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

December 13, 2015

Abstract

In this project we use multigrid method to solve the stochastic steady-state diffusion problem. We follow the stochastic finite element formulation for the discretization of the problem, and apply a multigrid algorithm to solve the linear system. Implementing the multigrid method to generate low-rank approximate solutions is studied as the second part of the project.

Contents

1 Project Review

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 $c(x,\omega) : D \times \Omega \to \mathbb{R}$. Note that we are considering the case where we have zero Dirichlet boundary condition 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) (Ghanem & Spanos, 2003), and apply a multigrid algorithm (Elman & Furnival, 2007) for the Galerkin system which is obtained from the finite element method.

In the next section we will briefly discuss the stochastic finite element method, which has beed detailed in the proposal report.

2 Stochastic FEM

By introducing the Karhunen-Loève (KL) expansion (Powell & Elman, 2009), we can write the stochastic coefficient $c(x,\omega)$ in terms of a finite collection of random variables $\{\xi_k\}_{k=1}^m$ which we assume to be independent and identically distributed:

$$c(x,\omega) = c_0(x) + \sum_{k=1}^{m} \sqrt{\lambda_k} c_k(x)\xi_k(\omega). \tag{2.1}$$

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

$$\int_\Gamma \rho(\xi) \int_D c(x,\xi)\nabla u(x,\xi)\nabla v(x,\xi)\mathrm{d}x\mathrm{d}\xi = \int_\Gamma \rho(\xi) \int_D f(x)v(x,\xi)\mathrm{d}x\mathrm{d}\xi \tag{2.2}$$

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

The finite-dimensional subspace is defined as

$$V^h = T \otimes S = \mathrm{span}\{\phi(x)\psi(\xi), \phi \in S, \psi \in T\}. \tag{2.3}$$

We are using the piecewise bilinear functions $\phi(x)$ for the discretization of spatial domain, and m-dimensional orthogonal polynomials $\psi(\xi)$ (Xiu & Karniadakis, 2003) 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:

$$u_{hp}(x,\xi) = \sum_{j=1}^{N}\sum_{s=1}^{M} u_{js}\phi_j(x)\psi_s(\xi). \tag{2.4}$$

Substituting (2.1) and (2.4) into (2.2), and taking the test function as any basis function $\phi_i(x)\psi_r(\xi)$, we get matrix form of (2.2): find $\mathbf{u} \in \mathbb{R}^{MN}$, such that

$$A\mathbf{u} = \mathbf{f}. \tag{2.5}$$

Using the tensor product notation, we have

$$A = G_0 \otimes K_0 + \sum_{k=1}^{m} G_k \otimes K_k, \tag{2.6}$$

where

$$\begin{aligned}
G_0(r,s) &= \int_\Gamma \psi_r(\xi)\psi_s(\xi)\rho(\xi)\mathrm{d}\xi \\
G_k(r,s) &= \int_\Gamma \xi_k\psi_r(\xi)\psi_s(\xi)\rho(\xi)\mathrm{d}\xi, \quad r,s = 1,\ldots,M \\
K_0(i,j) &= \int_D c_0(x)\nabla\phi_i(x)\nabla\phi_j(x)\mathrm{d}x \\
K_k(i,j) &= \int_D \sqrt{\lambda_k}c_k(x)\nabla\phi_i(x)\nabla\phi_j(x)\mathrm{d}x, \quad i,j = 1,\ldots,N.
\end{aligned} \tag{2.7}$$

All the G and K matrices are symmetric and have a sparse structure due to the orthogonality of $\psi(\xi)$ and local support of $\phi(x)$. In fact, G_0 is an identity matrix; $G_k(k = 1, \ldots, m)$ has at most 2 non-zero entries in each row, and all the diagonal entries are zeros. The right-hand side is a long vector of length MN which can also be written as a tensor product

$$\mathbf{f} = g_0 \otimes f_0, \tag{2.8}$$

where

$$\begin{aligned} g_0(r) &= \int_\Gamma \psi_r(\xi)\rho(\xi)\mathrm{d}\xi, \quad r = 1, \ldots, M, \\ f_0(i) &= \int_D f(x)\phi_i(x)\mathrm{d}x, \quad i = 1, \ldots, N. \end{aligned} \tag{2.9}$$

For the implementation of SFEM, we use the IFISS and SIFISS package (Silvester et al) to generate the Galerkin system, i.e., the G, K matrices, the right-hand side vector $\mathbf{f}$, mesh data, and other input parameters. Note that we never form the big matrix A. The main task next will be writing the multigrid solver for

$$(G_0 \otimes K_0 + \sum_{k=1}^{m} G_k \otimes K_k)\mathbf{u} = \mathbf{f}. \tag{2.10}$$

3 Multigrid

The multigrid solver is a recursive version of the two-grid correction scheme (Elman & Furnival, 2007):

Two-grid Correction Scheme
Choose initial guess $\mathbf{u}^{(0)}$
for $i = 0$ until convergence
    for k steps
        $\mathbf{u}^{(i)} \leftarrow \mathbf{u}^{(i)} + Q^{-1}(\mathbf{f} - A\mathbf{u}^{(i)})$ (pre-smoothing)
    end
    $\bar{\mathbf{r}} = \mathscr{R}(\mathbf{f} - A\mathbf{u}^{(i)})$ (restrict residual)
    solve $\bar{A}\bar{\mathbf{e}} = \bar{\mathbf{r}}$
    $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \mathscr{P}\bar{\mathbf{e}}$ (prolong and update)
    for k steps
        $\mathbf{u}^{(i+1)} \leftarrow \mathbf{u}^{(i+1)} + Q^{-1}(\mathbf{f} - A\mathbf{u}^{(i+1)})$ (post-smoothing)
    end
end

In the next few sections, we will talk about the essential components of the two-grid scheme, including the prolongation operator $\mathscr{P}$, the restriction operator $\mathscr{R}$, the coarse version of coefficient matrix $\bar{A}$, and the smoother

Q^{-1}. In the end, we will have the multigrid routine built on this two-grid scheme.

3.1 Prolongation Operator

For the two-grid correction scheme, we have two grid spaces: the fine grid space

$$V^h = T^p \otimes S^h,\ \dim(V^h) = M \times N_h$$

and the coarse grid space

$$V^{2h} = T^p \otimes S^{2h},\ \dim(V^{2h}) = M \times N_{2h}.$$

Note that the mesh size varies from h to $2h$, while the polynomial order p, which corresponds to the subspace T, is held constant.

The prolongation operator $\mathscr{P}$ maps a function in V^{2h} to V^h. In other words, for any $v_{2h} \in V^{2h}$, if $\mathbf{v}^{2h}$ is the coefficient vector of v_{2h} in V^{2h}, then the coefficient vector of v_{2h} in V^h is $\mathscr{P}\mathbf{v}^{2h}$. Any basis function $\phi_j^{2h} \in S^{2h}$ can be written as

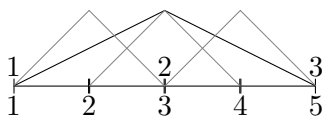
$$\phi_j^{2h} = \sum_{i=1}^{N_h} p_{ij}\phi_i^h,\ j = 1, \ldots, N_{2h} \tag{3.1}$$

since $S^{2h} \subset S^h$. Then the prolongation operator is given by

$$\mathscr{P} = I \otimes P, \text{ with } P_{ij} = p_{ij}. \tag{3.2}$$

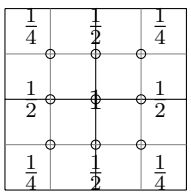
Therefore, we need to use the relationship in (3.1) to construct matrix P, whose j^{th} column consists of the coefficients of the linear combination for ϕ_j^{2h}.

In 1D, each basis function in S^{2h} can be written into an interpolation of 3 basis functions in S^h. For instance, if we use a uniform mesh as shown in the figure below, we have



$$\phi_2^{2h} = \frac{1}{2}\phi_2^h + \phi_3^h + \frac{1}{2}\phi_4^h.$$

In 2D, any basis function in S^{2h} will be a linear combination of the adjacent 9 basis function in S^h, with coefficients given in the following picture:



In other words, there will be 9 non-zero entries in each column of P.

Once we have matrix P, the prolongation operator is $I \otimes P$, the restriction operator is defined as

$$\mathscr{R} = I \otimes P^T, \tag{3.3}$$

and we have the following relationship

$$\bar{A} = \mathscr{R} A \mathscr{P}. \tag{3.4}$$

3.2 Construction of $\bar{A}$

Combining (2.10) and (3.2)-(3.4), we have

$$\begin{aligned} \bar{A} = \mathscr{R} A \mathscr{P} &= G_0 \otimes (P^T K_0 P) + \sum_{k=1}^{m} G_k \otimes (P^T K_k P) \\ &= G_0 \otimes \bar{K}_0 + \sum_{k=1}^{m} G_k \otimes \bar{K}_k. \end{aligned} \tag{3.5}$$

So for $\bar{A}$, we have the same G matrices but different K matrices. Given K (omitting the subscripts here), we can construct $\bar{K}$ by either (1) computing the matrix product $P^T K P$ or (2) assembling it directly on the coarse grid using (2.7). In fact, for the diffusion problem, we will end up with the same matrix since (Elman et al, 2014)

$$\begin{aligned} (P^T K P)_{ij} &= \sum_m \sum_n P^T_{im} \left(\int_D \sqrt{\lambda_k} c_k(x) \nabla \phi^h_m(x) \nabla \phi^h_n(x) \mathrm{d}x \right) P_{nj} \\ &= \int_D \sqrt{\lambda_k} c_k(x) \sum_m P_{mi} \nabla \phi^h_m(x) \sum_n P_{nj} \nabla \phi^h_n(x) \mathrm{d}x \\ &= \int_D \sqrt{\lambda_k} c_k(x) \nabla \phi^{2h}_i(x) \nabla \phi^{2h}_j(x) \mathrm{d}x. \end{aligned} \tag{3.6}$$

We will assemble the K matrices directly on the coarse grid because it's cheaper in computation than doing the matrix product, especially when K is sparse.

3.3 Smoother

We use the damped Jacobi smoother which is defined as follows: given matrix splitting $A = D - (-L - U)$, then

$$Q^{-1} = \omega D^{-1} \tag{3.7}$$

where ω is the damping coefficient, and D contains the diagonal components of A. Using the fact that G_0 is identity, and G_k has zero diagonal entries,

we have

$$\begin{aligned}\operatorname{diag}(A) &= \operatorname{diag}(G_0 \otimes K_0 + \sum_{k=1}^{m} G_k \otimes K_k) \\ &= \operatorname{diag}(I \otimes K_0) \end{aligned} \tag{3.8}$$

This can be used to simplify our definition of the smoother Q^{-1}.

3.4 Matrix Vector Product

In this section we show how we do the matrix vector products when the matrix is given as a tensor product. For instance, $A = G_0 \otimes K_0 + \sum_{k=1}^{m} G_k \otimes K_k$. We write the vector

$$\mathbf{u} = [u_{11}, \ldots, u_{N1}, \ldots, u_{1M}, \ldots, u_{NM}]^{\mathrm{T}} \tag{3.9}$$

as a matrix

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1M} \\ u_{21} & u_{22} & \cdots & u_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NM} \end{pmatrix}. \tag{3.10}$$

Then we can compute the matrix vector product via

$$A\mathbf{u} = \operatorname{vec}(K_0 U G_0 + \sum_{k=1}^{m} K_k U G_k), \tag{3.11}$$

where vec means writing the matrix into a long vector column-wisely.

This corresponds to our `stoch_matvec` function, which is used in several places in the code, such as computing the residual, the smoothing step, the prolongation operation, and the restriction operation.

3.5 Multigrid Solver

With the several components discussed above, now we are ready to write the multigrid solver. In the following is the function which executes one multigrid iteration, or one V-cycle. The function is called recursively, until we reach the coarsest grid level. The coarsest mesh we have is 4 by 4 in 2D, with 9 interior nodes. On this level we form the coefficient matrix A explicitly and solve the linear system directly using backslash.

```
function x=stoch_mg_iter(Ks,G,x0,f,smoother,level,
   npre,npost)
%STOCH_MG_ITER performs one stochastic MG iteration
% input
%    Ks            coefficient matrix K
```

```
%     G              coefficient matrix G
%     x0             initial iterate
%     f              right-hand side
%     smoother       smoothing operator
%     level          grid level
%     npre           number of presmoothing steps
%     npost          number of postsmoothing steps
% output
%     x              result of multigrid step
K=Ks(level).matrix; P{1}=Ks(level).prolong;
R{1}=P{1}'; I{1}=speye(size(G{1}));
% coarsest grid level
if level==2
   dimk=length(K);
   A=kron(G{1},K{1});
   for dim=2:dimk
       A=A+kron(G{dim},K{dim});
   end
   x=A\f;
else
   % pre-smoothing
   x=stoch_mg_pre(K,G,x0,f,npre,smoother,level);
   % restrict residual
   r=f-stoch_matvec(x,G,K);
   rc=stoch_matvec(r,I,R);
   % coarse grid correction
   cc=stoch_mg_iter(Ks,G,zeros(size(rc)),rc,smoother
      ,level-1,npre,npost);
   % prolong and update
   x=x+stoch_matvec(cc,I,P);
   % post-smoothing
   x=stoch_mg_post(K,G,x,f,npost,smoother,level);
end
```

Finally the multigrid algorithm solves the Galerkin systems in the following steps:

1. Construct K, P, Q^{-1} on each grid level
2. Initialize $\mathbf{u}^{(0)} = \mathbf{0}$, $r0 = \text{norm}(\mathbf{f} - A\mathbf{u}^{(0)})$, $i = 0$
3. while $r > tol * r0$ & $i <= maxit$
   execute one multigrid iteration for $A\mathbf{u} = \mathbf{f}$:
   $\mathbf{u}^{(i+1)} = \text{stoch\_mg\_iter}(K, G, \mathbf{u}^{(i)}, \mathbf{f}, \ldots)$
   $r = \text{norm}(\mathbf{f} - A\mathbf{u}^{(i+1)})$

$i = i + 1$
end

In the while loop above, we may also equivalently apply the multigrid iteration for the residual equation

execute one multigrid iteration for $A\mathbf{e} = \mathbf{r}$:
$\mathbf{e}^{(i+1)} = \text{stoch\_mg\_iter}(K, G, \mathbf{0}, \mathbf{f} - A\mathbf{u}^{(i)}, \ldots)$
$\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \mathbf{e}^{(i+1)}$
$r = \text{norm}(\mathbf{f} - A\mathbf{u}^{(i+1)})$
$i = i + 1$

4 Validation

4.1 Model Problem

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

$$r(x, y) = \sigma^2 e^{-\frac{1}{b_1}|x_1 - y_1| - \frac{1}{b_2}|x_2 - y_2|}. \tag{4.1}$$

This is convenient because we have analytical solutions for λ_k and $c_k(x)$ (Ghanem & Spanos, 2003). In the KL expansion

$$c(x, \omega) = c_0(x) + \sigma\sqrt{3}\sum_{k=1}^{m} \sqrt{\lambda_k} c_k(x)\xi_k(\omega), \tag{4.2}$$

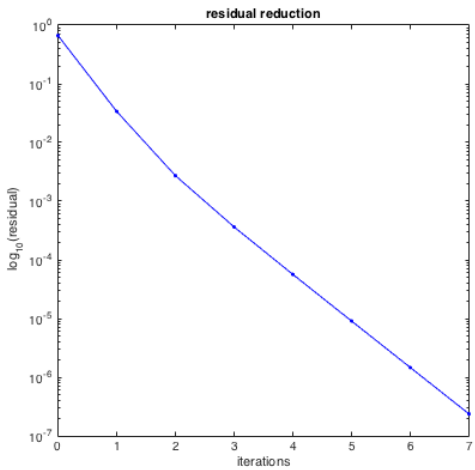
we take ξ_k to be uniformly distributed on $[-1, 1]$, so $\Gamma = [-1, 1]^m$, $\rho(\xi) = 1/2^m$. The random variable $\sigma\sqrt{3}\xi_k$ will have mean zero and standard deviation σ (Bespalov et al, 2014). The random field related parameters are selected as follows:

$$c_0(x) = 1,\ \sigma = 0.3,\ b_1 = b_2 = 2.0. \tag{4.3}$$

Take $h = 2^{-4}$, $m = 3$, $p = 3$, tol $= 10^{-6}$. The figure below shows the reduction of the residual. It takes 7 iterations for the multigrid solver to converge.

4.2 Convergence Performance

It has been shown theoretically in Elman and Furnival's paper (2007) that the convergence rate of the multigrid algorithm is independent of h, m, and p. In other words, the number of iterations required for multigrid to converge to a fixed error tolerance is independent of h, m, and p. In this section we demonstrate that our algorithm has such property.



To show the independence of mesh size h, we select the values of m and p and refine mesh size. From the following tables we see that the number of iterations n is basically constant. The increase in the number of iterations is neglectable compared to the increase in the size of the Galerkin system.

$m = 3, p = 3$

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

$m = 5, p = 3$

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

$m = 3, p = 5$

h	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
n	7	8	8	8	8	9

Similarly by fixing $h = 2^{-3}$, we vary the values of m and p to show that the convergence rate is independent of these two parameters as well.

$(m = 2)$ p	1	2	3	4	5	6
n	6	6	7	7	7	7

$(m = 3)$ p	1	2	3	4	5	6
n	6	6	7	7	8	8

$(p = 2)$ m	1	2	3	4	5	6
n	6	6	6	7	7	7

$(p = 3)$ m	1	2	3	4	5	6
n	6	7	7	7	8	8

4.3 Monte Carlo Method

The second part of validation is to compare the SFEM solution with computational results from Monte Carlo method (MCM). Since the SFEM solution is a random field

$$u_{hp}(x,\xi) = \sum_{j=1}^{N}\sum_{s=1}^{M} u_{js}\phi_j(x)\psi_s(\xi), \tag{4.4}$$

we will compare the mean value and variance to show that we are getting the correct result. Using the orthogonality relationship

$$\mathbb{E}[\psi_r(\xi)\psi_s(\xi)] = \delta_{rs} \tag{4.5}$$

and the fact that $\psi_1(\xi) = 1$, we have $\mathbb{E}[\psi_i(\xi)] = \delta_{i1}$, and thus

$$\mathbb{E}[u_{hp}(x,\xi)] = \sum_{j=1}^{N} u_{j1}\phi_j(x). \tag{4.6}$$

Besides, $\phi_j(x_i) = \delta_{ij}$ at a grid point x_i, so

$$\mathbb{E}[u_{hp}(x_j,\xi)] = u_{j1},\ j = 1,\ldots,N. \tag{4.7}$$

Similarly,

$$\mathbb{E}[u_{hp}^2(x,\xi)] = \sum_{s=1}^{M}(\sum_{j=1}^{N} u_{js}\phi_j(x))^2, \tag{4.8}$$

$$\begin{aligned}\mathbb{V}[u_hp(x_j,\xi)] &= \mathbb{E}[u_{hp}^2(x_j,\xi)] - (\mathbb{E}[u_{hp}(x_j,\xi)])^2 \\ &= (\sum_{s=1}^{M} u_{js}^2) - u_{j1}^2 = \sum_{s=2}^{M} u_{js}^2 \end{aligned} \tag{4.9}$$

for $j = 1,\ldots,N$. Therefore, we can use (4.7) and (4.9) to calculate the mean value and variance of the SFEM solution.

For the Monte Carlo method (Lord et al, 2014), we sample $\{\xi_k\}_{k=1}^m$ via

$$\xi = 2 * \texttt{rand}(m,1) - 1, \tag{4.10}$$

with each component uniformly distributed on $[-1,1]$. Then for each realization of ξ, we solve a deterministic PDE using a normal finite element method, which gives us the following linear system

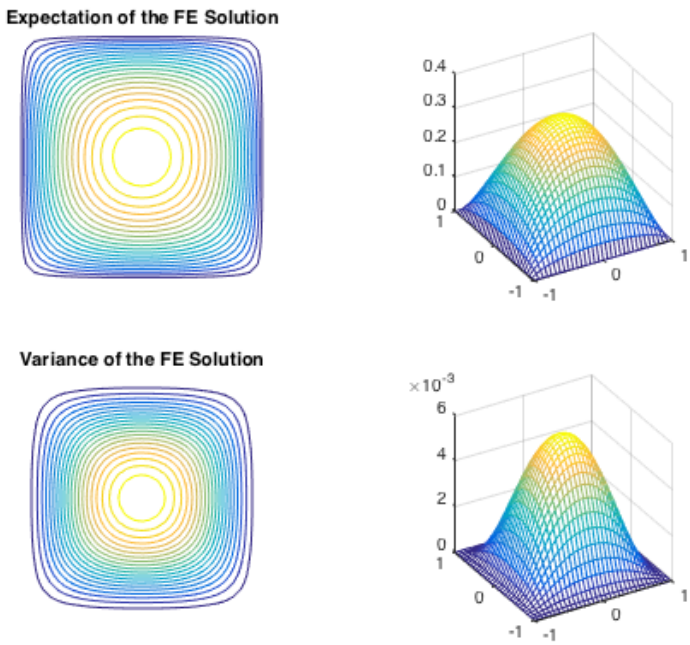
$$(K_0 + \sum_{k=1}^{m} \xi_k K_k)\mathbf{u} = f_0. \tag{4.11}$$

We have the same K matrices and f_0 as in (2.7) and (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 number of sampling, we can use the following two estimators to calculate the mean value and variance :

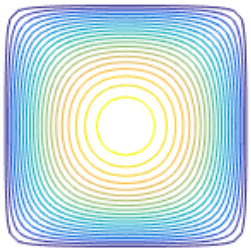
$$\mathbb{E}[u_{MC}] = \frac{1}{q}\sum_{r=1}^{q} u_{MC}^r \tag{4.12}$$

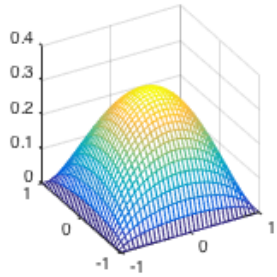
$$\begin{aligned}\mathbb{V}[u_{MC}] &= \frac{1}{q-1}\sum_{r=1}^{q}(u_{MC}^r - \mathbb{E}[u_{MC}])^2 \\ &= \frac{1}{q-1}\left(\sum_{r=1}^{q}(u_{MC}^r)^2 - q(\mathbb{E}[u_{MC}])^2\right).\end{aligned} \tag{4.13}$$

Take the same mesh size $h = 2^{-4}$ for SFEM and MCM. Let $m = 3$, $p = 9$, and $q = 1,000,000$. In the following pictures, we plot the mean values and variances of SFEM solution and MCM solution, as well as their differences. We can see that the differences are relatively very small (especially for the mean values). The differences can be further reduced if we have a larger q. This should validate the correctness of our SFEM solution.

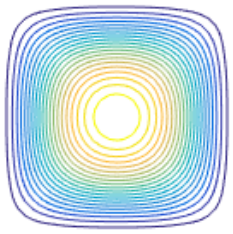


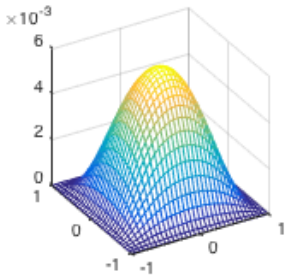
Expectation of the MC Solution



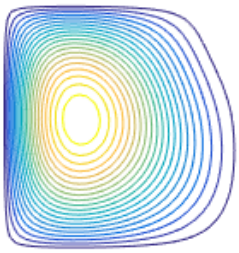


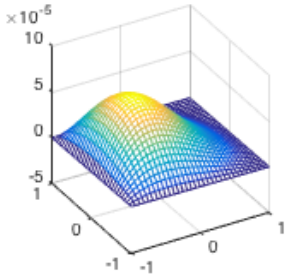
Variance of the MC Solution



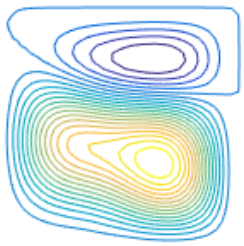


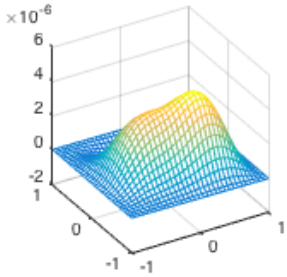
Difference in Expectation





Difference in Variance





5 Schedule

5.1 Project Status

Here is the proposed timeline and what we have accomplished so far:

- 10/15 Generate Galerkin system from IFISS/S-IFISS ✓
- 10/22 Write the multigrid routine and implement for model problem
  - Uniform distributions ✓
  - Normal distributions ✗
- 11/19 Validation
  - Convergence performance ✓
  - Comparison with Monte Carlo ✓
- 11/26 Prepare for mid-year presentation ✓
- 01/25 Validation (if not finished yet)
- 02/08 Implement multigrid for low-rank approximate solutions
- 03/07 Implement BiCGstab for low-rank approximate solutions (if time permits)
- 04/04 Collect computational results
- 04/25 Prepare for final presentation

Now we have the multigrid algorithm for solving the Galerkin system which is obtained from the stochastic FEM. We also have the code for Monte Carlo method for the stochastic diffusion problem.

5.2 Future Work

The second part of the project follows the idea of combining iterative methods for solving linear systems with low-rank approximation in Kressner and Tobler's paper (2011). As discussed in section 3.4, we can write the Galerkin system (2.10) as

$$K_0 U G_0 + \sum_{k=1}^{m} K_k U G_k = F, \tag{5.1}$$

where

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1M} \\ u_{21} & u_{22} & \cdots & u_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NM} \end{pmatrix}, F = \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1M} \\ f_{21} & f_{22} & \cdots & f_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ f_{N1} & f_{N2} & \cdots & f_{NM} \end{pmatrix}.$$

The computational cost can be further reduced if we can write U as

$$U \approx U_k = V_k W_k^{\mathrm{T}},\ V_k \in \mathbb{R}^{N\times k}, W_k \in \mathbb{R}^{M\times k}, k \ll N, M \tag{5.2}$$

and use iterative methods to solve the matrix version of the system. In the next semester, we will follow this idea and apply our multigrid solver to generate low-rank approximate solutions.

6 Bibliography

Bespalov, A., Powell, C. E., & Silvester, D. (2014). Energy norm a posteriori error estimation for parametric operator equations. *SIAM Journal on Scientific Computing*, 36(2), A339-A363.

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

Elman, H. C., Silvester, D. J., & Wathen, A. J. (2014). *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Oxford: Oxford University Press.

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

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.

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

Powell, C. E., & Elman, H. C. (2007). Block-diagonal preconditioning for spectral stochastic finite-element systems. *IMA Journal of Numerical Analysis*, 29, 350–375.

Silvester, D., Elman, H. C., & Ramage, A. Incompressible Flow and Iterative Solver Software, http://www.maths.manchester.ac.uk/∼djs/ifiss.

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