details of the proof in this paper.

According to Theorem 2.1, we realize that the window size cannot be as large as we wish, 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.

In order to formally define the Schwarz preconditioners in the space-time formulation, we begin with a fine mesh $\mathcal{D}_h$ on the spacial domain $D$. We first decompose $D$ into non-overlapping subregions $D_k, k = 1, 2, \cdots, N$. To obtain an overlapping decomposition of the domain, we extend each subregion $D_k$ to $D_k^\delta$ by including extra $\delta$ mesh layers from adjacent subregions. For the space-time formulation, we define a fine mesh $\mathcal{D}_h^s$ 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 $n$ fine mesh points inside $D \times [t^{l+1},\ t^{l+s}]$ and $n_k$ fine mesh points inside $D_k^\delta \times [t^{l+1},\ t^{l+s}]$, then the elements of a matrix $R_k^\delta$ of size $n_k \times n$ are defined as follows:

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

The matrix $R_k^\delta$ serves as a restriction operator by getting rid of all the components that are outside of $D_k^\delta \times [t^{l+1}\ t^{l+s}]$. If we transpose the restriction matrix, the resulting matrix $(R_k^\delta)^T$ works as an extension operator, which refills the components that do not belong to $D_k^\delta \times [t^{l+1}\ t^{l+s}]$ by 0. The one-level restricted additive Schwarz (RAS) preconditioner [9] for the matrix $A_s$ 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 $A_k = R_k^\delta A_s (R_k^\delta)^T$. The matrix vector multiplication involving $A_k^{-1}$ is usually computed by $LU$ or $ILU$ factorization.

For the 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 fine mesh to the coarse mesh, an interpolation operator $I_H^h$ from the coarse mesh to the fine mesh and a coarse matrix $A_0 = I_h^H A_s I_H^h$ on the coarse mesh. The two-level additive Schwarz preconditioner is

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

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 clearly, the two-level hybrid preconditioner is implemented according to

$$\begin{aligned} z &:= M_1^{-1} x, \\ z &:= z + M_C^{-1}(x - A_s z), \\ z &:= z + M_1^{-1}(x - A_s z), \end{aligned}$$

where $M_C^{-1} = I_H^h A_0^{-1} I_h^H$ is the coarse preconditioner constructed as the first term of $M_2^{-1}$. 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. Note that since the subdomains are space-time subdomains, "zero Dirichlet" implies that zero values are used in both the space and time boundaries of the subdomains. Several inexact additive Schwarz preconditioners are available. In this paper, we employ the two-level hybrid preconditioner with an incomplete factorization for the subdomain matrices in the implementation. We would like to point out that for the one-level and two-level algorithms just described, we do not have any computational complexity estimates, however, for the classical Schwarz methods, the estimates are available in [6, 7].

**3. Stochastic Galerkin method for parabolic problems.** In this section, 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, 14, 18]. We begin with an introduction of some notations. Let $\mathcal{A}$ be a $\sigma$-algebra defined on a given sample set $\Omega$ with sample points $\omega$. $P$ is a probability measure on $\mathcal{A}$. The triple $(\Omega, \mathcal{A}, P)$ forms a probability space. We define a real-valued random function $\mathcal{F}$ on the jointly measurable spacial domain $D$ and sample space $\Omega$

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

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 $\omega \in \Omega$.

We consider the following stochastic parabolic PDE:

$$(3.1)\quad \begin{cases} \dfrac{\partial u(x,t,\omega)}{\partial t} - \nabla \cdot (a(x,\omega)\nabla u(x,t,\omega)) &= f(x) \quad \text{on} \quad D \times (0,T] \times \Omega \\ u(x,0,\omega) &= u^0(x) \quad \text{on} \quad D \\ u(x,t,\omega) &= 0 \quad \text{on} \quad \partial D \times [0,T]. \end{cases}$$

We assume the coefficient $a(x,\omega)$ is bounded and strictly positive. 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

$$(3.2)\qquad \left\langle \int_D \frac{\partial u}{\partial t} v dx \right\rangle + \left\langle \int_D a(x,\omega)\nabla u(x,\omega) \cdot \nabla v dx \right\rangle = \left\langle \int_D f(x) v dx \right\rangle$$

for any $v \in H_0^1(D \times \Omega)$. Here $\langle \cdot \rangle$ denotes the mean (or expected value). The mean of the coefficient function $a(x, \omega)$ is defined as

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

In (3.2), the scalar product is the standard inner product in $L^2(D)$, i.e.,

$$\int_D f(x) v dx = (f(x), v)\,.$$

We use the Karhunen-Loève (KL) expansion [29] to represent the stochastic coefficient, and assume that the mean and covariance of $a(x, \omega)$ are given. According to the KL expansion, $a(x, \omega)$ can be expressed by a series expansion:

$$a(x, \omega) = a_0(x) + \sum_{i=1}^{\infty} \sqrt{\lambda_i} k_i(x) \xi_i(\omega), \tag{3.3}$$

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, $\lambda_n$ and $k_n(x)$ are eigenpairs of the covariance function $C_a(x_1, x_2)$, $x_1, x_2 \in D$. And positive eigenvalues $\{\lambda_n\}_{n=0}^{\infty}$ are ordered non-increasingly, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq \cdots > 0$.

In practice, we use a finite number of terms to approximate the series expansion of $a(x, \omega)$,

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

where $M$ depends on the decay of eigenvalues.

Let $\rho_i$ 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 $\rho = (\rho_1, \rho_2, \cdots, \rho_M)$. Suppose $\Gamma_i$ is the image of $\xi_i \in R$, thus $\Gamma = (\Gamma_1, \Gamma_2, \cdots, \Gamma_M) \subseteq R^M$ is the image of $\xi$, *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 $a_M(x, \xi)$

$$a_M(x, \xi) = a_0(x) + \sum_{i=1}^{M} \sqrt{\lambda_i} k_i(x) \xi_i. \tag{3.4}$$

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,

$$\frac{\partial u_M(x, t, \xi)}{\partial t} - \nabla \cdot (a_M(x, \xi) \nabla u_M(x, t, \xi)) = f(x). \tag{3.5}$$

A double orthogonal basis [18] is introduced in order to decouple the equation (3.5) 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:

$$\left\{ \begin{array}{rcll} \displaystyle\int_{\Gamma_j} \phi_{p,j}(z)\phi_{q,j}(z)\rho_j(z)dz & = & \delta_{p,q} & p, q = 0, 1, \cdots n_j, \\ \displaystyle\int_{\Gamma_j} z\phi_{p,j}(z)\phi_{q,j}(z)\rho_j(z)dz & = & C_{p,j}\delta_{p,q} & p, q = 0, 1, \cdots n_j, \end{array} \right. \tag{3.6}$$

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

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

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

$$\left\{ \phi_{\mathbf{i}}(\xi) \mid \phi_{\mathbf{i}}(\xi) = \prod_{j=1}^{M} \phi_{i_j,j}(\xi_j), i_j \in \{0, 1, \cdots, n_j\}, \mathbf{i} \in \mathbf{I} \right\}. \tag{3.7}$$

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

$$\left\langle \int_D \frac{\partial u}{\partial t} v dx \right\rangle + \left\langle \int_D a_M \nabla u \cdot \nabla v dx \right\rangle = \left\langle \int_D f(x) v dx \right\rangle, \tag{3.8}$$

for any $v = h(x)\phi_{\mathbf{j}}(\xi) \in H_0^1(D) \times P_{\mathbf{n}}(\Gamma)$, $\mathbf{j} \in \mathbf{I}$. $u_{M,i}$ is the coefficient for the basis function $\phi_i(\xi)$. The second term of the left-hand side in equation (3.8) 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

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

And the first term is

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

The right-hand side of equation (3.8) is

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

We summarize the results in the following theorem.

THEOREM 3.1. *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* (3.8)*, 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*

$$\frac{\partial u_{M,\mathbf{i}}(x,t)}{\partial t} - \nabla \cdot (a_{M,\mathbf{i}}(x) \nabla u_{M,\mathbf{i}}(x,t)) = f_{\mathbf{i}}(x), \tag{3.9}$$

*where*

$$\begin{cases} a_{M,\mathbf{i}}(x) & := & a_0(x) + \sum_{j=1}^{M} \sqrt{\lambda_j} k_j(x) C_{i_j,j} \\ f_{\mathbf{i}}(x) & := & f(x) \cdot \prod_{j=1}^{M} \int_{\Gamma_j} \phi_{i_j,j}(z) \rho_j(z) dz \\ \mathbf{i} & \in & \mathbf{I} = \{\{i_1, i_2, \cdots, i_M\} \mid i_j \le n_j, \text{for} j = 1, 2, \cdots, n_M\}. \end{cases} \tag{3.10}$$

*Taking the sum of (3.9) for all $\mathbf{i} \in \mathbf{I}$, we obtain the solution to the equation* (3.8) *(or* (3.5)*) is*

$$u_M(x,t,\xi) = \sum_{\mathbf{i}\in\mathbf{I}} u_{M,\mathbf{i}}(x,t) \phi_{\mathbf{i}}(\xi). \tag{3.11}$$

TABLE 4.1

*Reordering $C_{i_j,j}$ decreasingly by a certain permutation $\{0,1,\cdots,n_j\} \longrightarrow \{l_0^j, l_1^j, \cdots, l_{n_j}^j\}$, for all $j$, $0 \le j \le M$.*

$$\begin{array}{lccccccccc}
\lambda_1\colon & C_{0,1} & C_{1,1} & \cdots & C_{n_1,1} & \longrightarrow & C_{l_0^1,1} & C_{l_1^1,1} & \cdots & C_{l_{n_1}^1,1} \\
\lambda_2\colon & C_{0,2} & C_{1,2} & \cdots & C_{n_2,2} & \longrightarrow & C_{l_0^2,2} & C_{l_1^2,2} & \cdots & C_{l_{n_2}^2,2} \\
\vdots & & \vdots & & & \vdots & & \vdots & & \\
\lambda_M\colon & C_{0,M} & C_{1,M} & \cdots & C_{n_M,M} & \longrightarrow & C_{l_0^M,M} & C_{l_1^M,M} & \cdots & C_{l_{n_M}^M,M}
\end{array}$$

**4. Recycling Krylov subspace method and grouping algorithm.** Based on the above Theorem 3.1, 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. Following the idea of [26], 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.

There are several choices of recycling Krylov subspace algorithms [10, 12, 13, 19, 32, 35, 36]. We choose to use the so-called GCROT version of recycling GMRES introduced in [34] since both the matrix and the right-hand side change in the sequence of systems. When solving a single linear system of equations, GCROT is a truncated minimum residual method which retains a subspace between cycles in a way that minimizes the loss of orthogonality with respect to the vectors that are not retained. 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.

**4.1. Ordering algorithm.** In order to maximize the benefit of the recycling strategy, we arrange the systems following a decreasing order of the perturbation. This is similar to the approach of [26]. In this way, we can determine when to restart with a new Krylov subspace and new preconditioner when the cumulative perturbation has grown too large using the current Krylov subspace and preconditioner. Theorem 3.1 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

$$\sum_{j=1}^{M} \sqrt{\lambda_j} k_j(x) C_{i_j,j} \tag{4.1}$$

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 (4.1) 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 $\mathbf{i} = \{i_1, i_2, \cdots, i_M\} \in \mathbf{I}$, $0 \le i_j \le n_j$, $1 \le j \le M$. We reorder $\{0, 1, \cdots, n_j\}$ 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 $\lambda_j$, for all $j = 1, 2, \cdots, M$. We call this a "pseudo-decreasing" ordering because (4.1) 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

**4.2. 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 impacts 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 $\lambda_1$. 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 2.1, 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 $\gamma(\ge 0)$ 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.,

$$\min_{x\in\mathcal{V}_C^H}\{a_{M,\mathrm{i}}(x)\} < \gamma. \tag{4.2}$$

When the inequality (4.2) 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 (4.2), 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 $\gamma$ depends on the specific problem. For some problems with $0 < a_1 \leq a(x,\omega) \leq a_2 < \infty$, where $a_1$ is relatively far from zero, we don't need to set up such a cutoff parameter $\gamma$, since there is no sensitive system in this case.

**4.3. 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

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

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 being 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 contained in the last group $G_{n_1}$, which have to be extracted from some of the subgroups in $G_{n_1}$ by the cutoff parameter $\gamma(\geq 0)$. All the sensitive systems satisfying (4.2) 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 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 the 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 $\lambda_p > \overline{\lambda} > \lambda_{p+1}, 2 \leq p \leq M$.
- 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 some of the subgroups in $G_{n_1}$ that satisfy (4.2), 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.

**4.4. A numerical example.** We illustrate our grouping algorithms by the following example [26] with the mean and covariance functions:

$$a_0(x) = 3 + sin(\pi x_1) \quad C_a(x,x') = e^{-|x-x'|^2}, \quad x \in [0,1]^2. \tag{4.3}$$

The series expansion of $a(x,\xi)$ is truncated at $M = 11$ by the decay of the eigenvalues. The selection algorithm proposed in [14] 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 $\xi_j$ in the KL expansion be uniformly distributed in $\Gamma_j = [-\sqrt{3},\sqrt{3}]$. 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.

We calculate the largest eigenvalue $\lambda_1 = 0.7480$, which 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 $C_{0,1} = 1.49, C_{1,1} =$

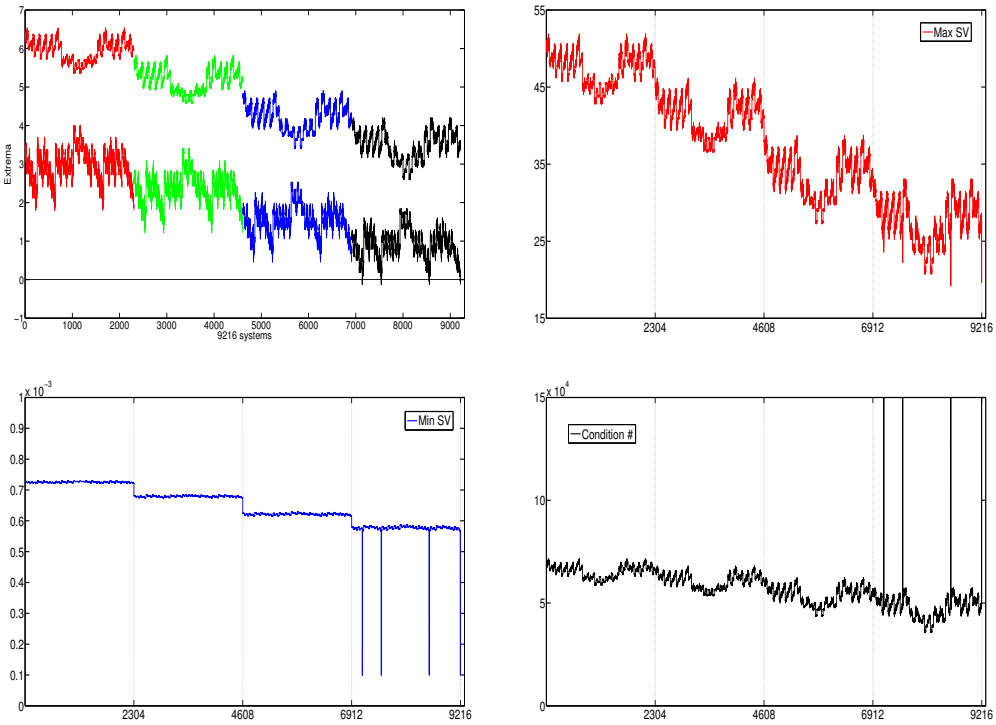


FIG. 4.1. *The top left figure shows the maxima and minima of the diffusion coefficients* $a_{M,\mathbf{i}}(x)$ *for all the* 9216 *systems. The top right and bottom left figures show the maxima and minima singular values of all the* 9216 *systems without any preconditioning. The bottom right figure shows the condition numbers of all the* 9216 *systems without any preconditioning.*

$0.59, C_{2,1} = -1.49, C_{3,1} = -0.59$. By our ordering algorithm and grouping algorithm for 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. The top left figure of Fig.4.1 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 the top left figure of Fig.4.1 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 the top left figure of Fig.4.1, 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

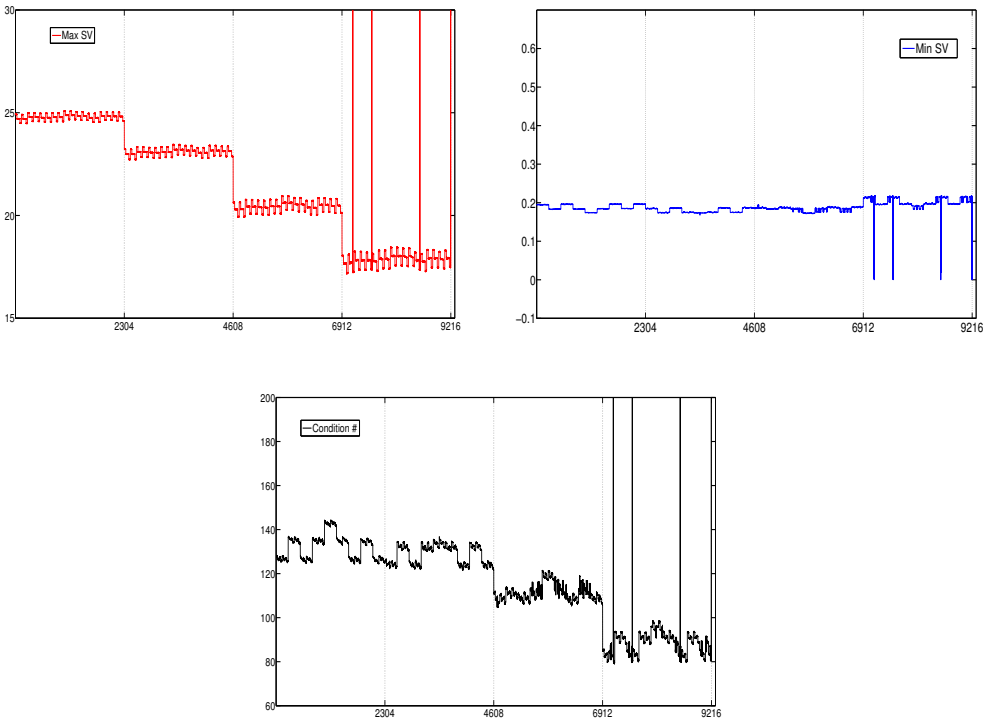


FIG. 4.2. *The top two figures show the maxima and minima singular values of all the* 9216 *systems with the two-level hybrid preconditioner, the bottom figure shows the condition numbers of all the* 9216 *systems with the two-level hybrid preconditioner.*

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 mention that the choice of the window size $s$ is made experimentally. In practice, the optimal value of $s$ depends on the problem, as well as the available computing resources.

We also plot the extrema singular values and condition numbers for the 9216 systems without preconditioning in Fig.4.1. All the systems are coupled by 16 time steps for the regular groups, but only one time step for the bad group. The top right figure of Fig.4.1 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 the top left figure of Fig.4.1. In the bottom two figures of Fig.4.1, 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 the top left figure of Fig.4.1 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.4.2. After applying the preconditioner on the systems, the maximum singular values are decreased by about 50% and the minimum singular values are increased 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 the top two figures of Fig.4.2, we coupled 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.

**5. Numerical experiments.** In this section, we report some numerical experiments to illustrate the performance of the Schwarz preconditioned recycling FGMRES of [34] 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 (3.1) 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 (4.3), 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 stop 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 $497 \times 497$ 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 and the average number of iterations for various dimensions of the recycling subspaces. We mark the best computing time in red. 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

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. $k_m$ is defined in (5.1). $np = 1024$, 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 in red. "total" represents the total dimension, i.e., the summation of "best" and "aiter" corresponding to the minimal computing time.*

| | 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 |

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 preconditioning in Table 5.3. The coarse mesh is $32 \times 32$ and the fine mesh is $497 \times 497$. 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

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 |

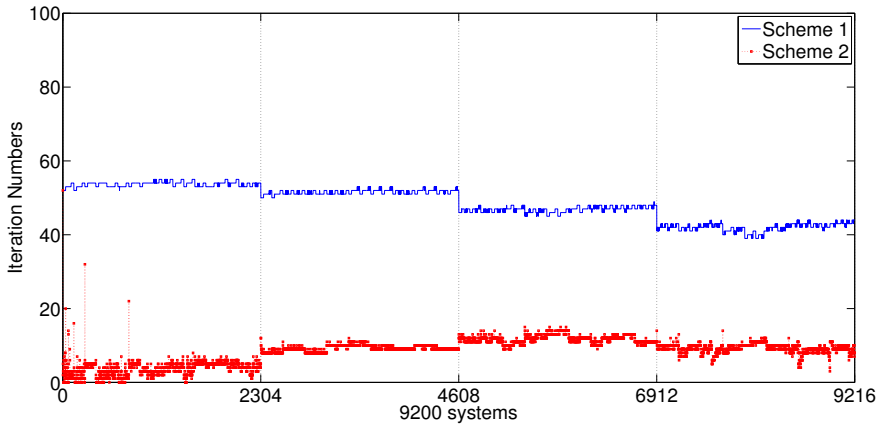


FIG. 5.1. *One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 2*

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 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.

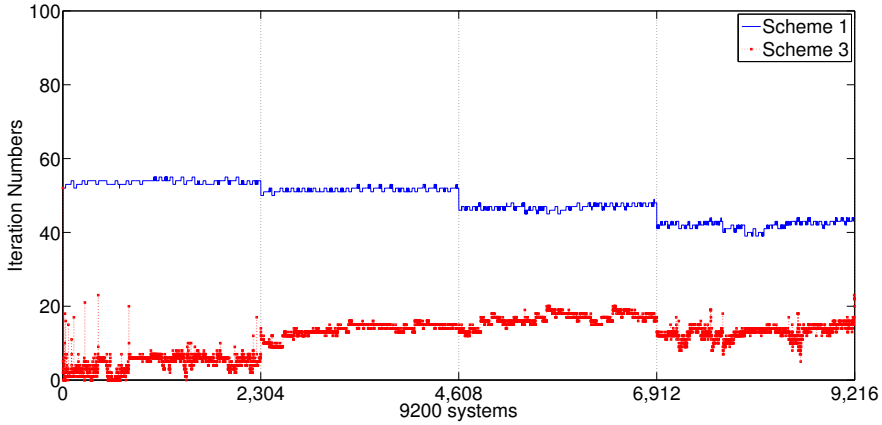


FIG. 5.2. *One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 3*

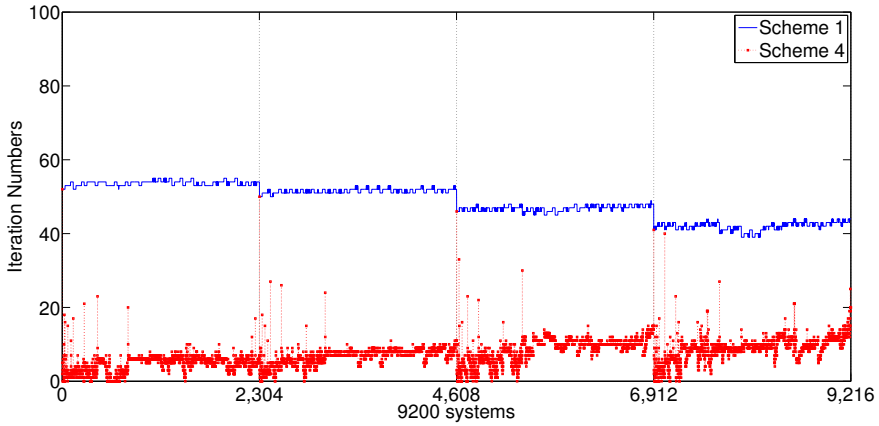


FIG. 5.3. *One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 4*

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%.

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.

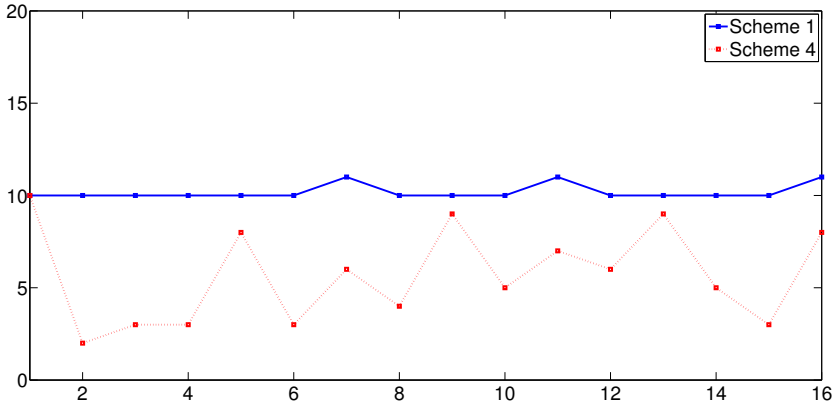


FIG. 5.4. *One-level RAS preconditioning, comparison of numbers of iterations for Scheme 1 and 4 in the bad group*

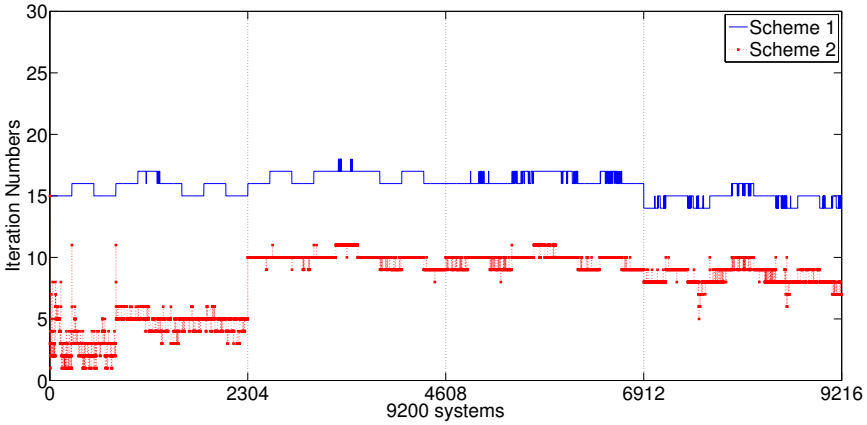


FIG. 5.5. *Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 2*

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

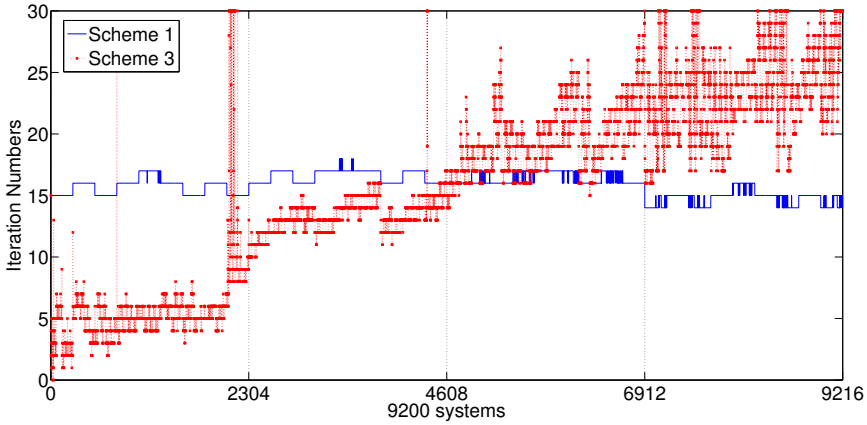


FIG. 5.6. *Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 3*

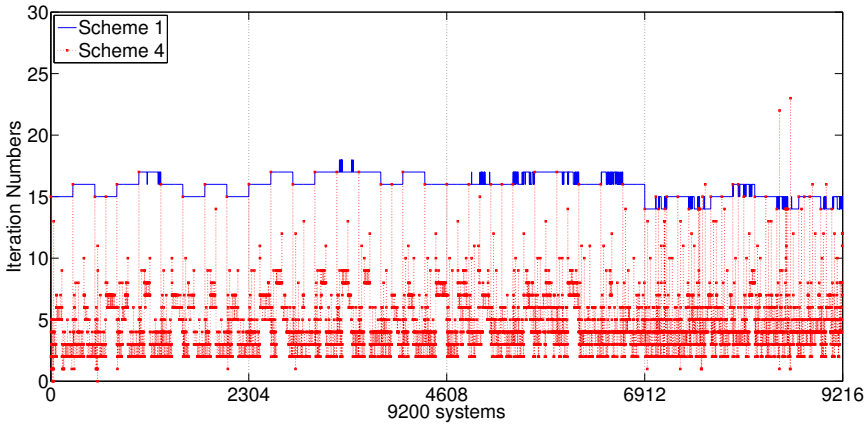


FIG. 5.7. *Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 4*

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.

**5.3. Scalabilities of the two-level hybrid preconditioning.** In this section, we test the dependency of the numbers of iterations on the mesh size, the number

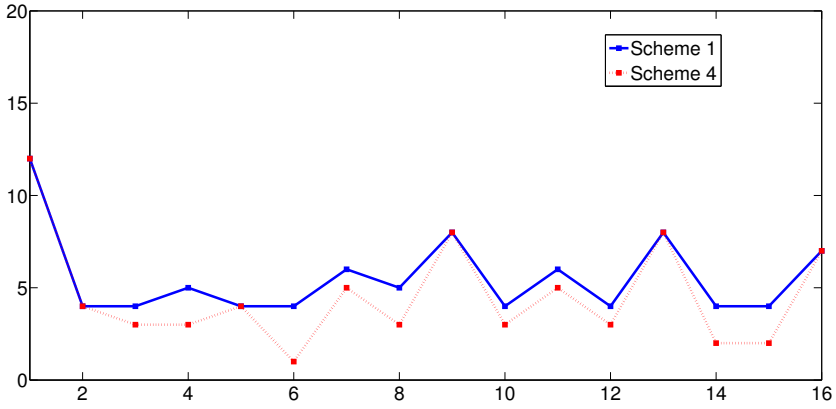


FIG. 5.8. *Two-level hybrid preconditioning, comparison of numbers of iterations for Scheme 1 and 4 in the bad group*

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 |

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 $\Delta t$. We notice that the computing time and the average numbers of iterations do not change too much, even with large $\Delta t$. 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 the right figure of Fig.5.9, 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.

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* $32 \times 32$.

| 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 |

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.*

| $497 \times 497$ | 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 |

Similarly, for the larger case, the average number of iterations increases very slowly as the number of processors increases up to 1024.

**6. Conclusion.** In this paper, 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 with doubly orthogonal basis, we decouple 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 are proposed to maximize the benefit of the recycling Krylov subspace and preconditioners. By using the grouping algorithms, the total computing time can be reduced by almost 80%. Based on the experiments obtained on a supercomputer with over one thousand processors, we conclude that the two-level hybrid preconditioning technique with the corresponding grouping algorithm is the best choice in terms of the total computing time. In this paper, we only considered domain decomposition methods, but multigrid methods [23, 24] may also work for the space-time discretized problems with the proposed ordering and grouping algorithms.

**Acknowledgments.** The authors would like to thank Dr. Chao Jin, Professor Alireza Doostan, and the referees for many helpful discussions and suggestions.

REFERENCES

[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, Vol. 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, 2012.

TABLE 5.7

*Computing time (second) and average numbers of iterations (denoted by "aiter") for the two-level hybrid preconditioning with different $\Delta t$.*

| $\Delta$t | 0.0001 | 0.001 | 0.01 | 0.1 |
|---|---|---|---|---|
| time | 1307 | 1278 | 1556 | 1499 |
| aiter | 5.49 | 4.38 | 5.20 | 5.19 |

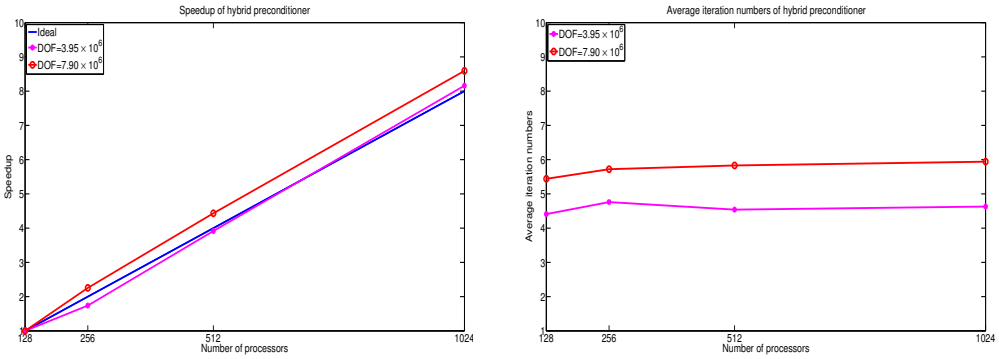


FIG. 5.9. *Speedup and average numbers of iterations of two-level hybrid preconditioner*


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

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

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

[9] 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.

[10] A. CHAPMAN AND Y. SAAD, *Deflated and argumented Krylov subspace techniques*, Numer. Linear Algebra Appl., 4 (1997), pp. 43-66.

[11] C. FARHAT AND M. CHANDESRIS, *Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications*, Intern. J. Num. Methods. Engin., 58 (2003), pp. 1397-1434.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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

[17] M. J. GANDER AND S. VANDEWALLE, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 29 (2007), pp. 556-578

[18] R. G. GHANEM AND P. D. SPANOS, *Stochastic Finite Elements: A Spectral Approach*, Revised Edition, Dover, 2003.

[19] K. GURUPRASAD, D. E. KEYES, AND J. H. KANE, *GMRES for sequentially multiple nearby systems*, Technical Report (1995), Old Dominium University.

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

[21] G. Horton and S. Vandewalle, *A space-time multigrid method for parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848-864.
[22] 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.
[23] H. Elman and D. Furnival, *Solving the stochastic steady-state diffusion problem using multigrid*, IMA J. Number. Anal., 27 (2007), pp. 675-688.
[24] 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.
[25] C. Jin and X.-C. Cai, *A preconditioned recycling GMRES solver for stochastic Helmholtz problems*, Commun. Comput. Phys., 6 (2009), pp. 342-353.
[26] 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.
[27] 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.
[28] J.-L. Lions, Y. Maday and G. Turinici, *Eásolution d'EDP par un schéma en temps "pararéal"*, C. R. Acad. Sci. Paris Sér. I Math. 332, (2001), pp. 661-668.
[29] M. Loève, *Probability Theory, Vol. I, II*, Springer, New York, 1978.
[30] Y. Maday and G. Turinici, *A parareal in time procedure for the control of partial differential equations*, C. R. Acad. Sci. Paris, Sér. I Math., 335 (2002), pp. 387-392.
[31] 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, Vol. 40, (2005), pp. 441-448.
[32] R. B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20-37.
[33] 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.
[34] 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.
[35] 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.
[36] E. de Sturler, *Nested Krylov methods based on GCR*, J. Comput. Appl. Math., 67 (1996), pp. 15-41.
[37] M.-B. Tran, *Parallel Schwarz waveform relaxation algorithm for an n-dimensional semilinear heat equation*, arXiv preprint arXiv:1006.1323 (2010).
[38] S. Vandewalle, *Parallel Multigrid Waveform Relaxation for Parabolic Problems*, B.G. Teubner Verlag, Stuttgart, 1993.
[39] 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.
[40] J. White, A. Sangiovanni-Vincentelli, F. Odeh, and A. Ruehli, *Waveform relaxation: theory and practice*, Trans. Soc. Comput. Simul., 2 (1985), pp. 95-133.
[41] 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.