Solving the Stochastic Steady-State Diffusion Problem Using Multigrid

Tengfei Su
tengfesu@math.umd.edu
Applied Mathematics and Scientific Computing Program

Advisor: Howard Elman
elman@cs.umd.edu
Department of Computer Science

May 10, 2016

Abstract

In this project we study multigrid for solving the steady-state diffusion problem with random coefficients. The equation is discretized using the stochastic finite element method, and we apply a multigrid algorithm to solve the Galerkin systems. The solutions can be well approximated by low-rank matrices under certain assumptions. We propose a multigrid method with low-rank truncation and show by numerical experiments that it's more efficient in solving the Galerkin systems than the original multigrid solver.

1 Project Review

1.1 Project Goal

In this project, we study the stochastic steady-state diffusion equation

$$\begin{cases} -\nabla \cdot (c(x,\omega)\nabla u(x,\omega)) = f(x) & \text{in } D \times \Omega \\ u(x,\omega) = 0 & \text{on } \partial D \times \Omega \end{cases} \tag{1.1}$$

with the stochastic coefficient $c(x,\omega) : D \times \Omega \to \mathbb{R}$. We consider the case where we have zero Dirichlet boundary conditions and the source term f is deterministic. The solution of equation (1.1) will be a random field $u(x,\omega) : D \times \Omega \to \mathbb{R}$.

The goal of the project is to solve the diffusion equation following the stochastic finite element method (SFEM) [1], and apply a multigrid algorithm [2] for the Galerkin system. We further explore low-rank approximations to reduce computational efforts.

In the next section we briefly discuss the stochastic finite element method.

1.2 Stochastic FEM

By introducing the Karhunen-Loève (KL) expansion [3], we can write the stochastic coefficient $c(x,\omega)$ in terms of a finite collection of random variables $\{\xi_i\}_{i=1}^m$ which we assume to be independent and identically distributed:

$$c(x,\omega) \approx c_0(x) + \sum_{i=1}^{m} \sqrt{\lambda_k} c_i(x)\xi_i(\omega). \tag{1.2}$$

Here $c_0(x)$ is the mean function, $(\lambda_i, c_i(x))$ is the eigen-pair of the covariance function $r(x,y)$. The weak form of (1.1) is then given as follows:

$$\int_\Gamma \rho(\xi) \int_D c(x,\xi)\nabla u(x,\xi)\nabla v(x,\xi)\mathrm{d}x\mathrm{d}\xi = \int_\Gamma \rho(\xi) \int_D f(x)v(x,\xi)\mathrm{d}x\mathrm{d}\xi \tag{1.3}$$

where $\rho(\xi)$ is the joint density function, and Γ is the joint image of $\{\xi_i\}_{i=1}^m$.

The finite-dimensional subspace is defined as

$$V^{hp} = T \otimes S = \mathrm{span}\{\phi(x)\psi(\xi), \phi \in S, \psi \in T\}. \tag{1.4}$$

We use piecewise bilinear functions $\phi(x)$ for the basis of the spatial space, and m-dimensional orthonormal polynomials $\psi(\xi)$ [4] for the stochastic space. The total order of $\psi(\xi)$ doesn't exceed p. Given the subspace, we can write the SFEM solution as a linear combination of the basis fuctions

$$u_{hp}(x,\xi) = \sum_{j=1}^{N}\sum_{s=1}^{M} u_{js}\phi_j(x)\psi_s(\xi). \tag{1.5}$$

N and M are the degrees of freedom of the spatial space and stochastic space, respectively.

Substituting (1.2) and (1.5) into (1.3), and taking the test function as any basis function $\phi_l(x)\psi_r(\xi)$, we get matrix form of (1.3) [5]: find $\mathbf{u} \in \mathbb{R}^{MN}$, such that

$$A\mathbf{u} = \mathbf{f}. \tag{1.6}$$

Using the tensor product notation, we have

$$A = G_0 \otimes K_0 + \sum_{i=1}^{m} G_i \otimes K_i, \tag{1.7}$$

where

$$\begin{aligned} G_0(r,s) &= \int_\Gamma \psi_r(\xi)\psi_s(\xi)\rho(\xi)\mathrm{d}\xi,\ G_i(r,s) = \int_\Gamma \xi_i\psi_r(\xi)\psi_s(\xi)\rho(\xi)\mathrm{d}\xi, \\ K_0(l,j) &= \int_D c_0(x)\nabla\phi_l(x)\nabla\phi_j(x)\mathrm{d}x, \\ K_i(l,j) &= \int_D \sqrt{\lambda_i}c_i(x)\nabla\phi_l(x)\nabla\phi_j(x)\mathrm{d}x, \end{aligned} \tag{1.8}$$

$i = 1,\ldots,m$; $r,s = 1,\ldots,M$; $l,j = 1,\ldots,N$. The right-hand side can also be written as a tensor product

$$\mathbf{f} = g_0 \otimes f_0, \tag{1.9}$$

where

$$\begin{aligned} g_0(r) &= \int_\Gamma \psi_r(\xi)\rho(\xi)\mathrm{d}\xi,\ r = 1,\ldots,M, \\ f_0(l) &= \int_D f(x)\phi_l(x)\mathrm{d}x,\ l = 1,\ldots,N. \end{aligned} \tag{1.10}$$

Matrix A is symmetric and positive definite if the problem is well-posed. It's also block-wisely sparse due to the orthogonality of $\psi(\xi)$ (see Figure 1). For the implementation of SFEM, we use the IFISS and SIFISS package [6] to generate the Galerkin system, i.e., the G, K matrices, the right-hand side vector $\mathbf{f}$, and mesh data. Note that we never form the big matrix A. In the next section we discuss the multigrid algorithm used to solve (1.6).

2 Multigrid

2.1 Algorithm

The basic idea of the multigrid solver proposed by Elman and Furnival [2] is that the mesh size h varies for different grid levels, while polynomial degree p is held constant. Define the fine and coarse grid spaces as

$$V^{hp} = T^p \otimes S^h,\ V^{2h,p} = T^p \otimes S^{2h}. \tag{2.1}$$

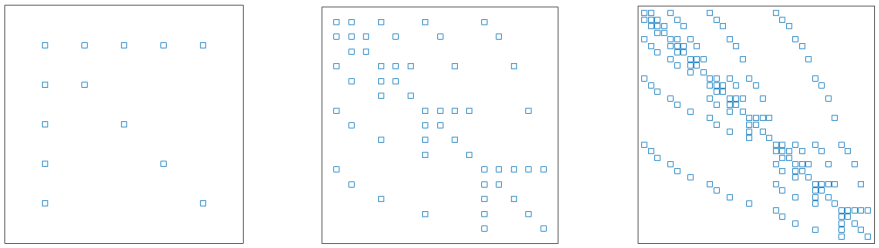

Figure 1: Block structure of A ($m = 4, p = 1, 2, 3$ from left to right. Block size is $N \times N$).

Then the prolongation and restriction operators are in the form of

$$\mathscr{P} = I \otimes P, \ \mathscr{R} = I \otimes P^T, \tag{2.2}$$

where P is the same prolongation matrix as in the deterministic case. We only need to construct K matrices on the coarse grid, and

$$\bar{A} = G_0 \otimes K_0^{2h} + \sum_{i=1}^{m} G_i \otimes K_i^{2h}. \tag{2.3}$$

The damped Jacobi smoother is used for the smoothing steps:

$$Q = \frac{1}{\omega} D, \ D = \mathrm{diag}(A) = I \otimes \mathrm{diag}(K_0). \tag{2.4}$$

Algorithm 1 describes the multigrid solver for stochastic Galerkin system (1.6). In each iteration, we apply one multigrid step for the residual equation

$$A\mathbf{e}^{(i)} = \mathbf{r}^{(i)} = \mathbf{f} - A\mathbf{u}^{(i)} \tag{2.5}$$

and update the solution $\mathbf{u}$ and residual $\mathbf{r}$. On the coarsest grid level we form matrix A and solve the linear system directly using `backslash`.

Algorithm 1: Multigrid for stochastic Galerkin systems

initialization: $i = 0, \mathbf{r}^{(0)} = \mathbf{f}, r_0 = \|\mathbf{f}\|$
while $r > tol * r_0$ & $i \leq maxit$ **do**
  $\mathbf{e}^{(i)} = \texttt{MgIter}(A, \mathbf{0}, \mathbf{r}^{(i)}, level)$
  $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \mathbf{e}^{(i)}$
  $\mathbf{r}^{(i+1)} = \mathbf{f} - A\mathbf{u}^{(i+1)}$
  $r = \|\mathbf{r}^{(i+1)}\|, i = i + 1$
function $\mathbf{u}^{(1)} = \texttt{MgIter}(A, \mathbf{u}^{(0)}, \mathbf{f}, level)$
  if $level == 2$ **then**
    $\mathbf{u}^{(1)} = A \backslash \mathbf{f}$
  else
    for k *steps* **do**
      $\mathbf{u}^{(0)} \leftarrow \mathbf{u}^{(0)} + Q^{-1}(\mathbf{f} - A\mathbf{u}^{(0)})$
    $\bar{\mathbf{r}} = \mathscr{R}(\mathbf{f} - A\mathbf{u}^{(0)})$
    $\bar{\mathbf{e}} = \texttt{MgIter}(\bar{A}, \mathbf{0}, \bar{\mathbf{r}}, level - 1)$
    $\mathbf{u}^{(1)} = \mathbf{u}^{(0)} + \mathscr{P}\bar{\mathbf{e}}$
    for k *steps* **do**
      $\mathbf{u}^{(1)} \leftarrow \mathbf{u}^{(1)} + Q^{-1}(\mathbf{f} - A\mathbf{u}^{(1)})$

2.2 Validation

Consider the model problem with spatial domain $D = (-1, 1)^2$ and source term $f = 1$. The covariance function of $c(x, \omega)$ is in the form of

$$r(x, y) = \sigma^2 \exp(-\frac{1}{b}|x_1 - y_1| - \frac{1}{b}|x_2 - y_2|). \tag{2.6}$$

This is convenient because we have analytical solutions for λ_i and $c_i(x)$. In the KL expansion

$$c(x, \omega) = c_0(x) + \sqrt{3} \sum_{i=1}^{m} \sqrt{\lambda_i} c_i(x) \xi_i(\omega), \tag{2.7}$$

we take ξ_i to be uniformly distributed on $[-1, 1]$. The random field related parameters are selected as follows:

$$c_0(x) = 1,\ \sigma = 0.3,\ b = 2.0. \tag{2.8}$$

First we demonstrate that the multigrid algorithm shows "textbook" convergence behavior: convergence rate is independent of parameters h, m, and p. Fix the relative tolerance as 10^{-6}. Let $m = 5, p = 3$. We see that the number of iterations are basically the same as we vary h:

h	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
n	7	8	8	8	8	8

Similarly, if we fix $m = 3, h = 2^{-3}$ and vary p:

p	1	2	3	4	5	6
n	6	6	7	7	8	8

Let $p = 3, h = 2^{-3}$ and vary m:

m	1	2	3	4	5	6
n	6	7	7	7	8	8

Secondly, we compare the SFEM solutions with the results from Monte Carlo method (MCM) [7]. For MCM, we sample the random variables $\{\xi_i\}_{i=1}^m$ and for each realization solve a deterministic PDE using the finite element method, which gives us the following linear system

$$(K_0 + \sum_{i=1}^{m} \xi_i K_i)\mathbf{u} = f_0. \tag{2.9}$$

The size of the system $(N \times N)$ is much smaller than the stochastic case $(MN \times MN)$. After computing the Monte Carlo solutions $\{u_{MC}^r\}_{r=1}^q$, where q is the sample size, use the following two estimators to calculate the mean value and variance:

$$\begin{aligned} \mathbb{E}[u_{MC}] &= \frac{1}{q}\sum_{r=1}^{q} u_{MC}^r \\ \mathbb{V}[u_{MC}] &= \frac{1}{q-1}\sum_{r=1}^{q}(u_{MC}^r - \mathbb{E}[u_{MC}])^2. \end{aligned} \tag{2.10}$$

Take mesh size $h = 2^{-4}$. Let $m = 3, p = 9, q = 1,000,000$. In Figure 2 we plot the MC solutions and compare them with what we get from SFEM. Compute the relative differences:

$$\begin{aligned} \frac{\|\mathbb{E}[u_{FE}] - \mathbb{E}[u_{MC}]\|_2}{\|\mathbb{E}[u_{MC}]\|_2} &= 3.35 \times 10^{-4}, \\ \frac{\|\mathrm{Var}[u_{FE}] - \mathrm{Var}[u_{MC}]\|_2}{\|\mathrm{Var}[u_{MC}]\|_2} &= 1.17 \times 10^{-3}. \end{aligned} \tag{2.11}$$

This should validate the correctness of the SFEM solutions.

3 Low-Rank Approximation

3.1 Motivation

In the Galerkin system (1.6), solution $\mathbf{u}$ is a long vector

$$\mathbf{u} = [u_{11}, \ldots, u_{N1}, \ldots, u_{1M}, \ldots, u_{NM}]^T \tag{3.1}$$

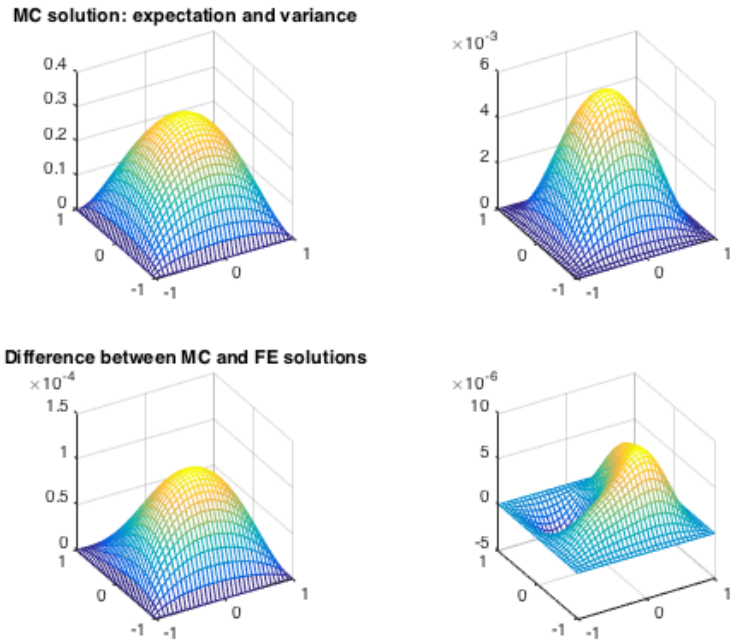

Figure 2: Sample means and variances for MC and SFEM solutions.

and can be written as a matrix

$$U = \mathrm{mat}(\mathbf{u}) = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1M} \\ u_{21} & u_{22} & \cdots & u_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NM} \end{pmatrix}. \tag{3.2}$$

Then (1.6) is equivalent to

$$\mathcal{A}(U) = K_0 U G_0^T + \sum_{i=1}^{m} K_i U G_i^T = F. \tag{3.3}$$

The motivation of low-rank approximation is based on the observation that the decay of singular values for the solution matrix U is very fast (Figure 3), and thus U can be well approximated by a low-rank matrix.

We approximate U by a low-rank matrix U_k and assume $U_k = VW^T$, where $V \in \mathbb{R}^{N\times k}, W \in \mathbb{R}^{M\times k}, k \ll N, M$. Now (3.3) can be written into low-rank format

$$(K_0 V)(G_0 W)^T + \sum_{i=1}^{m} (K_i V)(G_i W)^T = F_l F_r^T. \tag{3.4}$$

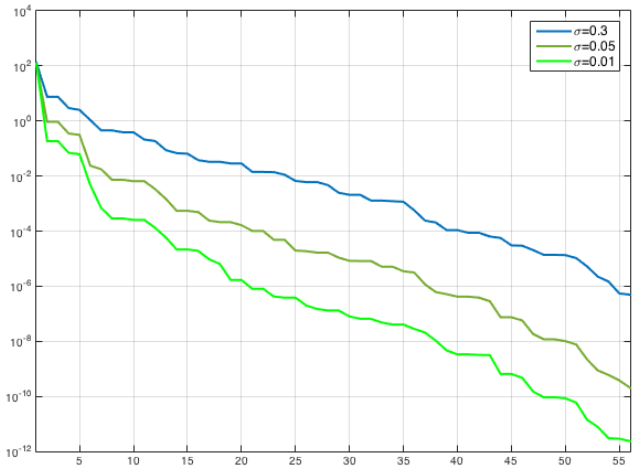

Figure 3: Decay of singular values of U. $h = 2^{-6}, m = 5, p = 3$.

The right-hand side also admits a low-rank decomposition $F = F_l F_r^T$. If we use V, W as iterates in the multigrid algorithm, both memory and computational (matrix-vector products) cost can be reduced from NM to $(N+M)k$. This low-rank format is also convenient for implementation. For instance, the $\mathcal{A}$ operator can be written as the product of two low-rank matrices:

$$\begin{aligned}\mathcal{A}(VW^T) &= (K_0V)(G_0W)^T + \sum_{i=1}^{m}(K_iV)(G_iW)^T \\ &= [K_0V, K_1V, \ldots, K_mV][G_0W, G_1W, \ldots, G_mW]^T. \end{aligned} \tag{3.5}$$

3.2 Algorithm

We use the low-rank format in the multigrid solver and describe the algorithm below. Note that in each iteration the matrix rank grows rapidly (the $\mathcal{A}$ operator increases matrix rank to $(m+1)k$ in the worst case), so we need to truncate the iterates to keep the cost low. In algorithm 2, $\mathscr{S} = I \otimes S$ is the smoothing operator and

$$\mathscr{S}(VW^T) = (SV)(IW)^T. \tag{3.6}$$

The prolongation and restriction operators are implemented in a similar manner.

Algorithm 2: Multigrid with low-rank truncation

initialization: $i = 0, R^{(0)} = F$ in low-rank format, $r_0 = \|F\|$
while $r > tol * r_0$ & $i \leq maxit$ **do**
  $E^{(i)} = \texttt{MgIter}(A, 0, R^{(i)}, level)$
  $U^{(i+1)} = U^{(i)} + E^{(i)}, \qquad U^{(i+1)} = \mathcal{T}_{\text{abs}}(U^{(i+1)})$
  $R^{(i+1)} = F - \mathcal{A}(U^{(i+1)}), \qquad R^{(i+1)} = \mathcal{T}_{\text{abs}}(R^{(i+1)})$
  $r = \|R^{(i+1)}\|, i = i + 1$

function $U^{(1)} = \texttt{MgIter}(A, U^{(0)}, F, level)$
  if $level == 2$ **then**
    solve $\mathcal{A}(U^{(1)}) = F$ directly
  else
    for k *steps* **do**
      $U^{(0)} \leftarrow U^{(0)} + \mathscr{S}(F - \mathcal{A}(U^{(0)})), \qquad U^{(0)} = \mathcal{T}_{\text{rel}}(U^{(0)})$
    $\bar{R} = F - \mathcal{A}(U^{(0)}), \qquad \bar{R} = \mathcal{T}_{\text{rel}}(\bar{R})$
    $\bar{R} = \mathscr{R}\bar{R}$
    $\bar{E} = \texttt{MgIter}(\bar{A}, 0, \bar{R}, level - 1)$
    $U^{(1)} = U^{(0)} + \mathscr{P}\bar{E}$
    for k *steps* **do**
      $U^{(1)} \leftarrow U^{(1)} + \mathscr{S}(F - \mathcal{A}(U^{(1)})), \qquad U^{(1)} = \mathcal{T}_{\text{rel}}(U^{(1)})$

For the low-rank truncation, we use the idea from Kressner and Tobler [8]. Assume $U = VW^T, V \in \mathbb{R}^{N\times k}, W \in \mathbb{R}^{M\times k}$, and $\tilde{U} = \mathcal{T}(U)$ is truncated to rank $\tilde{k}$ with $\tilde{U} = \tilde{V}\tilde{W}^T, \tilde{V} \in \mathbb{R}^{N\times\tilde{k}}, \tilde{W} \in \mathbb{R}^{M\times\tilde{k}}$. First, compute the QR factorization for both V and W

$$V = Q_V R_V, W = Q_W R_W, \text{ so } U = Q_V R_V R_W^T Q_W^T. \tag{3.7}$$

The R matrices are of the size $k \times k$. Next, apply the singular value decomposition for $R_V R_W^T$:

$$R_V R_W^T = \hat{V}\text{diag}(\sigma_1, \ldots, \sigma_k)\hat{W}^T \tag{3.8}$$

where $\sigma_1, \ldots, \sigma_k$ are the singular values. Now we can take the first $\tilde{k}$ singular values using a relative criterion so that

$$\sqrt{\sigma_{\tilde{k}+1}^2 + \cdots + \sigma_k^2} \leq \epsilon_{\text{rel}}\sqrt{\sigma_1^2 + \cdots + \sigma_k^2} \tag{3.9}$$

or an absolute one so that

$$\tilde{k} = \max\{k \mid \sigma_k \geq \epsilon_{\text{abs}}\}. \tag{3.10}$$

The total cost of this computation is $O((M + N + k)k^2)$.

3.3 Numerical Results

Consider the same model problem as in section 2.2. The correlation length b affects the decay of λ_i in the KL expansion. m and b are chosen so that

$$\sum_{i=1}^{m} \lambda_i / \sum_{i=1}^{\infty} \lambda_i \geq 95\%. \tag{3.11}$$

In the following the performance of the multigrid solver with low-rank truncation is studied. The algorithm stops converging after a few iterations. We also run the multigrid solver without truncation to reach a comparable relative residual. The numerical results, i.e. the rank of matrix U, number of iterations, and elapsed time for solving the Galerkin system, are given in Table 1-3. All computations are done in MATLAB R2015a on a MacBook with 1.6 GHz Intel Core i5 and 4 GB SDRAM.

We have observed when the standard deviation σ in the covariance function (2.6) is smaller, the singular values of the matrix U decay faster, and it's more suitable for low-rank approximation. In Figure 4 we plot the best approximations we can get by taking the largest k singular values of U (lines with markers). For example, when $\sigma = 0.01, k = 6$, the relative residual is below 10^{-4}. However, we will need to keep the first 40 singular values in the case where $\sigma = 0.3$. This is also shown in the numerical results (Table 1). As we increase the value of σ, the rank of matrix U and the computing time also increase. As $\sigma = 0.3$, the low-rank truncation even makes the algorithm slower. This is mainly because of the overhead introduced in the truncation operator when the rank k is relatively large. In the later numerical experiments, we take $\sigma = 0.01$.

Table 2 shows the performance of the multigrid solver for various mesh sizes h, or spatial degrees of freedom N. We can see from the 3rd and 5th columns that multigrid with low-rank truncation uses less time than the standard multigrid solver. This is especially true when N is large. Also, the improvement is more significant (see the 4th and 6th columns) if the problem doesn't require very high accuracy for the solution.

Table 3 shows the case where we have various degrees of freedom M in the stochastic space. The multigrid solver with truncation tolerance 10^{-6} takes about half the time compared with no truncation. Similarly, the difference in elapsed time is more significant if we can use a larger tolerance 10^{-4}.

4 Conclusions

From the discussion above, we can conclude that the multigrid solver works well for the stochastic Galerkin systems and with low-rank approximation the computational efforts can be further reduced. We proposed the multigrid algorithm with low-rank truncation by writing the Galerkin solution

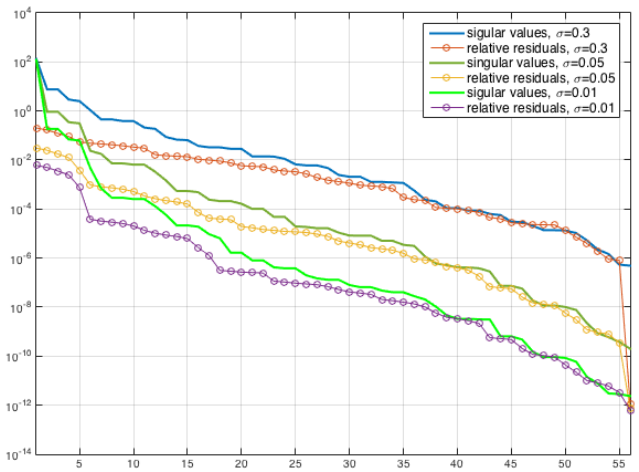

Figure 4: Decay of singular values of U and relative residuals for low-rank approximation. $h = 2^{-6}, m = 5, p = 3$.

	Truncation	10^{-6}	10^{-4}	No truncation	
$\sigma = 0.001$	Rank	6	2		
	Iterations	5	3	5	3
	Elapsed time	1.46	0.27	5.50	3.34
	Rel residual	3.0e-6	6.2e-4	1.2e-6	2.2e-4
$\sigma = 0.01$	Rank	13	6		
	Iterations	5	3	4	3
	Elapsed time	2.58	0.73	4.56	3.39
	Rel residual	1.1e-5	6.0e-4	1.6e-5	2.2e-4
$\sigma = 0.1$	Rank	40	14		
	Iterations	7	4	4	3
	Elapsed time	11.2	2.09	4.58	3.47
	Rel residual	1.7e-5	1.1e-3	1.9e-5	2.4e-4
$\sigma = 0.3$	Rank	55	35		
	Iterations	9	6	6	3
	Elapsed time	29.1	7.90	6.57	3.30
	Rel residual	1.8e-5	1.7e-3	1.4e-5	1.7e-3

Table 1: Performance of multigrid solver with truncation tolerance 10^{-6}, 10^{-4}, and no truncation for various σ. $N = 16129, M = 56$.

	Truncation	10^{-6}	10^{-4}	No truncation	
	Rank	13	6		
$nc = 7$	Iterations	5	3	4	3
$N = 16129$	Elapsed time	2.58	0.73	4.56	3.39
	Rel residual	1.1e-5	6.0e-4	1.6e-5	2.2e-4
	Rank	16	6		
$nc = 8$	Iterations	5	3	4	3
$N = 65025$	Elapsed time	13.20	3.14	18.87	14.08
	Rel residual	1.3e-5	5.8e-4	1.8e-5	2.3e-4
	Rank	11	5		
$nc = 9$	Iterations	5	3	4	2
$N = 261121$	Elapsed time	34.20	12.54	80.89	39.64
	Rel residual	4.6e-5	1.7e-3	1.9e-5	3.3e-3
	Rank	11	2		
$nc = 10$	Iterations	5	2	4	2
$N = 1046529$	Elapsed time	207.12	21.14	1144.10	572.71
	Rel residual	4.5e-5	6.7e-3	1.9e-5	3.3e-3

Table 2: Performance of multigrid solver with truncation tolerance 10^{-6}, 10^{-4}, and no truncation for various N. $N = (2^{nc} - 1)^2$. $\sigma = 0.01, M = 56$.

	Truncation	10^{-6}	10^{-4}	No truncation	
	Rank	13	6		
$m = 5$	Iterations	5	3	4	3
$M = 56$	Elapsed time	2.58	0.73	4.56	3.39
	Rel residual	1.1e-5	6.0e-4	1.6e-5	2.2e-4
	Rank	24	8		
$m = 7$	Iterations	6	4	5	3
$M = 84$	Elapsed time	7.56	1.84	14.84	8.77
	Rel residual	5.5e-6	4.6e-4	1.2e-6	2.2e-4
	Rank	36	10		
$m = 9$	Iterations	7	4	5	3
$M = 220$	Elapsed time	16.93	4.88	31.90	19.04
	Rel residual	4.0e-6	1.7e-4	1.2e-6	2.2e-4
	Rank	47	12		
$m = 11$	Iterations	7	5	5	4
$M = 364$	Elapsed time	31.86	7.21	61.40	49.86
	Rel residual	2.4e-6	7.5e-5	1.2e-6	1.6e-5

Table 3: Performance of multigrid solver with truncation tolerance 10^{-6}, 10^{-4}, and no truncation for various M. $M = (m+p)!/(m!p!)$. $\sigma = 0.01, N = 16129, p = 3$.

into matrix form and introducing low-rank truncation for the iterates. Deliverables of this project include MATLAB code for

- Multigrid for stochastic Galerkin systems,
- Multigrid with low-rank truncation, and
- Monte Carlo method for the stochastic diffusion equation,

and the presentations and reports.

We have seen from our numerical experiments that the QR factorization is the most expensive part in the multigrid solver with low-rank truncation. It takes more than half of the total time used for solving the linear system. It will be beneficial if this can be done more efficiently. The low-rank truncation we used is based on singular values. It's not clear if one can take advantage of the grid hierarchy and propose a better truncation strategy. This will be some future work that is worth looking into.

σ	0.01	0.1	0.3
QR	51.2%	67.7%	73.2%
SVD	3.4%	2.2%	2.2%

Table 4: Time consumption of QR and SVD in Algorithm 2. N=16129, M=56.

References

[1] Ghanem, R. G. & Spanos, P. D. (2003). *Stochastic Finite Elements: A Spectral Approach*. New York: Dover Publications.

[2] Elman, H. & Furnival D. (2007). Solving the stochastic steady-state diffusion problem using multigrid. *IMA Journal of Numerical Analysis*, 27, 675–688.

[3] Loève, M. (1960). *Probability Theory*. New York: Van Nostrand.

[4] Xiu, D. & Karniadakis G. M. (2003). Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of Computational Physics*, 187, 137–167.

[5] Powell, C. & Elman H. (2009). Block-diagonal preconditioning for spectral stochastic finite-element systems. *IMA Journal of Numerical Analysis* 29, 350–375.

[6] Silvester, D., Elman, H. & Ramage, A. Incompressible Flow and Iterative Solver Software,
`https://www.cs.umd.edu/users/elman/ifiss3.4.html`

[7] Lord, G. J., Powell, C. E., & Shardlow, T. (2014). *An Introduction to Computational Stochastic PDEs*. No. 50. Cambridge University Press.

[8] Kressner D. & Tobler C. (2011). Low-rank tensor Krylov subspace methods for parametrized linear systems. *SIAM Journal of Matrix Analysis and Applications* 32.4, 1288–1316.