SIAM/ASA J. Uncertainty Quantification
Vol. 1, pp. 452–493

A Data-Driven Stochastic Method for Elliptic PDEs with Random Coefficients*

Mulin Cheng†, Thomas Y. Hou†, Mike Yan†, and Zhiwen Zhang†

Abstract. We propose a data-driven stochastic method (DSM) to study stochastic partial differential equations (SPDEs) in the multiquery setting. An essential ingredient of the proposed method is to construct a data-driven stochastic basis under which the stochastic solutions to the SPDEs enjoy a compact representation for a broad range of forcing functions and/or boundary conditions. Our method consists of offline and online stages. A data-driven stochastic basis is computed in the offline stage using the Karhunen–Loève (KL) expansion. A two-level preconditioning optimization approach and a randomized SVD algorithm are used to reduce the offline computational cost. In the online stage, we solve a relatively small number of coupled deterministic PDEs by projecting the stochastic solution into the data-driven stochastic basis constructed offline. Compared with a generalized polynomial chaos method (gPC), the ratio of the computational complexities between DSM (online stage) and gPC is of order $O((m/N_p)^2)$. Here m and N_p are the numbers of elements in the basis used in DSM and gPC, respectively. Typically we expect $m \ll N_p$ when the effective dimension of the stochastic solution is small. A timing model, which takes into account the offline computational cost of DSM, is constructed to demonstrate the efficiency of DSM. Applications of DSM to stochastic elliptic problems show considerable computational savings over traditional methods even with a small number of queries. We also provide a method for an a posteriori error estimate and error correction.

Key words. stochastic partial differential equations (SPDEs), data-driven methods, Karhunen–Loève (KL) expansion, uncertainty quantification, reduced-order model

AMS subject classifications. 60H35, 65C30, 35R60

DOI. 10.1137/130913249

1. Introduction.

Uncertainty arises in many complex real-world problems of physical and engineering interests. Many physical applications involving uncertainty quantification can be described by stochastic partial differential equations (SPDEs). One of the essential challenges in these applications is how to solve SPDEs efficiently when the dimension of stochastic input variables is high. In applications, we often need to solve the same SPDE many times with multiple forcing functions or boundary conditions. This is also known as the multiquery problem. Many numerical methods have been proposed in the literature to solve SPDEs; see, e.g., [38, 5, 9, 10, 16, 23, 29, 40, 3, 41, 28, 37, 21, 36, 1, 35, 27, 11, 12, 30, 33]. Most of these methods use a problem-independent basis. These methods are usually very expensive when the dimension of the input stochastic variables is high. There have been some attempts to use a problem-dependent basis to explore the hidden data sparsity structure of the solution; see [31, 34]. However, almost all these methods focus on constructing a reduced spatial basis,

*Received by the editors March 18, 2013; accepted for publication (in revised form) August 21, 2013; published electronically November 12, 2013. This work was supported in part by AFOSR MURI grant FA9550-09-1-0613, DOE grant DE-FG02-06ER25727, and NSF FRG grant DMS-1159138.
http://www.siam.org/journals/juq/1/91324.html

†Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125 (mulinch@caltech.edu, hou@cms.caltech.edu, MIKEYAN@SLCG.COM, zhangzw@caltech.edu).

which depends sensitively on the forcing or the boundary condition. The reduced basis needs to be reconstructed if one changes the forcing function or the boundary condition.

In this paper, we propose a data-driven stochastic method (DSM) to study the multiquery problem for solving SPDEs with a family of forcing functions or boundary conditions. Unlike other reduced basis methods, we focus on constructing a data-driven *stochastic basis* that can be reused for a family of forcing functions or boundary conditions. By exploiting the effective low-dimensional structure of the stochastic solution, our method provides a compact representation of the stochastic solution, which leads to considerable computational savings over traditional methods during the online stage.

Multiquery problems arise in many physical and engineering applications. Here we consider the case where the forcing functions or the boundary conditions are parameterized by a family of deterministic input parameters and the stochastic coefficients that appear in the SPDE are independent of these input parameters. A typical scenario is to study uncertainty in a subsurface flow in which the permeability field is modeled by some stochastic process, and we want to know its responses under different forces [39]. To illustrate the main idea of our approach, we consider an SPDE of the form

$$\mathcal{L}(x,\omega)u(x,\omega) = f(x,\theta), \quad x \in D,\ \omega \in \Omega, \tag{1a}$$

$$u(x,\omega) = 0, \quad x \in \partial D,\ \omega \in \Omega, \tag{1b}$$

where $D \in R^d$ is a bounded spatial domain and $\mathcal{L}(x,\omega)$ is a stochastic differential operator. Clearly, $\mathcal{L}(x,\omega)$ represents the random part of the problem, while $f(x,\theta)$ is the deterministic forcing function parameterized by θ. $u(x,\omega)$ is the stochastic solution.

Our DSM uses the Karhunen–Loève (KL) expansion [22, 25] of the SPDE solutions. The KL expansion is well known for generating the optimal basis in the sense that its first m-term truncation gives the smallest mean square error among all expansions using an orthonormal basis. As a result, it gives the most compact representation of a stochastic solution. More details about KL expansion will be elaborated in section 2.1. We note that the stochastic basis generated by the KL expansion is problem dependent and is a functional of the input stochastic variable. Moreover, the mapping between the input stochastic variable and the stochastic basis is nonlinear. The KL expansion has found many applications in statistics, image processing, and uncertainty quantification. In these applications, the eigenvalues of the covariance function are often found to decay very quickly, which indicates that these stochastic solutions have certain low-dimensional structures. How to extract a compact data-driven stochastic basis from the KL expansion of the stochastic solution with a family of forcing functions is the main focus of our paper.

We remark that a dynamically biorthogonal (DyBO) method has been proposed and developed to solve time-dependent SPDEs; see [6, 7, 8]. By solving an equivalent system that governs the evolution of the spatial and stochastic basis, the DyBO method essentially tracks the KL expansion of the stochastic solution on the fly without the need of solving the expensive eigenvalue problem associated with the covariance matrix. Applications of the DyBO method to the one-dimensional (1D) Burgers equation, the two-dimensional (2D) incompressible Navier–Stokes equations, and the 2D Boussinesq approximation with Brownian forcings show that the DyBO method can solve nonlinear time-dependent SPDEs accurately and efficiently.

1.1. Constructing a data-driven stochastic basis offline. One of the main contributions of this paper is that we propose an effective strategy to construct a data-driven basis $\{A_i(\omega)\}_{i=0}^m$ in the offline stage, where $A_0(\omega) = 1$ and m is the number of elements in the basis. As a first step, we construct a compact representation of $f(x,\theta)$ by assuming that the forcing function $f(x,\theta)$ can be approximated by a finite dimensional basis $f_i(x)$, i.e., $f(x,\theta) \approx \sum_{i=0}^K c_i(\theta) f_i(x)$. Such an expansion can be obtained by applying the singular value decomposition (SVD) or empirical interpolation method (EIM) [2] to $f(x,\theta)$. With such a parametrization of f, we begin our construction of the stochastic basis $\{A_i(\omega)\}_{i=0}^m$ based on the KL expansion of the SPDE solution of (1) with $f_0(x)$ as a forcing function. We propose an error analysis to evaluate the completeness of the data-driven basis $\{A_i(\omega)\}_{i=0}^m$. To ensure that the stochastic basis $\{A_i\}$ is applicable to the entire range of forcing functions $f(x,\theta)$, we design a two-level algorithm to enrich the stochastic basis based on the trial functions $f_k(x)$, $k = 1, 2, \ldots, K$. When this enriching process is done, the resulting data-driven basis $\{A_i(\omega)\}_{i=0}^m$ provides a compact representation of the SPDE solutions that can be used to solve this parameterized family of forcing functions. This enriching algorithm is illustrated in Figure 1 in section 3. The detailed implementation of this enriching algorithm depends on the specific numerical representation of the stochastic basis, which will be elaborated in detail in section 3.

1.2. Computing the stochastic solution online. The online stage of the DSM is straightforward. For each $f(x,\theta)$ of interest in (1), i.e., each query (or a choice of θ) in an application, we project the stochastic solution to the stochastic basis that we constructed in the offline stage:

$$u(x,\omega) \approx \sum_{i=0}^{m} u_i(x) A_i(\omega),$$

where $A_0 = 1$ and $u_0(x)$ is the mean of the solution. We use the Galerkin projection to derive a coupled deterministic system of PDEs for $u_i(x)$ and solve this system by any standard numerical method. To obtain an estimate on the error of our method, we apply a Monte Carlo method to solve the residual equation and obtain an a posteriori error estimate. In general, only a relatively small number of Monte Carlo realizations is needed in this error correction step since the variance of the residual is expected to be small. If the residual error is larger than our prescribed threshold, then we can add the residual error correction to the stochastic solution obtained by the DSM. This would give an improved approximation to the stochastic solution. Once we obtain the numerical solution $u(x,\omega)$, we can use it to compute statistical quantities of interest, such as mean, variance, and joint probability distributions.

1.3. Comparison of computational complexities. We have performed a complexity analysis for our DSM and have compared it with other commonly used methods, such as the generalized polynomial chaos method (gPC) and gSC (generalized stochastic collocation method). Let m and N_p be the numbers of elements in the basis used in DSM and gPC, respectively. Let J be the stochastic collocation points used in gSC. Let t_1 denote the overhead time of generating the stiffness matrix and the Cholesky decomposition in the gPC solver, and let CN_p^2 denote the computation time of one-time forward/back substitution, where C is a constant that depends on the physical grid number. Let t_2 denote the overhead time of DSM in the

offline stage and n denote the total query number. The computational cost of gPC and DSM will be $t_{gPC}(n) = t_1 + nCN_p^2$ and $t_{DSM}(n) = t_2 + nCm^2$, respectively. The computational cost of gSC is $t_{gSC}(n) = nJt_0$, where t_0 is the computing time of the deterministic solver on one collocation point. A simple calculation shows that the DSM solver will be superior to the gPC solver when we need to solve the original SPDE with more than $n_c = [\frac{t_2 - t_1}{C(N_p^2 - m^2)}] + 1$ different forcing functions. Typically, we expect $N_p^2 \gg m^2$. The larger N_p is, the smaller n_c becomes. Similarly, the DSM solver will be superior to the gSC solver when we need to solve the original SPDE with more than $n_c = [\frac{t_2}{Jt_0 - Cm^2}] + 1$ different forcing functions. The comparison between DSM and the Monte Carlo method can be analyzed similarly.

To further reduce n_c, we would like to reduce the overhead time, t_2, in DSM. If we construct the KL expansion by first forming the covariance matrix and then solving the large-scale eigenvalue problem, the overhead time t_2 and memory consumption could be very large. To alleviate this difficulty, we adopt the randomized SVD algorithm [20] to directly calculate the KL expansion of the stochastic solution. This avoids the need to form the covariance matrix and solve the expensive eigenvalue problem. This approach significantly reduces the computational cost and memory consumption in the offline stage. As we will show in section 4, the offline computational cost in constructing the KL expansion is negligibly small compared with the overall offline computational cost.

To further reduce the overhead time in DSM, we propose a greedy-type algorithm combined with a two-level preconditioning [13] to enrich our data-driven stochastic basis. First, we derive an error equation for the stochastic solution obtained by the most recently enriched basis. We solve the error equation for each trial function $f_k(x)$, $k = 1, 2, \ldots, K$, only *on the coarse grid*, and we identify the maximum error τ_{k^*} along with the corresponding trial function f_{k^*}. The error equation for this trial function is solved again on the fine grid. The KL expansion of the residual error is then used to enrich the stochastic basis. This process is repeated until the maximum residual error is below the prescribed threshold ϵ. The two cost-saving measures described above play an important role in reducing the overhead time t_2. We find that n_c is typically quite small. See section 4 for more discussions.

The rest of the paper is organized as follows. In section 2, we give some preliminary introductions about the KL expansion and gPC basis. In section 3, we provide the detailed derivation of DSM. In addition, we will describe our error analysis of the stochastic basis and propose an optimization approach to enrich the stochastic basis. The error correction of the method will also be discussed. A computational time model is built in section 4 to show the computational complexities of different methods. In section 5, we apply our method to both the 1D and 2D elliptic PDEs with random elliptic coefficients to demonstrate its computational efficiency. Finally, some concluding remarks are given in section 6.

2. Some preliminaries.

2.1. The Karhunen–Loève expansion. In the theory of stochastic processes, the KL expansion [22, 25] is a representation of a stochastic process as an infinite linear combination of orthogonal functions, analogous to a Fourier series representation of a function on a bounded interval. The importance of the KL expansion is that it yields an optimal basis in the sense that it minimizes the total mean square error.

Consider a probability space (Ω, F, P), whose event space is Ω and is equipped with σ-

algebra F and probability measure P. Suppose $u(x,\omega)$, defined on a compact spatial domain $D \subseteq R^d$, is a second-order stochastic process, i.e., $u(x,\omega) \in L^2(D \times \Omega)$. Its KL expansion reads as

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

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

$$\int_D C(x,y)\phi(y)\,\mathrm{d}y = \lambda\phi(x). \tag{2}$$

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

$$C(x,y) = \mathbb{E}\left[(u(x,\omega) - \bar{u}(x))(u(y,\omega) - \bar{u}(y))\right]. \tag{3}$$

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

$$\xi_i(\omega) = \frac{1}{\sqrt{\lambda_i}} \int_D (u(x,\omega) - \bar{u}(x))\phi_i(x)\,\mathrm{d}x. \tag{4}$$

Moreover, these random variables $\{\xi_i(\omega)\}$ are of zero mean and are uncorrelated; i.e., $\mathbb{E}[\xi_i] = 0$, $\mathbb{E}[\xi_i\xi_j] = \delta_{ij}$. Generally, the eigenvalues λ_i's are sorted in *descending* order. Their decay rates depend on the regularity of the covariance kernel $C(x,y)$. It has been proven that an algebraic decay rate, i.e., $\lambda_k = O(k^{-\gamma})$, is achieved asymptotically if the covariance kernel is of finite Sobolev regularity or an exponential decay, i.e., $\lambda_k = O(e^{-\gamma k})$ for some $\gamma > 0$, if the covariance kernel is piecewise analytic [36]. In general, the decay rate depends on the correlation length of the stochastic solution. Small correlation length results in slow decay of the eigenvalues. In any case, an m-term truncated KL expansion converges in $L^2(D \times \Omega)$ to the original stochastic process $u(x,\omega)$ as m tends to infinity. If we denote by ϵ_m the truncation error, we have

$$||\epsilon_m||^2_{L^2(D\times\Omega)} = \left|\left| \sum_{i=m+1}^{\infty} \sqrt{\lambda_i}\xi_i(\omega)\phi_i(x) \right|\right|^2_{L^2(D\times\Omega)} = \sum_{i=m+1}^{\infty} \lambda_i \to 0, \quad m \to \infty, \tag{5}$$

where we have used the biorthogonality of the KL expansion.

In practical computations, we truncate the KL expansion into its first m terms and obtain the following truncated KL expansion:

$$u(x,\omega) \approx \bar{u}(x) + \sum_{i=1}^{m} \sqrt{\lambda_i}\xi_i(\omega)\phi_i(x). \tag{6}$$

The truncation error analysis in (5) reveals the most important property of KL expansion. More specifically, given any integer m and orthonormal basis $\{\psi_i(x)\}_{i=1}^{m}$, we may approximate the stochastic process $u(x,\omega)$ by

$$u_{m,\psi}(x,\omega) = \bar{u}(x) + \sum_{i=1}^{m} \zeta_i(\omega)\psi_i(x), \tag{7}$$

where $\zeta_i(\omega)$, $i = 1, \ldots, m$, are the expansion coefficients. Among all m-term approximations using an orthonormal basis, the KL expansion given by (6) is the one that minimizes the total mean square error. In this sense, we say that the KL expansion gives the optimal (or the most compact) basis to represent the stochastic solution in the energy norm. Due to the biorthogonality of the KL expansion, we will call the stochastic part of the KL expansion the *data-driven basis* in the rest part of this paper.

Remark 2.1. It is important to note that if the correlation length of the solution is small, then the number of expansion terms m may be large due to the strong correlation of the stochastic solution. In this case, the DSM based on the KL expansion is not an optimal choice, although it still has some advantages over the Monte Carlo method or a stochastic spectral method. To develop a more effective DSM for stochastic solutions with small correlation length, we need to develop a multiscale version of the DSM. This will be investigated in a subsequent paper.

2.2. The generalized polynomial chaos (gPC) basis. In many physical and engineering problems, randomness generally comes from various independent sources, so randomness in SPDE (1) is often given in terms of independent random variables. We assume that the randomness in the differential operator $\mathcal{L}(x, \omega)$ is given in terms of r independent random variables, i.e., $\boldsymbol{\xi}(\omega) = (\xi_1(\omega), \xi_2(\omega), \ldots, \xi_r(\omega))$. Without loss of generality, we can further assume that such independent random variables have the same distribution function $\rho(x)$. We get $\mathcal{L}(x, \omega) = \mathcal{L}(x, \xi_1(\omega), \ldots, \xi_r(\omega))$. By the Doob–Dynkin lemma [32], the solution of (1) can still be represented by these random variables, i.e., $u(x, \omega) = u(x, \xi_1(\omega), \ldots, \xi_r(\omega))$.

Let $\{\mathrm{H}_i(\xi)\}_{i=1}^{\infty}$ denote the 1D $\rho(\xi)$-orthogonal polynomials, i.e.,

$$\int_{\Omega} \mathrm{H}_i(\xi)\mathrm{H}_j(\xi)\rho(\xi)d\xi = \delta_{ij}.$$

For some commonly used distributions, such as the Gaussian distribution and the uniform distribution, such orthogonal polynomial sets are Hermite polynomials and Legendre polynomials, respectively. For general distributions, such polynomial sets can be obtained by numerical methods [37]. Furthermore, by a tensor product representation, we can use the 1D polynomial $\mathrm{H}_i(\xi)$ to construct a sufficient orthonormal basis $\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ of $L^2(\Omega)$ as follows:

$$\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \prod_{i=1}^{r} \mathrm{H}_{\alpha_i}(\xi_i), \quad \boldsymbol{\alpha} \in \mathfrak{J}_r^{\infty}, \tag{8}$$

where $\boldsymbol{\alpha}$ is a multi-index and $\mathfrak{J}_r^{\infty}$ is a multi-index set of countable cardinality,

$$\mathfrak{J}_r^{\infty} = \{\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_r) \mid \alpha_i \geq 0, \alpha_i \in \mathbb{N}\}.$$

The zero multi-index corresponding to $\mathbf{H}_{\mathbf{0}}(\boldsymbol{\xi}) = 1$ is used to represent the mean of the solution. Clearly, the cardinality of $\mathfrak{J}_r^{\infty}$ is infinite. For the purpose of numerical computations, we prefer a finite set of polynomials. There are many choices of truncations. One possible choice is the set of polynomials whose total orders are at most p, i.e.,

$$\mathfrak{J}_r^{p} = \left\{ \boldsymbol{\alpha} \,|\, \boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_r), \alpha_i \geq 0, \alpha_i \in \mathbb{N}, |\alpha| = \sum_{i=1}^{r} \alpha_i \leq p \right\}. \tag{9}$$

The cardinality of $\mathfrak{J}_r^p$ in (9) or the number of polynomial basis functions, denoted by $N_p = |\mathfrak{J}_r^p|$, is equal to $\frac{(p+r)!}{p!r!}$. Another good choice is the sparse truncation method proposed in Luo's thesis [26]. We may simply write such a truncated set as $\mathfrak{J}$ when no ambiguity arises. The orthonormal basis $\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ is the standard gPC basis; see [16, 40, 21] for more details.

2.3. Stochastic elliptic equations with low-dimensional structure. To illustrate our ideas, we consider the DSM for the elliptic equation with a homogeneous Dirichlet boundary condition, i.e.,

$$-\frac{\partial}{\partial x_p}\left(a_{pq}(x,\omega)\frac{\partial}{\partial x_q}u(x,\omega)\right) = f(x), \qquad x \in \mathcal{D} \subset \mathbb{R}^d,\ \omega \in \Omega, \tag{10}$$

$$u(x,\omega) = 0, \qquad x \in \partial\mathcal{D},\ \omega \in \Omega, \tag{11}$$

where $p, q = 1, \ldots, d$ and Einstein summation is assumed. Equation (10) arises in many physical and engineering fields. For example, this equation can be used to model the flow and transport in natural porous media such as water aquifer and oil reservoirs [24, 14, 13, 39], where the permeability field $\big(a_{pq}(x,\omega)\big)$ is a stochastic process whose exact values are infeasible to obtain in practice due to the low resolution of seismic data.

The simplest approach to a numerical solution of (10) is the Monte Carlo method. To solve (10) by the Monte Carlo method, we need to generate numerous samples of the permeability field $\big(a_{pq}(x,\omega)\big)$ with a prescribed distribution, solve (10) for each sample, and determine the statistics of $u(x,\omega)$ from this ensemble of solutions. Due to the slow convergence of the Monte Carlo method, this approach requires a large number of samples.

Further improvements on convergence rate can be achieved by exploring certain structures of the stochastic solutions. The stochastic spectral finite element method (SSFEM) [16] and the gPC method [40, 41] explore certain smoothness of stochastic solutions with respect to random variables and use a set of orthogonal polynomials to represent the solution. The solution generally takes the form of $u(x,\omega) = \sum_{\boldsymbol{\alpha}\in\mathfrak{J}} u_{\boldsymbol{\alpha}}(x)\mathbf{H}_{\boldsymbol{\alpha}}(\omega)$, where $\mathfrak{J}$ is some multi-index set and $\mathbf{H}_{\boldsymbol{\alpha}}(\omega)$ is a set of orthogonal polynomials. There has been considerable progress in developing efficient numerical methods to solve (10) in the past two decades, such as the stochastic collocation (SC) method [1]. However, the SSFEM, gPC, and SC methods suffer from the curse of dimensionality. They use a stochastic basis that is problem independent. Such a feature is attractive in the sense that it makes these methods very general. However, the use of the problem-independent basis is also the reason that they do not give a compact representation of the stochastic solution. Constructing a problem-dependent or data-driven stochastic basis is essential in exploiting the data-sparsity structure of the stochastic solution to design more efficient numerical algorithms. The sparsity that we explore here is not the usual entrywise sparsity, i.e., only a few of nonzero entries or coefficients, but the data-sparsity, i.e., a few data are required to provide an accurate description of the stochastic solutions. For more discussions on data-sparsity, we refer the reader to [18, 19].

Let us first consider the 1D stochastic elliptic equation with a homogeneous Dirichlet

boundary condition as follows:

$$-\frac{\partial}{\partial x}\left(a(x,\omega)\frac{\partial}{\partial x}u(x,\omega)\right)=f(x),\quad x\in(0,1),\ \omega\in\Omega, \tag{12}$$

$$u(0,\omega)=0,\quad u(1,\omega)=0, \tag{13}$$

where the random field $a(x,\omega)$ satisfies the ellipticity, $0<a_1\le a(x,\omega)\le a_2<\infty$, almost surely. After some simple calculations, we obtain its analytic solution,

$$u(x,\omega)=-\int_0^x\frac{ds_2}{a(s_2,\omega)}\int_0^{s_2}f(s_1)ds_1+C(\omega)\int_0^x\frac{ds}{a(s,\omega)}, \tag{14}$$

where $C(\omega)$ is determined by the boundary condition, given by

$$C(\omega)=\frac{\int_0^1\frac{ds_2}{a(s_2,\omega)}\int_0^{s_2}f(s_1)ds_1}{\int_0^1\frac{ds}{a(s,\omega)}}.$$

In addition, we denote $b(x,\omega)=\frac{1}{a(x,\omega)}$ and assume $b(x,\omega)$ has an exact M-term KL expansion

$$b(x,\omega)=\sum_{i=0}^{M}b_i(x)B_i(\omega), \tag{15}$$

where $b_0(x)=\bar{b}(x)$ and $B_0(\omega)=1$. Substituting the KL expansion (15) into the solution (14), we obtain

$$\begin{aligned}u(x,\omega)=&-\sum_{i=0}^{M}B_i(\omega)\int_0^x b_i(s_2)ds_2\int_0^{s_2}f(s_1)ds_1\\&+\sum_{i,j=0}^{M}\frac{B_i(\omega)B_j(\omega)}{\int_0^1 b(s,\omega)ds}\left(\int_0^1 b_j(s_2)ds_2\int_0^{s_2}f(s_1)ds_1\right)\int_0^x b_i(s)ds.\end{aligned} \tag{16}$$

Apparently, the solution expression (16) reveals that solution of the 1D stochastic elliptic equation (12) lies in the space spanned by the stochastic basis $\{B_i(\omega)\}_{i=1,\dots,M}\bigcup\{\frac{B_i(\omega)B_j(\omega)}{\int_0^1 b(s,\omega)ds}\}_{i,j=0,\dots,M}$. If the covariance kernel of the random field $b(x,\omega)=\frac{1}{a(x,\omega)}$ is smooth enough, the number of effective terms in the KL expansion of $b(x,\omega)$ will be small, and the solution has a low-dimensional structure independent of the forcing function f.

For the stochastic elliptic equation (10) in two dimensions, we cannot obtain an explicit solution expression. Our numerical study seems to indicate that similar results still hold for (10) with a certain class of coefficient $a(x,\omega)$. The above analysis may shed light on exploring the data-sparsity of the solutions of SPDEs and developing an effective numerical method.

3. A data-driven stochastic method.

3.1. General framework of the data-driven stochastic method. The central task of our DSM is to look for a data-driven stochastic basis under which the solution of an SPDE enjoys a compact expansion. Clearly, such a stochastic basis should be constructed through learning some information about the stochastic solution. To obtain useful information and grasp physical insights of the system involving randomness, certain postprocessing of the stochastic solution is necessary. Due to its error-minimizing property, the KL expansion is a natural choice for postprocessing of the solution and constructing a problem-dependent stochastic basis.

We first outline the general framework of our DSM, which consists of offline and online stages. In the offline stage, we propose an effective strategy to construct a data-driven basis $\{A_i(\omega)\}_{i=0}^m$, where $A_0(\omega) = 1$ and m is the number of elements in the basis. Our method is a greedy-type algorithm combined with a two-level preconditioning [13] to reduce the offline computational cost. Once the data-driven basis is constructed, we can use them in the standard Galerkin method to solve the SPDEs (1) in the online stage. Specifically, we expand the stochastic solution in terms of this stochastic basis $u(x,\omega) = \sum_{i=0}^m u_i(x)A_i(\omega)$ and solve a coupled system of PDEs for the deterministic coefficients, $\{u_i(x)\}_{i=0}^m$. Since the online stage is pretty straightforward, we state only the offline computation algorithm as follows; see also Figure 1 for an illustration of the main ideas.

DSM offline computation.

- Step 0 (preparation):
  - Set the error threshold ϵ_0; partition spatial domain D into a fine grid D_h and a coarse grid D_H.
  - Approximate $f(x,\theta)$ by a finite dimensional basis $\{f_k(x)\}_{k=0}^K$, i.e., $f(x,\theta) \approx \sum_{k=0}^K c_k(\theta)f_k(x)$.
- Step 1 (initial learning step on the fine grid D_h):
  - Solve (1) with $f_0(x)$ as a forcing function to obtain $u(x,\omega;f_0)$.
  - Calculate the truncated KL expansion of $u(x,\omega;f_0)$ and use the first m_1 terms of the stochastic modes to obtain the current data-driven basis $\{A_i(\omega)\}_{i=0}^{m_1}$, where $A_0(\omega) = 1$.
- Step 2 (preconditioning on the coarse grid D_H):
  - For each trial function $f_k(x)$, solve (1) on the coarse grid D_H using the current stochastic basis $\{A_i(\omega)\}_{i=0}^{m_1}$ and the stochastic Galerkin method to obtain DSM solution $u_{DSM}(x,\omega;f_k)$.
  - For each trial function $f_k(x)$, solve a residual error equation on the coarse grid D_H to obtain the approximate residual error $\tau_k = \tau(x,\omega;f_k)$.
  - If $\max_{1\le k\le K} ||\tau_k|| < \epsilon_0$, goto step 4; else set $k^* = \arg\max_{0\le k\le K} ||\tau_k||$ and $f_{k^*}(x)$, and goto step 3.
- Step 3 (update on fine grid D_h):
  - Solve the residual equation associated with $f_{k^*}(x)$ to obtain the residual error $\tau_{k^*} = \tau(x,\omega;f_{k^*})$.
  - Enrich the current stochastic basis $\{A_i(\omega)\}_{i=0}^{m_1}$ by the KL expansion of τ_{k^*}, and use $\{A_i(\omega)\}_{i=0}^{m_2}$ to denote the updated stochastic basis. Goto step 2.

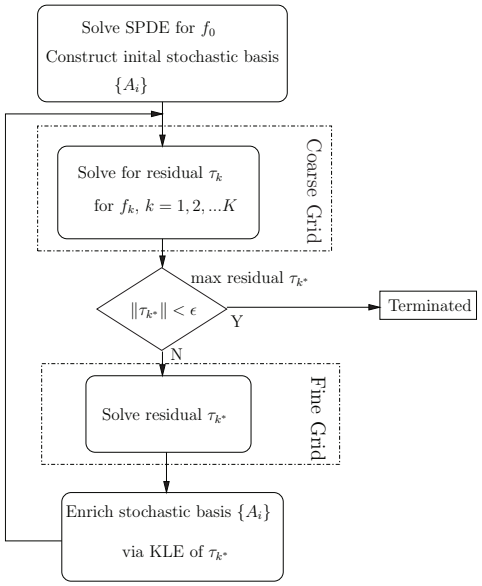


Figure 1. *Greedy stochastic basis enriching algorithm on a coarse-fine grid hierarchy.*

- Step 4 (termination):
  - Save the data-driven stochastic basis, denoted by $\{A_i(\omega)\}_{i=0}^m$ and relevant statistical quantities.

The detailed implementation of this greedy-type algorithm depends on the numerical representation of the stochastic basis, which will be elaborated at length in the next three sections. In this paper, we discuss three ways to represent the stochastic basis:

- Ensemble representation, i.e., a sampling method, such as the Monte Carlo method or the quasi Monte Carlo method.
- Stochastic collocation representation, such as the sparse grid based stochastic collocation (SC) basis.
- Spectral representation, such as the gPC basis.

3.2. Data-driven stochastic basis via an ensemble representation. In this section, we introduce our DSM via an ensemble representation, i.e., Monte Carlo samples. We consider the following elliptic SPDE:

$$-\nabla \cdot (a(x,\omega)\nabla u(x,\omega)) = f(x,\theta), \quad x \in D,\ \omega \in \Omega, \tag{17}$$

$$u(x,\omega) = 0, \quad x \in \partial D, \tag{18}$$

where the coefficient $a(x,\omega)$ is assumed to be positive with upper and lower bounds almost surely. The forcing function $f(x,\theta)$ is approximated by a finite basis, $\{f_k(x)\}_{i=0}^K$, i.e., $f(x,\theta) = \sum_{k=0}^K c_k(\theta) f_k(x)$.

In the *initial learning step* of our DSM, we first use the Monte Carlo method to generate N samples of the random coefficient $\{a(x,\omega)\}$, $\omega \in \Omega_N = \{\omega_n\}_{n=1}^N$, and solve (17)–(18) with $f_0(x)$ as the right-hand side to obtain $\{u(x,\omega_n;f_0)\}_{n=1}^N$, abbreviated $u(x,\omega;f_0)$. The m_1-term KL expansion of the Monte Carlo solution $u(x,\omega;f_0)$ gives the dominant components

in the random space. We use the decaying property of eigenvalues to select parameter m_1, i.e., to select m_1 elements from the stochastic basis such that $\lambda_{m_1+1}/\lambda_1$ is smaller than some predefined threshold, say, 10^{-4}. We denote the truncated KL expansion as

$$u(x,\omega;f_0) \approx \bar{u}(x;f_0) + \sum_{i=1}^{m_1} \sqrt{\lambda_i} A_i(\omega)\phi_i(x;f_0). \tag{19}$$

We call the stochastic basis $\{A_i(\omega)\}_{i=0}^{m_1}$ in (19) the data-driven stochastic basis in ensemble representation, where $A_0(\omega) = 1$.

In general, the stochastic basis constructed by using f_0 may not be adequate to give an accurate approximation of the SPDE for another right-hand side, $f(x,\theta)$. We need to supplement the stochastic basis by using multiple trial functions involving other f_k.

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

$$u(x,\omega;f_1) \approx \bar{u}(x;f_1) + \sum_{i=1}^{m_1} A_i(\omega)u_i(x;f_1) \equiv \sum_{i=0}^{m_1} A_i(\omega)u_i(x;f_1). \tag{20}$$

In the rest of this subsection, we also use $u_i(x) \equiv u_i(x;f_1)$ for simplification. We use the standard stochastic Galerkin method to obtain the coefficient $u_i(x)$. Specifically, we substitute the expansion (20) into the SPDE (17), multiply both sides by $A_j(\omega)$, and take expectations. This gives rise to a coupled PDE system for the expansion coefficient $u_i(x)$,

$$-\nabla\cdot(E[aA_iA_j]\nabla u_i) = f_1(x)E[A_j], \quad x\in D,\ j=0,1,\dots,m_1, \tag{21}$$

$$u_i(x) = 0, \quad x\in\partial D, \tag{22}$$

where Einstein summation is assumed. Solving the coupled deterministic PDE system (21)–(22) by numerical methods, such as the finite element method (FEM) or the finite difference method (FDM), we obtain the expansion coefficient $\{u_i(x)\}_{i=0}^{m_1}$ and thus the approximate solution for $u(x,\omega;f_1)$ in (20). We know that the exact solution can be written as

$$u(x,\omega;f_1) = \sum_{i=0}^{m_1} A_i(\omega)u_i(x;f_1) + \tau(x,\omega;f_1), \tag{23}$$

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

$$-\nabla\cdot(a(x,\omega)\nabla\tau(x,\omega;f_1)) = f_1(x) + \sum_{i=0}^{m_1} \nabla\cdot(a(x,\omega)A_i(\omega)\nabla u_i(x)). \tag{24}$$

To check the completeness of the stochastic basis, we solve the residual (24) on a coarse grid for each $f_k(x)$ $(k=1,\dots,K)$ and obtain the error $\{\tau(x,\omega;f_k)\}_{i=k}^{K}$. If $\max_{1\le k\le K}||\tau(x,\omega;f_k)|| < \epsilon_0$, then this stochastic basis is considered complete. Here $||\cdot||$ can be the $L^2(D\times\Omega)$ norm

of the variance of the stochastic solution. If this is not the case, we identify the maximum error $\tau_{k^*} = \max_{1\le k\le K} ||\tau(x,\omega;f_k)||$ along with the corresponding trial function $f_{k^*}(x)$. We then solve the residual (24) for this trial function $f_{k^*}(x)$ again on a fine grid. We perform the KL expansion for the residual solution $\tau(x,\omega;f_{k^*})$, extract several dominant components in the random space, and supplement them to the current stochastic basis. We use $\{A_i(\omega)\}_{i=0}^{m_2}$ to denote the updated stochastic basis. This process is repeated until the maximum residual is below the prescribed threshold ϵ_0. We save this data-driven stochastic basis, denoted by $\{A_i(\omega)\}_{i=0}^{m}$ and relevant statistical quantities.

In *the online stage*, for each query $f(x,\theta)$, with our data-driven stochastic basis $\{A_i(\omega)\}_{i=0}^{m}$, we use the standard stochastic Galerkin method to solve the SPDEs (17)–(18). The construction of the stochastic basis could be expensive. However, once the stochastic basis is constructed, it can be used repeatedly for different right-hand side functions $f(x,\theta)$ in the online stage. In the multiquery scenario, our DSM could offer considerable computational savings over the Monte Carlo method when the number of queries is large. We will demonstrate this through numerical examples in section 5.

Remark 3.1. In the offline stage, we need to save the realization of the stochastic basis $\{A_i(\omega)\}_{i=0}^{m}$, $\omega \in \Omega_N = \{\omega_n\}_{n=1}^{N}$. This would enable us to solve for the residual equation (24) and update the stochastic basis. In the online stage, the Galerkin projection reduces the SPDEs (17)–(18) to a coupled system of PDEs (21)–(22) whose coefficients involve only $E[aA_iA_j]$, which can be stored in the offline stage. We can save other quantities such as $E[A_iA_jA_k]$ if we want to calculate the high-order moment of the stochastic solution.

3.3. Data-driven stochastic basis via a collocation representation. In the ensemble representation version of the DSM, the Monte Carlo method may introduce a relatively large sampling error, especially in computing expectations of high-order terms. For instance, the term $E[aA_iA_j]$ is calculated by

$$E[aA_iA_j] = \frac{1}{N}\sum_{n=1}^{N} a(x,\omega_n)A_i(\omega_n)A_j(\omega_n). \tag{25}$$

We need a large number of Monte Carlo samples to obtain an accurate result in (25). The sampling error will be carried over to the online computation of the DSM and may lead to a relatively large error. In this section, we consider expanding the data-driven stochastic basis in a certain basis to alleviate this difficulty.

3.3.1. Stochastic collocation representation. In order to perform residual error correction and improve the accuracy, we would like to expand the stochastic basis $A_i(\omega)$ in a certain stochastic basis, i.e.,

$$A_i(\omega) = \sum_{\boldsymbol{\alpha}} A_{\boldsymbol{\alpha}i}\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)), \tag{26}$$

where $\{\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega))\}$ is a gPC basis in the stochastic space and $A_{\boldsymbol{\alpha}i}$ is the expansion coefficient. Moreover, we expand the random coefficient $a(x,\omega)$ in the same basis,

$$a(x,\omega) = \sum_{\boldsymbol{\alpha}} a_{\boldsymbol{\alpha}}(x)\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)), \tag{27}$$

where $a_{\boldsymbol{\alpha}}(x)$ is the expansion coefficient. Then the term $E[aA_iA_j]$ can be calculated as follows:

$$E[aA_iA_j] = a_{\boldsymbol{\alpha}}(x)A_{\boldsymbol{\beta}i}A_{\boldsymbol{\gamma}j}E[\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega))\mathbf{H}_{\boldsymbol{\beta}}(\boldsymbol{\xi}(\omega))\mathbf{H}_{\boldsymbol{\gamma}}(\boldsymbol{\xi}(\omega))], \tag{28}$$

where Einstein summation is assumed. In practical computations, the expansion coefficients $A_{\boldsymbol{\alpha}i}$ in (26), $a_{\boldsymbol{\alpha}}(x)$ in (27), and $E[\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega))\mathbf{H}_{\boldsymbol{\beta}}(\boldsymbol{\xi}(\omega))\mathbf{H}_{\boldsymbol{\gamma}}(\boldsymbol{\xi}(\omega))]$ in (28) can be approximated with high accuracy by quadrature rules. If the input random variables have a modest dimension and the stochastic solution is smooth, the sparse grid based quadrature rule works quite effectively; see [1, 4, 42] for more details.

In the following, we choose the multi-index of the gPC basis as $\mathfrak{J}_r^p$ (see (9)) and discuss the DSM in the (sparse grid based) stochastic collocation representation. The expansion coefficient $A_{\boldsymbol{\alpha}i}$ is given by

$$A_{\boldsymbol{\alpha}i} = E[A_i(\omega)\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega))] \approx \sum_{j=1}^{J} A_i(\mathbf{z}_j)\mathbf{H}_{\boldsymbol{\alpha}}(\mathbf{z}_j)w_j, \quad \boldsymbol{\alpha} \in \mathfrak{J}_r^p, \tag{29}$$

where $\mathbf{z}_j \in R^r$ and $w_j \in R$ are the sparse grid points and the associated weights, respectively. J is the number of sparse grid points. The terms $a_{\boldsymbol{\alpha}}(x)$ in (27) and $E[\mathbf{H}_{\boldsymbol{\alpha}}(\omega)\mathbf{H}_{\boldsymbol{\beta}}(\omega)\mathbf{H}_{\boldsymbol{\gamma}}(\omega)]$ in (28) can be calculated in the same way as follows:

$$a_{\boldsymbol{\alpha}}(x) = E[a(x,\omega)\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega))] \approx \sum_{j=1}^{J} a(x,\mathbf{z}_j)\mathbf{H}_{\boldsymbol{\alpha}}(\mathbf{z}_j)w_j, \quad \boldsymbol{\alpha} \in \mathfrak{J}_r^p, \tag{30}$$

and

$$E[\mathbf{H}_{\boldsymbol{\alpha}}(\omega)\mathbf{H}_{\boldsymbol{\beta}}(\omega)\mathbf{H}_{\boldsymbol{\gamma}}(\omega)] \approx \sum_{j=1}^{J} \mathbf{H}_{\boldsymbol{\alpha}}(\mathbf{z}_j)\mathbf{H}_{\boldsymbol{\beta}}(\mathbf{z}_j)\mathbf{H}_{\boldsymbol{\gamma}}(\mathbf{z}_j)w_j, \quad \boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma} \in \mathfrak{J}_r^p. \tag{31}$$

We use the N_p-by-m matrix $\mathbf{A}$ to denote the expansion coefficient $A_{\boldsymbol{\alpha}i}$, which is essentially the data-driven stochastic basis in the stochastic collocation representation.

Generally speaking, the data-driven stochastic basis in the collocation representation is the same as that in the ensemble representation. All the methods used in the previous section, such as the *initial learning step* and the *preconditioning and update step*, can be used directly here. The only difference is that instead of solving (17)–(18) with Monte Carlo samples, we solve the same equations with the sparse grid points. In addition, the sample average in calculating expectation is replaced by the quadrature rules based on the sparse grid. This simple change significantly improves the accuracy of the DSM and reduces the computational cost. The performance of this method depends on the regularity of the stochastic solution. When the solution of the SPDE is sufficiently smooth, the DSM in the collocation representation is very efficient.

3.3.2. A randomized SVD approach. In the offline stage, we need to calculate the KL expansion of the stochastic solutions. This requires us to solve for the eigenvalues and eigenvectors of the covariance kernel $C(x,y)$ and to project the stochastic solution onto the eigenvectors. The covariance kernel $C(x,y)$ is a function whose dimensionality is twice that of the

physical space of the solution $u(x,\omega)$. Thus it is very expensive to compute the eigenvalues and eigenvectors of the covariance kernel $C(x,y)$. Thanks to the recently developed randomized algorithms for large-scale linear algebra [20], we can use the randomized SVD algorithm to directly calculate the KL expansion of the stochastic solution. This avoids the need to form the covariance kernel and to solve the expensive eigenvalue problem. Below, we give a brief introduction to this method. To simplify the notation, we assume the solution has zero mean.

First, we solve (17)–(18) with the random variable evaluated at the sparse grid points

$$-\nabla \cdot (a(x,\mathbf{z}_j)\nabla u(x,\mathbf{z}_j)) = f(x,\theta), \quad x \in D, j = 1,\dots,J, \tag{32}$$

$$u(x,\mathbf{z}_j) = 0, \quad x \in \partial D. \tag{33}$$

Let X^h be a spatial finite element approximation space of dimension N_h. For each $\mathbf{z}_j$, we use an FEM to solve (32)–(33) and denote $u^h(\mathbf{z}_j)$ as the finite element solution associated with $\mathbf{z}_j$. Collecting all the solutions of (32)–(33) with respect to all $\mathbf{z}_j$, $j = 1,\dots,J$, together, we define a solution set that consists of all the solutions

$$S = \{u^h(\mathbf{z}_j), j = 1,\dots,J\}.$$

The matrix form of the solution set is denoted by $\mathbf{U} = [\mathbf{U}_1,\dots,\mathbf{U}_J] \in R^{N_h \times J}$; i.e., each column of $\mathbf{U}$ is the vector of nodal point values of a finite element solution associated to a sparse grid point $\mathbf{z}_j$. The covariance matrix $\mathbf{C} \in R^{N_h \times N_h}$ is given by

$$\mathbf{C} = \sum_{i=1}^{J} \mathbf{U}_i \mathbf{U}_i^T w_i, \tag{34}$$

where the w_i's are the associated weights of the sparse grid. One can solve the eigenvalue problem for $\mathbf{C}$ and obtain the KL expansion for $\mathbf{U}$. However, this approach is expensive, or even infeasible, for high-dimensional problems. To overcome this difficulty, we adopt an equivalent approach to get the KL expansion for $\mathbf{U}$ directly without forming the covariance matrix $\mathbf{C}$. Let matrix $\tilde{U}$ denote the weighted solution set, i.e., $\tilde{U} = [\sqrt{w_1}\mathbf{U}_1,\dots,\sqrt{w_J}\mathbf{U}_J] \in R^{N_h \times J}$. It should be noted that when w_j is negative, the j column of $\tilde{U}$ is a pure imaginary vector. Actually, the eigendecomposition for $\mathbf{C}$ and the SVD decomposition for $\tilde{U}$ are closely related. We have made some minor modifications to the randomized SVD algorithm [20]. The resulting algorithm is summarized in Algorithm 1, where k is equal to the target KL expansion mode number m plus an oversampling number p (usually $p = 5$ or 10 is sufficient; see [20] for more details).

We can easily see that $\tilde{U}$'s left-singular vectors and the square of $\tilde{U}$'s singular values are the eigenvectors and eigenvalues of the covariance matrix $\mathbf{C}$, respectively. The memory consumption of $\mathbf{C}$ is proportional to $O(N_h^2)$, and the computational cost of obtaining the first m eigenpairs of $\mathbf{C}$ by the direct SVD algorithm is proportional to $O(N_h^2 m)$. On the other hand, the randomized SVD algorithm works with the matrix $\tilde{U}$ directly instead of the covariance matrix C. The memory consumption of the randomized SVD for $\tilde{U}$ is proportional to $O(N_h J)$, and the computational cost of obtaining the first m left-singular vectors and singular values is proportional to $O(N_h J \log(m) + m^2(N_h + J))$. Therefore, our randomized

Algorithm 1. The randomized SVD for the weighted snapshots set $\tilde{U} \approx U \sum V^T$.

1: Draw a $J \times k$ Gaussian random matrix Ω. If the weight w_j corresponding to the jth sparse grid point is negative, multiply the jth row of the matrix Ω by the imaginary unit $i = \sqrt{-1}$. Let $\tilde{\Omega}$ denote the modified random matrix.
2: Form the $N_h \times k$ sample matrix $Y = \tilde{U}\tilde{\Omega}$.
3: Form an $N_h \times k$ orthonormal matrix Q such that $Y = QR$.
4: Form the $k \times J$ matrix $B = Q^T\tilde{U}$.
5: Compute the SVD of the small matrix B: $B = \hat{U} \sum V^T$.
6: Form the matrix $U = Q\hat{U}$.

SVD approach significantly reduces the memory consumption and computational cost in the offline stage, especially when the dimension of the physical space is high, i.e., $N_h \gg J$.

Remark 3.2. When the stochastic solutions data $\tilde{U}$ do not fit into the memory (RAM), we can generate the sample matrix $Y = \tilde{U}\tilde{\Omega}$ in a sequential way; see Algorithm 2. Now the only requirement is that matrices of size $J \times k$ and $N_h \times k$ must be stored in RAM. This approach significantly reduces the memory consumption.

Algorithm 2. A sequential way to generate sample matrix $Y = \tilde{U}\tilde{\Omega}$.

1: Let $\tilde{\Omega}$ denote the $J \times k$ modified random matrix and Y denote the $N_h \times k$ zero matrix. Let $\tilde{U}_{:,j}$ denote the jth column of $\tilde{U}$.
2: **for** $j = 1 \to J$ **do**
3: $\quad Y = Y + [\tilde{U}_{:,j}\tilde{\Omega}_{j1}, \tilde{U}_{:,j}\tilde{\Omega}_{j2}, \ldots, \tilde{U}_{:,j}\tilde{\Omega}_{jk}]$
4: **end for**

3.4. Data-driven stochastic basis via a spectral representation. In this section, we discuss the data-driven stochastic basis via a spectral representation, such as the polynomial chaos basis. We still consider the SPDEs (17)–(18) as an example. If the coefficient $a(x,\omega)$ is given in terms of r independent random variables, i.e., $a(x,\omega) = a(x,\boldsymbol{\xi}(\omega)) = a(x,\xi_1(\omega),\ldots,\xi_r(\omega))$, the solution of (17) can be represented by these random variables, i.e., $u(x,\omega) = u(x,\xi_1(\omega),\ldots,\xi_r(\omega))$. To simplify notation, we assume that the solution $u(x,\omega)$ has zero mean. By the Cameron–Martin theorem, we know that the solution to (17) admits a generalized polynomial chaos expansion,

$$u(x,\omega) = \sum_{\boldsymbol{\alpha}\in\mathfrak{J}_r^\infty} v_{\boldsymbol{\alpha}}(x)\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)) \approx \sum_{\boldsymbol{\alpha}\in\mathfrak{J}} v_{\boldsymbol{\alpha}}(x)\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)), \tag{35}$$

where $\mathfrak{J}_r^\infty$ and $\mathfrak{J}$ are the multi-index sets for the polynomial chaos basis defined in section 2.2 (see (9)). If we write the polynomial chaos basis and its expansion coefficient in a vector form

$$\mathbf{H}(\boldsymbol{\xi}) = \left(\mathbf{H}_{\boldsymbol{\alpha}_1}(\boldsymbol{\xi}), \mathbf{H}_{\boldsymbol{\alpha}_2}(\boldsymbol{\xi}), \ldots, \mathbf{H}_{\boldsymbol{\alpha}_{N_p}}(\boldsymbol{\xi})\right)_{\boldsymbol{\alpha}_i\in\mathfrak{J}},$$
$$\mathbf{V}(x) = \left(v_{\boldsymbol{\alpha}_1}(x), v_{\boldsymbol{\alpha}_2}(x), \ldots, v_{\boldsymbol{\alpha}_{N_p}}(x)\right)_{\boldsymbol{\alpha}_i\in\mathfrak{J}},$$

the gPC solution (35) can be compactly written in a vector form

$$u(x,\omega) \approx \mathbf{V}(x)\mathbf{H}(\boldsymbol{\xi})^T. \tag{36}$$

Using the fact that the stochastic basis $\mathbf{H}(\boldsymbol{\xi})$ in the gPC representation (36) is orthonormal, we can derive the KL expansion of the solution without calculating its covariance function, which is given in the appendix. Now, we are ready to present our data-driven stochastic basis via the gPC representation. We assume that the cardinality $|\mathfrak{J}| = N_p$ of the gPC basis $\{\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega))\}_{\boldsymbol{\alpha}\in\mathfrak{J}}$ is large enough, so that the numerical solution obtained by the gPC method can serve as the "exact" solution.

In the *initial learning step* of our DSM, we pick $f_0(x)$ as the right-hand side and use the gPC method to solve (17). Assuming the solution is given by $u(x,\omega;f_0) = \mathbf{V}(x)\mathbf{H}(\boldsymbol{\xi})^T$, we can calculate its m_1-term truncated KL expansion as

$$u(x,\omega) \approx \sum_{i=1}^{m_1}\sum_{\boldsymbol{\alpha}\in\mathfrak{J}} u_i(x)A_{\boldsymbol{\alpha}i}\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)) = \mathbf{U}(x)\mathbf{A}^T\mathbf{H}^T, \tag{37}$$

where $\mathbf{U}(x) = [u_1(x), u_2(x), \ldots, u_{m_1}(x)]$ and $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_{m_1}]$ is an N_p-by-m_1 matrix satisfying $\mathbf{A}^T\mathbf{A} = \mathbf{I}_{m_1}$. The matrix $\mathbf{A}$ is essentially the data-driven stochastic basis in the gPC representation, which has the same form as in the stochastic collocation representation but is obtained in a different way.

In the *preconditioning and update step* of our DSM, we first complement the matrix $\mathbf{A}$ into an N_p-by-N_p orthonormal matrix, i.e., $\mathbf{A}^{gPC} = [\mathbf{A}, \hat{\mathbf{A}}]$. The N_p-by-$(N_p - m_1)$ matrix $\hat{\mathbf{A}}$ is the orthogonal complement of matrix $\mathbf{A}$. Here the N_p-by-N_p orthonormal matrix $\mathbf{A}^{gPC}$ spans the same solution space as the gPC basis.

We use the stochastic basis $\mathbf{A} = (A_{\boldsymbol{\alpha}i})$, $i = 1, \ldots, m_1$, to represent the solution of (17) with another right-hand side function, $f_1(x) = f(x,\theta)$ for some choice of θ,

$$u^{DSM}(x,\omega) = \sum_{i=1}^{m_1}\sum_{\boldsymbol{\alpha}\in\mathfrak{J}} v_i(x)A_{\boldsymbol{\alpha}i}\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)). \tag{38}$$

Substituting the expansion (38) into (17), multiplying both sides by $\sum_{\boldsymbol{\beta}\in\mathfrak{J}} A_{\boldsymbol{\beta}j}\mathbf{H}_{\boldsymbol{\beta}}(\boldsymbol{\xi}(\omega))$, $j = 1, \ldots, m_1$, and taking the expectations, we obtain a coupled PDE system for the expansion coefficient $v_i(x)$,

$$-\nabla\cdot(\mathfrak{T}^A_{ij}(x)\nabla v_i(x)) = f_1(x)E[\mathbf{H}_{\boldsymbol{\beta}}(\boldsymbol{\xi}(\omega))]A_{\boldsymbol{\beta}j}, \quad j = 1, \ldots, m_1, \tag{39}$$

where the tensor $\mathfrak{T}^A_{ij}(x) = \mathfrak{T}^{\mathbf{H}}_{\boldsymbol{\alpha}\boldsymbol{\beta}}(x)A_{\boldsymbol{\alpha}i}A_{\boldsymbol{\beta}j}$, $\mathfrak{T}^{\mathbf{H}}_{\boldsymbol{\alpha}\boldsymbol{\beta}}(x) = E[a(x,\omega)\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega))\mathbf{H}_{\boldsymbol{\beta}}(\boldsymbol{\xi}(\omega))]$ and the Einstein summation is assumed. By solving (39), we can obtain the DSM solution $u^{DSM}(x,\omega)$.

Similarly, we expand the solution using the gPC basis

$$u(x,\omega) = \sum_{i=1}^{N_p}\sum_{\boldsymbol{\alpha}\in\mathfrak{J}} v_i(x)\mathbf{A}^{gPC}_{\boldsymbol{\alpha}i}\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)) \tag{40}$$

or, equivalently,

$$u(x,\omega) = \sum_{i=1}^{m_1} \sum_{\boldsymbol{\alpha}\in\mathfrak{J}} v_i(x)\mathbf{A}_{\boldsymbol{\alpha}i}\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)) + \sum_{j=1}^{N_p-m_1} \sum_{\boldsymbol{\alpha}\in\mathfrak{J}} v_{j+m_1}(x)\hat{\mathbf{A}}_{\boldsymbol{\alpha}j}\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)). \tag{41}$$

In (41) the first part represents the data-driven stochastic solution captured by the current stochastic basis $\mathbf{A}$, and the second part represents the residual error. We substitute the expansion (40) into (17), multiply both sides by $\mathbf{A}^{gPC}_{\boldsymbol{\beta}j}\mathbf{H}_{\boldsymbol{\beta}}$, $j = 1,\dots,N_p$, and take expectations. This gives rise to a coupled PDE system for the expansion coefficient $v_i(x)$ in (40),

$$-\nabla\cdot(\mathfrak{T}^{\mathbf{A}^{gPC}}_{ij}(x)\nabla v_i(x)) = f_1(x)E[\mathbf{H}_{\boldsymbol{\beta}}(\boldsymbol{\xi}(\omega))]A_{\boldsymbol{\beta}j}, \quad j = 1,\dots,N_p, \tag{42}$$

where the tensor $\mathfrak{T}^{\mathbf{A}^{gPC}}_{ij} = \mathfrak{T}^{\mathbf{H}}_{\boldsymbol{\alpha}\boldsymbol{\beta}}(x)\mathbf{A}^{gPC}_{\boldsymbol{\alpha}i}\mathbf{A}^{gPC}_{\boldsymbol{\beta}j}$ and the Einstein summation is assumed. By solving (42), we can get the expansion coefficient $v_i(x)$ in (40) and thus the error of the DSM solution. Let

$$\tau(x,\omega;f_1) = \sum_{j=1}^{N_p-m_1} \sum_{\boldsymbol{\alpha}\in\mathfrak{J}} v_{j+m_1}(x)\hat{\mathbf{A}}_{\boldsymbol{\alpha}j}\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)) \tag{43}$$

denote the error of the DSM. According to (41), the variance of the error $\tau(x,\omega;f_1)$ is given by $\sum_{j=1+m_1}^{N_p} v_j^2(x)$. We can apply the same greedy-type algorithm combined with the two-level preconditioning approach to enrich the stochastic basis. We will omit the details here.

Remark 3.3. One advantage of representing the data-driven basis via a certain spectral basis is that the spectral representation of the data-driven basis gives very accurate approximation to the statistical quantities of interest, such as $E[A_i(\omega)A_j(\omega)A_k(\omega)]$ or $E[a(x,\omega)A_i(\omega)A_j(\omega)]$. By using a spectral representation, we can use the modified Gram–Schmidt process to decompose the gPC solutions to obtain the data-driven basis. The memory consumption and computational cost are relatively small.

Remark 3.4. The data-driven basis obtained by the stochastic collocation (SC) representation and the generalized polynomial chaos (gPC) representation has the same accuracy and computational cost in the online stage if we fix the index of the orthonormal basis $\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$. However, the computational cost in the offline stage is quite different. The gPC method is intrusive in the sense that we need to solve a coupled deterministic PDE system to obtain the expansion coefficients. When the physical degree of freedom N_h and/or the number of polynomial basis N_p is large, the computational cost is very expensive. The SC representation is nonintrusive in the sense that we need to solve a set of uncoupled deterministic equations. Each solution corresponds to a collocation point and has its own weight. From our computational experience, the DSM using the SC representation is computationally more attractive than the DSM using the gPC representation.

3.5. An a posteriori error estimate and error correction in the online stage. In this section, we will perform a posteriori error estimates to quantify the residual error between the data-driven solution and the exact solution. Such error estimates provide us with an adaptive strategy to improve the accuracy of our DSM. Thanks to the spectral structure of

the data-driven basis (both in the SC representation and the gPC representation), we can easily address this issue.

In many applications, we are interested in studying the statistical quantities of the random solution $u(x,\omega)$, such as the mean and the variance. Denote the statistical moments of $u(x,\omega)$ as $E[g(u)]$, where $g(x) = x^n$, $n = 1, 2, \ldots$. To simplify the notation, we suppress the spatial variables in the function $u(x,\omega)$. Suppose the $u(\omega_k)$'s $(k = 1, \ldots, N)$ are realizations of the random solution computed by Monte Carlo simulations. The expectation of $E[g(u)]$ can be approximated by the Monte Carlo ensemble average

$$I_N[g(u)] = \frac{1}{N}\sum_{k=1}^{N} g(u(\omega_k)). \tag{44}$$

Denote the error of the Monte Carlo estimator (44) as

$$\epsilon_g(N) = E[g(u)] - I_N[g(u)]. \tag{45}$$

The error $\epsilon_g(N)$ itself is a random variable. According to the central limit theorem [15], the root mean square error of the Monte Carlo estimator (44) decays like

$$\sqrt{E[\epsilon_g^2(N)]} \simeq \frac{\sigma[g(u)]}{\sqrt{N}}, \tag{46}$$

where $\sigma[g(u)]$ is the standard deviation (STD) of $g(u)$. Thus, the ensemble average (44) converges to $E[g(u)]$ at the rate of $\frac{1}{\sqrt{N}}$ with a proportional constant given by the variance of $g(u)$. This slow convergence rate is an inevitable result due to the central limit theorem. One way to accelerate the convergence of the Monte Carlo ensemble average (44) is to reduce the variance. Using the data-driven stochastic method, we can split $E[g(u)]$ into two parts

$$E[g(u)] = E[g(u_{DSM})] + E[g(u) - g(u_{DSM})], \tag{47}$$

where $E[g(u_{DSM})]$ can be obtained by the DSM and we need only use Monte Carlo simulation to estimate $E[g(u) - g(u_{DSM})]$. As a result, we obtain

$$E[g(u)] \approx E[g(u_{DSM})] + \frac{1}{N}\sum_{k=1}^{N}[g(u(\omega_k)) - g(u_{DSM}(\omega_k))]. \tag{48}$$

The error of the estimation (48) is

$$\epsilon_{g-g_{DSM}}(N) = E[g(u) - g(u_{DSM})] - \frac{1}{N}\sum_{k=1}^{N}[g(u(\omega_k)) - g(u_{DSM}(\omega_k))], \tag{49}$$

According to the central limit theorem, the root mean square error of the estimation (49) is

$$\sqrt{E[\epsilon_{g-g_{DSM}}^2(N)]} \simeq \frac{\sigma[g(u) - g(u_{DSM})]}{\sqrt{N}}, \tag{50}$$

where $\sigma(g(u) - g(u_{DSM}))$ is the STD of $g(u) - g(u_{DSM})$.

If the DSM solution u_{DSM} is a good approximation of the true solution u, then $\sigma(g(u) - g(u_{DSM})) \ll \sigma(g(u))$, and the ensemble average (48) will converge faster than the direct ensemble average (44). Therefore, the DSM provides an effective variance reduction method in the Monte Carlo simulation.

In practical computations, if the dimension of the SPDE solution space is high, we should avoid resolving all the small scales of the stochastic solution by adding more and more elements into the data-driven stochastic basis. We can put an upper limit on the total number of elements in the stochastic basis and stop the update approach with a predefined maximum mode number m_{max}. The solution $u_{DSM}(x,\omega)$ obtained by the DSM with m_{max} modes may not be very accurate, but it has already captured the large-scale structure of the stochastic solution. If we subtract the data-driven solution from the exact solution, the error will be another random variable with a smaller variance. Then we use Monte Carlo simulations to correct the error in the data-driven solution. This *error correction* procedure further improves the accuracy of the DSM.

The error correction procedure can also provide an a posteriori error estimate for the DSM. We consider the a posteriori error estimate of the mean of the solution as an example. The same idea can be applied to compute other moments. Let $r(x,\omega) = u(x,\omega) - u_{DSM}(x,\omega)$ denote the residual error function. According to (47)–(49), the mean of the random solution can be written as

$$\begin{aligned} E[u] &= E[u_{DSM}] + E[r] \\ &= E[u_{DSM}] + \frac{1}{N}\sum_{k=1}^{N} r(x,\omega_k) + E[r] - \frac{1}{N}\sum_{k=1}^{N} r(x,\omega_k). \end{aligned} \tag{51}$$

In (51), $E[u_{DSM}]$ is the mean of the data-driven solution. The term $\frac{1}{N}\sum_{k=1}^{N} r(x,\omega_k)$ is the Monte Carlo ensemble average of the residual error. Due to the central limit theorem, this ensemble average approximates a normal distribution, i.e.,

$$\frac{1}{N}\sum_{k=1}^{N} r(x,\omega_k) \sim \mathcal{N}\left(E[r], \frac{\sigma_r^2}{N}\right). \tag{52}$$

In (52), the mean $E[r]$ and STD σ_r of the residual error are unknown. We use the sample mean and sample STD to approximate them. Let $\bar{r}(x)$ denote the sample mean, i.e., $\bar{r}(x) = \frac{1}{N}\sum_{k=1}^{N} r(x,\omega_k)$, and let $\tilde{r}(x)$ denote the sample STD, i.e.,

$$\tilde{r}(x) = \sqrt{\frac{1}{N}\sum_{k=1}^{N} r^2(x,\omega_k) - \bar{r}^2(x)}. \tag{53}$$

Let $||\cdot||$ define a norm on some function space over the physical space D; for instance, $||\cdot||$ can be the $L^2(D)$ norm. We define the norm of the sample mean and sample STD of the residual error as $\tau_N^1 = ||\bar{r}(x)||$ and $\tau_N^2 = ||\frac{\tilde{r}(x)}{\sqrt{N}}||$, respectively.

We now demonstrate the a posteriori error estimate and error correction algorithm in the online stage. Without loss of generality, we calculate the mean of $u(x,\omega)$. First, we solve (10)–(11) with our DSM to obtain the solution $u_{DSM}(x,\omega)$ and the norm of its expectation $||E[u_{DSM}]||$. Then we generate a small number of samples $\{\omega_k\}_{k=1}^N$ and solve (10)–(11) using the Monte Carlo method to obtain solutions $\{u(x,\omega_k)\}_{k=1}^N$. Substituting the same samples $\{\omega_k\}_{k=1}^N$ into $u_{DSM}(x,\omega)$, we can easily obtain the samples of residual error functions $\{r(x,\omega_k)\}_{k=1}^N$ and calculate $\tau_N^1 = ||\bar{r}(x)||$ and $\tau_N^2 = ||\frac{\tilde{r}(x)}{\sqrt{N}}||$.

The norm of the sample mean $\tau_N^1 = ||\bar{r}(x)||$ gives the magnitude of the residual. The norm $\tau_N^2 = ||\frac{\tilde{r}(x)}{\sqrt{N}}||$ measures the fluctuation of each realization of the residual error function around its mean. We can use τ_N^2 to construct a confidence interval with regard to a predefined confidence level for the residual error. For instance, at each fixed spatial point $x \in D$, a 95%-level confidence interval is approximately given by $[\bar{r}(x) - \frac{2\tilde{r}(x)}{\sqrt{N}}, \bar{r}(x) + \frac{2\tilde{r}(x)}{\sqrt{N}}]$. We perform our error estimate according to the following scenarios (ϵ_1 and ϵ_2 are predefined thresholds).

- Case 1: The ratio $\tau_N^1 + 2\tau_N^2 < \epsilon_1 ||E[u_{DSM}]||$, which means the data-driven solution is accurate and the residual error is negligible.
- Case 2: Case 1 does not hold, but $\tau_N^2 < \epsilon_2 \tau_N^1$, which means that the residual error between the exact solution and the data-driven solution cannot be ignored. However, the fluctuation of the residual error function is small. In this case, $\{r(x,\omega_k)\}_{k=1}^N$ provides a very good error correction for the data-driven solution.
- Case 3: Neither case 1 nor case 2 holds, which means the sample number N is too small. We double the sample number to repeat the Monte Carlo calculation.

In the a posteriori error estimate and error correction framework, the Monte Carlo simulation is used as an error correction step to the data-driven solution. Alternatively, we can also interpret the data-driven solution as a precomputation step to obtain a control variate for Monte Carlo simulations. This hybrid method takes the advantages of both the DSM and the Monte Carlo method, which improves the efficiency and accuracy of the proposed method.

Remark 3.5. If the dimension of the SPDE solution space is low or medium, it is sufficient to use hundreds of Monte Carlo simulations to obtain a good error correction. However, when the dimension of the stochastic solution is large, we need a large number of Monte Carlo simulations. Even in this case, our method still offers some advantage over existing methods, although the computational savings are not as significant as in the case when the effective dimension of the stochastic solution is low or moderate. Moreover, we can adopt various Monte Carlo acceleration techniques to speed up the error correction step, such as the multilevel Monte Carlo method [17].

4. Computational complexity analysis.

The computational time of the DSM consists of both the offline and online parts. The offline computation can be very expensive if we use a brute-force way to construct the data-driven basis. We will show that using the randomized SVD solver and the preconditioning on a coarse grid can significantly reduce this offline computational time. In addition, we construct a time model to demonstrate that the DSM is superior to the traditional methods in a multiquery setting. We focus our discussion on the DSM with the stochastic collocation representation since it is an optimal choice when the stochastic solution is smooth. On the other hand, when the stochastic input dimension is

Table 1
Computational time of forming the stiffness matrix in the gPC or DSM solver. N_b is the basis number.

Grid number	$N_b = 10$	$N_b = 20$	$N_b = 40$
$N_h = 32^2$	0.7531	2.6669	14.2577
$N_h = 64^2$	3.2968	12.8500	63.2856
$N_h = 128^2$	15.6921	59.0485	294.0339

Table 2
Computational time of the (AMD permuted) Cholesky decomposition. (Time: Sec.)

Grid number	$N_b = 5$	$N_b = 10$	$N_b = 20$	$N_b = 40$
$N_h = 32^2$	0.0902	0.3190	1.2929	5.4715
$N_h = 64^2$	0.4552	1.5145	6.3516	27.0306
$N_h = 128^2$	2.0270	7.5956	31.8679	136.3302

high, the DSM with the ensemble representation would perform superiorly to other methods.

4.1. A computational time model in the multiquery setting. In this section, we construct a computational time model for the 2D elliptic SPDEs in the multiquery setting. Let N_h and J denote the number of the physical fine grid points and sparse grid points, respectively. We assume $N_h \gg J$. The fine grid will be chosen as $N_h = 128^2$. All the simulations and comparisons are conducted on a single computing node with 16 GB memory at Caltech Center for Advanced Computing Research (CACR).

4.1.1. The computational time model of gPC solver. Let $t_1 = t_1(N_h, N_p)$ denote the "offline" computation time of the gPC solver. The offline cost t_1 can be approximated by $t_1 \approx t_{1s}(N_h, N_p) + t_{1chol}(N_h, N_p)$, where t_{1s} is the time of generating the stiffness matrix and t_{1chol} is the time for the Cholesky decomposition. In Table 1, we list the computational cost for generating the stiffness matrix. On the fine grid $N_h = 128^2$, t_{1s} is approximately given by

$$t_{1s} \approx 0.1152 N_p^2. \tag{54}$$

We then consider the time for the Cholesky decomposition. Recall that if S is an n-by-n positive definite *dense* matrix, the Cholesky decomposition costs about $\frac{1}{3}n^3$ flops. If S is *sparse*, the cost is much less than $\frac{1}{3}n^3$, and the exact cost depends on the number of nonzero elements, sparsity patterns, etc. In Table 2 we list the time of the (approximate minimum degree (AMD) permuted) Cholesky decomposition. On the grid $N_h = 128^2$, t_{1chol} is approximated by

$$t_{1chol} \approx 0.0745 N_p^2. \tag{55}$$

Therefore, on the fine grid $N_h = 128^2$, the "offline" time gPC solver can be approximated by

$$t_1 \approx 0.1897 N_p^2. \tag{56}$$

In the multiquery setting, the stiffness matrix S for the gPC solver is fixed, and the load vector b is different for each query. We can precompute the Cholesky decomposition of S

Table 3
Computational time of forward/back substitution. (Time: Sec.)

Grid number	$N_b = 5$	$N_b = 10$	$N_b = 20$	$N_b = 40$
$N_h = 32^2$	0.0290	0.0655	0.2013	1.0329
$N_h = 64^2$	0.1358	0.3519	1.4529	7.7249
$N_h = 128^2$	0.7088	2.2714	9.7002	51.7330

in advance, and the computational time is determined only by the forward and backward substitutions in solving the linear equation system. Let t_{fb} denote this time. In Table 3, we list the computation time of t_{fb} for different grid points and basis elements. On the grid $N_h = 128^2$, t_{fb} is approximately given by

$$t_{fb} \approx 0.0223 N_b^2. \tag{57}$$

Let n denote the total number of queries. The computational time of gPC will be

$$t_{gPC}(n) = 0.1897 N_p^2 + 0.0223 N_p^2 n. \tag{58}$$

4.1.2. The computational time model of DSM solver. Let $t_2 = t_2(N_h, K, m_1, dm, m)$ denote the computational time of DSM in the offline stage, where K is the number of trial functions, m_1 is the number of basis elements obtained in the initial learning step, dm is the number of basis elements added each time in the updating on fine grid step, and m is the total number of basis elements obtained in the offline stage, which depends on the prescribed threshold ϵ. Roughly speaking, we need one-time initial learning and $n_{up} = [\frac{m-m_1}{dm}] + 1$ times updating learning. From our experience, we find that m is much smaller than N_p when the effective dimension of the stochastic solution is small.

It is easy to show that t_2 can be approximated by

$$t_2 \approx t_{SC} + t_{Pre} + t_{KLE}, \tag{59}$$

where t_{SC} is the cost of using the stochastic collocation method to solve the SPDE in the initial learning step (step 1) and the residual error equation in the updating on fine grid step (step 3), t_{Pre} is the cost of the preconditioning on the coarse grid step (step 2), and t_{KLE} is the cost of KL expansion in our offline stage. We need to do the KL expansion $1 + n_{up}$ times. In addition, t_{Pre} consists of several parts, i.e.,

$$t_{Pre} \approx t_{Pre-s} + t_{Pre-chol} + t_{Pre-fb}, \tag{60}$$

where t_{Pre-s} is the time spent forming the stiffness matrix on a coarse grid, t_{Pre-c} is the time spent doing the Cholesky decomposition, and t_{Pre-fb} is the time spent solving the linear equation on the coarse grid. We neglect the time spent solving residual error equation, since this is done on a coarse physical grid and low-level sparse grids. We will do a thorough study of all the parts in (65) and (60). We choose fine grid $N_h = 128^2$ and coarse grid $Nc_h = 32^2$ in the following discussion.

In Table 4 we list the computational time of the randomized SVD on different mesh grids. We also show the computational time spent solving the linear equation once using the

Table 4
Computational time of the randomized SVD solver. (Time: Sec.)

Solver	$N_h = 8^2$	$N_h = 16^2$	$N_h = 32^2$	$N_h = 64^2$	$N_h = 128^2$
$rSVD$	0.0009	0.0016	0.0037	0.0129	0.0504

Table 5
Computational time of the linear equation solver on one sparse grid. (Time: Sec.)

$N_h = 8^2$	$N_h = 16^2$	$N_h = 32^2$	$N_h = 64^2$	$N_h = 128^2$
0.0003	0.0013	0.0060	0.0340	0.2455

stochastic collocation method in Table 5 (the same result can be applied for the Monte Carlo solver). One can see that the computational time to perform the KL expansion is even less than that to solve the linear equation once. For instance, on the fine grid $N_h = 128^2$, if we choose sparse grid number $J = 200$, the time ratio of the KL expansion and the stochastic collocation solver in the initial learning stage will be $\frac{0.0504}{200\times 0.2455} = 0.102\%$. Therefore, t_{KLE} is negligible.

On the other hand, it is easy to obtain

$$t_{SC} \approx 0.2455J(1 + n_{up}). \tag{61}$$

Finally, on the coarse grid, we approximate t_{Pre-s}, $t_{Pre-chol}$, and t_{Pre-fb} as follows:

$$t_{Pre-s} \approx 0.0053m^2, \tag{62}$$

$$t_{Pre-chol} \approx 0.0035m^2, \tag{63}$$

and

$$t_{Pre-fb} \approx 0.0029m^{1.4}, \tag{64}$$

where m is the basis number in the DSM. Putting everything together, we obtain the following estimate for t_2:

$$\begin{aligned} t_2 &\approx t_{SC} + t_{Pre} \\ &= 0.2455J(1 + n_{up}) + \sum_{i=1}^{n_{up}} (0.0053m_i^2 + 0.0035m_i^2 + 0.0029Km_i^{1.4}) \\ &\le 0.2455J(1 + n_{up}) + n_{up}(0.0088m^2 + 0.0029Km^{1.4}), \end{aligned} \tag{65}$$

where $m_i = m_{i-1} + dm$ is the number of basis elements in DSM in the ith updating step. Recall (57); the computational time of DSM will be

$$t_{DSM}(n) = t_2 + 0.0223m^2 n. \tag{66}$$

Let m and N_p be the number of basis elements used in DSM and gPC, respectively. We can see that on the same physical grid the ratio of computational complexities between DSM (online stage) and gPC is of order $O((m/N_p)^2)$.

4.1.3. The computational time model of gSC solver. The "offline" cost of gSC is zero. For each forcing function, one needs to solve the SPDE for every stochastic collocation point. Let n denote the total number of queries. On a fine grid $N_h = 128^2$, the computational time of gSC will be

$$t_{gSC}(n) = 0.2455Jn. \tag{67}$$

4.1.4. Comparison of computational complexities. First, we compare the computational cost of DSM and gPC solvers. Simple calculations show that if we need to solve the original SPDE with more than $n_c = [\frac{t_2 - t_1}{0.0223(N_p^2 - m^2)}] + 1$ different forcing functions, the DSM solver will be superior to the gPC solver. Let us substitute all the previous results, i.e., (56) and (65), into n_c; we can obtain a rough estimate for n_c as follows:

$$n_c(N_p, m, n_{up}, J, K) \leq \left[\frac{0.2455J(1 + n_{up}) + 0.0088n_{up}m^2 + 0.0029n_{up}Km^{1.4} - 0.1897N_p^2}{0.0223(N_p^2 - m^2)} \right] + 1. \tag{68}$$

Similarly, we can get the result for the comparison between DSM and the gSC solver; i.e., the DSM solver will be superior to the gSC solver when we need to solve the original SPDE with more than $n_c = [\frac{t_2}{0.2455J - 0.0223m^2}] + 1$ different forcing functions, where n_c can be bounded by

$$n_c(m, n_{up}, J, K) \leq \left[\frac{0.2455J(1 + n_{up}) + 0.0088n_{up}m^2 + 0.0029n_{up}Km^{1.4}}{0.2455J - 0.0223m^2} \right] + 1. \tag{69}$$

When the SPDE solution is not smooth in the stochastic dimension, the DSM in the collocation representation may not be efficient since one needs a large number of collocation points to represent the solution. In this case, all the existing methods are expensive, and the DSM with the ensemble representation will be the method of choice. We can compare the computational complexity of the DSM Monte Carlo solver and the reference solver, i.e., the Monte Carlo solver. Let M_{dsm} and M_{mc} denote the sample number used in the DSM Monte Carlo solver and the reference solver, respectively. The DSM Monte Carlo solver will be superior to the reference solver when we need to solve the original SPDE with more than $n_c = [\frac{t_2}{0.2455M_{mc} - 0.0223m^2}] + 1$ different forcing functions, where n_c is bounded by

$$n_c(m, n_{up}, M_{dsm}, M_{mc}, K) \leq \left[\frac{0.2455M_{dsm}(1 + n_{up}) + 0.0088n_{up}m^2 + 0.0029n_{up}Km^{1.4}}{0.2455M_{mc} - 0.0223m^2} \right] + 1. \tag{70}$$

Inequalities (68), (69), and (70) give the upper bound of n_c obtained from our computational time model. The specific value of n_c depends on many parameters and is problem dependent. From our computational experience, we find that n_c can be relatively small. We will further demonstrate this through a model problem in the next subsection.

4.2. A 1D model problem. Choosing the gPC basis number N_p, stochastic collocation points number J, and the data-driven stochastic basis number m to obtain an accurate solution is problem dependent. In this subsection, we design a model problem to understand this issue.

Table 6

Compare gSC, gPC, and DSM-gSC. The data marked with an asterisk are obtained by extrapolation.

$\beta = 2$				$\beta = 1$				$\beta = \frac{1}{2}$			
r	J	N_p	m	r	J	N_p	m	r	J	N_p	m
2	9	10	8(0.29%)	2	37	15	8(0.28%)	2	45	28	8(1.45%)
3	19	20	8(1.01%)	3	93	35	8(1.39%)	3	165	84	17(1.23%)
4	33	35	11(1.10%)	4	201	70	14(1.26%)	4	441	210	29(1.49%)
5	51	56	11(1.08%)	5	401	126	20(1.47%)	5	993	462	38(2.11%)
6	73	84	11(1.41%)	6	749	210	26(1.40%)	6	2021	792(*)	44(4.23%)
7	99	120	11(1.49%)	7	1317	330	29(1.63%)	7	3837	1716(*)	44(7.33%)
8	129	165	14(1.62%)	8	2193	495	32(1.81%)	8	6897	6435(*)	44(12.82%)

Recall that our computational time model in section 4.1 is obtained from a 2D elliptic SPDE with physical grid chosen as $N_h = 128^2$. Due to constraints in our computational resources, we cannot perform a meaningful comparison of different methods for a very challenging 2D stochastic problem with high-dimensional random coefficients whose stochastic solutions are not very smooth. Instead, we construct a carefully designed 1D model problem which shares some essential difficulties of the 2D problem.

We consider the following 1D elliptic SPDE:

$$-\frac{\partial}{\partial x}\left(a(x,\omega)\frac{\partial}{\partial x}u(x,\omega)\right) = f(x), \quad x \in D = (0,1),\ \omega \in \Omega, \tag{71}$$

$$u(0,\omega) = 0, \quad u(1,\omega) = 0. \tag{72}$$

The random coefficient is defined by

$$a(x,\omega) = e^{\sum_{n=1}^{r}\sqrt{\theta_n}\xi_n(\omega)\phi_n(x)}, \quad \xi_n \in \mathcal{N}(0,1), \tag{73}$$

where

$$\theta_n = \theta_0\left(\frac{1}{n}\right)^{\beta}, \quad n = 1,2,\ldots,$$
$$\phi_n(x) = \sin(n\pi x), \quad n = 1,2,\ldots.$$

The parameter β is used to control the decay rate of the eigenvalues and θ_0=0.4. Generally speaking, slow decay in the eigenvalues results in a hard problem, which requires more basis elements in the DSM and gPC methods or sparse grid points in the stochastic collocation method to accurately resolve the stochastic solution. In our test, we will choose $\beta = 2$, $\beta = 1$, and $\beta = 0.5$. They correspond to easy, moderate, and difficult cases, respectively.

We choose r ranging from 2 to 8 as a low-dimensional input test and from 15 to 20 as a high-dimensional input test. The function class of the right-hand side is chosen to be $\mathfrak{F} = span\{\sin(i\pi x), \cos(i\pi x)\}_{i=1}^{10}$. We use the standard piecewise linear finite element method with mesh size $h = \frac{1}{256}$ to solve this elliptic problem.

In Table 6 we compare the results of different methods. For each fixed β and r, we list the minimal number of gPC basis elements (denoted by N_p) and stochastic collocation points (denoted by J) so that the relative error of the STD of the solution is less than 1% for all

Table 7
DSM-MC for high-dimensional input. $\beta = 1$.

r	J	N_p	m
15	27701	3876	38(4.93%)
16	36129	4845	38(5.58%)
17	46377	5985	41(5.91%)
18	58693	7315	44(6.89%)
19	73341	8855	44(7.41%)
20	90601	10626	47(7.83%)

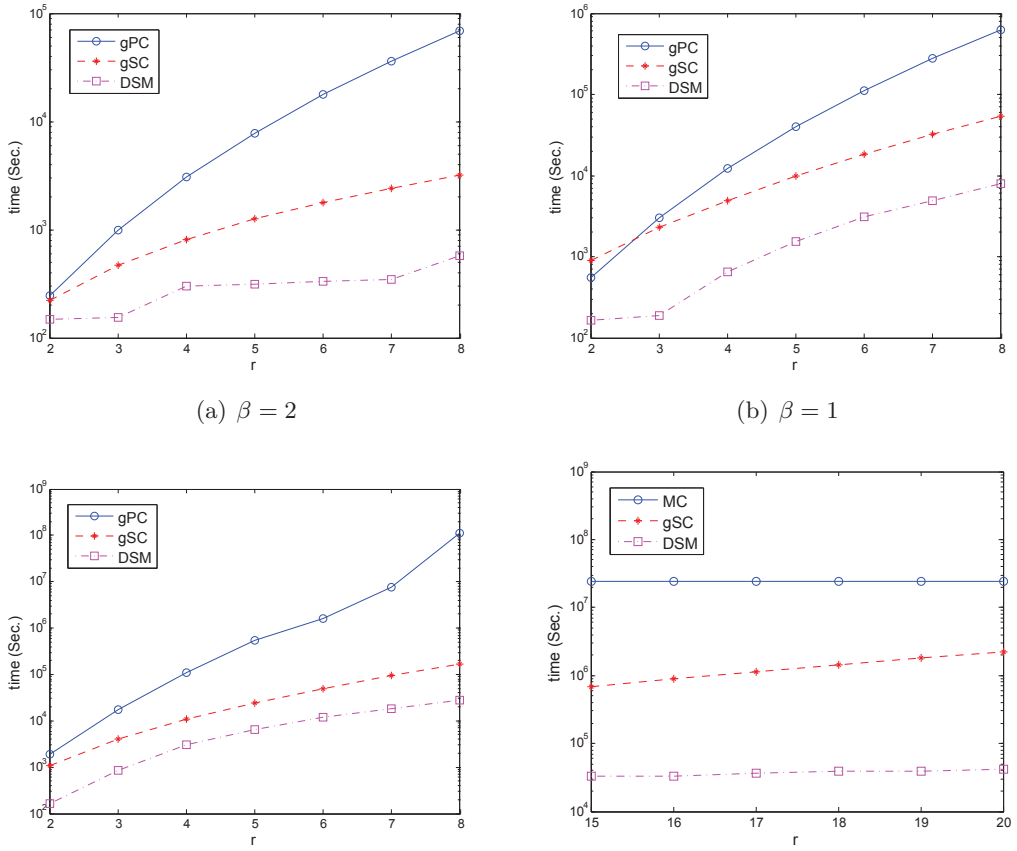
$f(x) \in \mathfrak{F}$. We also list the basis number of the DSM and the corresponding maximum STD error. For the case $\beta = 2$, our DSM can capture the low-dimensional structure of the solution very well. For the case $\beta = 1$, we need more basis elements or sparse grids to obtain accurate results. Our DSM still gives a compact representation for the SPDE solutions. For the case $\beta = \frac{1}{2}$, one needs a high-order polynomial basis to approximate the rough solution which makes the gPC method more and more expensive as the dimension of the input variable increases. The data marked with an asterisk are obtained by extrapolation since it is already out of memory using the gPC method. The gSC still works but requires many more collocation points, becoming very expensive. For this difficult problem, our DSM can still give a good approximation of the SPDE solution. Combined with the error correction approach proposed in section 3.5, our DSM still offers a very effective alternative.

In Table 7 we show the results for the high-dimensional input test. The reference solution is obtained by the Monte Carlo method with 10^6 samples. Both gSC and gPC become extremely expensive in this case. Since DSM with the collocation representation also becomes quite expensive, we choose the DSM with the ensemble representation. We use 10^4 Monte Carlo samples to represent the DSM basis. Our DSM can still capture about 95% of the STD information.

We conjecture that the results shown in Tables 6 and 7 may still be valid to some extent for 2D SPDE problems. Thus, it would be interesting to investigate what the implications for 2D SPDEs would be if we used the results obtained in Tables 6 and 7 as a guide. For this reason, we substitute these parameters into our computational time model to compare the computational cost of different methods. In Figure 2, we show the total computational time of 100 queries using different methods. As we can see, DSM offers significant computational savings over other traditional methods. The savings of DSM over gPC are several orders of magnitude. Even compared with gSC, our DSM method still offers considerable savings. In Figure 3, we plot the critical query number n_c for different scenarios. We can see that with all the cost-saving measures in the offline stage, n_c is relatively small. We notice that $n_c = 1$ when we compare with gPC and n_c is less than 9 when compared with gSC even for the moderate and difficult cases. All these results demonstrate that DSM is very effective in a multiquery setting.

5. Numerical examples.

In this section, we perform a number of numerical experiments to test the performance and accuracy of the proposed DSM for elliptic SPDEs with random coefficients. As we will demonstrate, the DSM could offer accurate numerical solutions to SPDEs with significant computational saving in the online stage over traditional stochastic



(a) $\beta = 2$ (b) $\beta = 1$

(c) $\beta = 0.5$ (d) $\beta = 1$

Figure 2. *Total computational time for* 100 *queries with different* β.

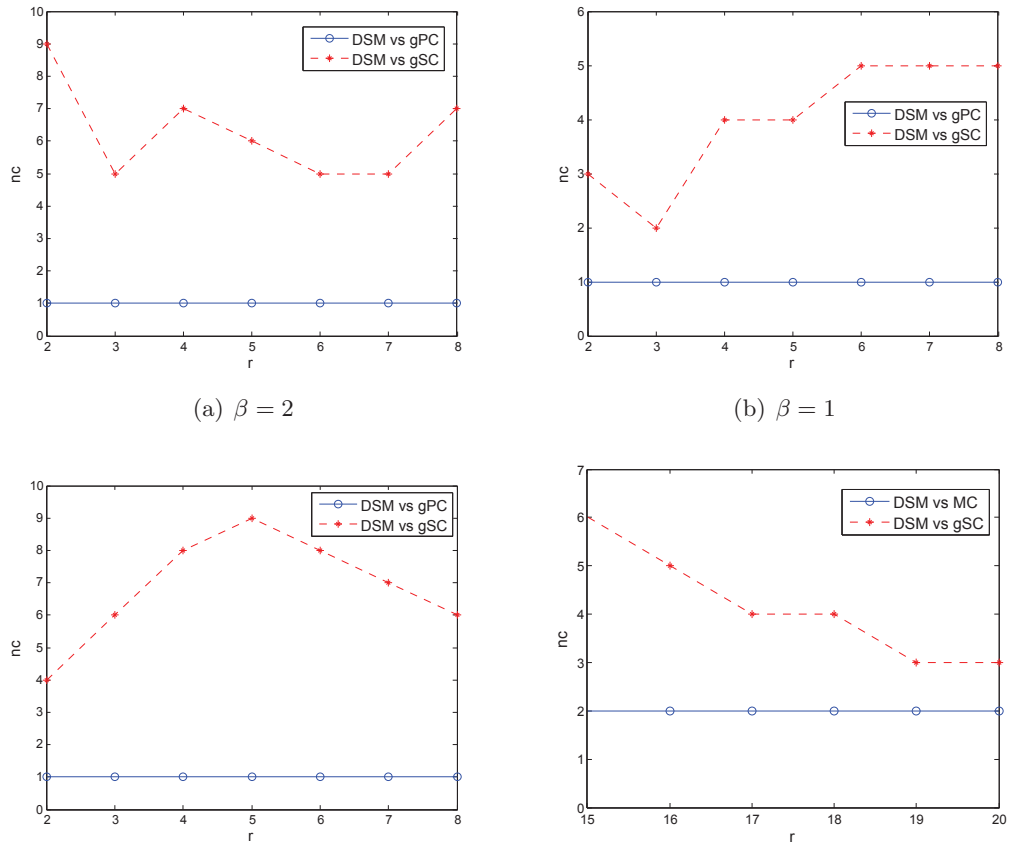
methods. The specific rate of savings will depend on how we represent the data-driven stochastic basis. We will use three methods to represent the data-driven stochastic basis: (i) Monte Carlo methods, (ii) generalized stochastic collocation methods (gSC), and (iii) generalized polynomial chaos methods (gPC). We denote them as DSM-MC, DSM-gSC, and DSM-gPC, respectively.

5.1. DSM for a 1D elliptic SPDE. We consider the following 1D elliptic SPDE with random coefficient:

$$-\frac{\partial}{\partial x}\left(a(x,\omega)\frac{\partial}{\partial x}u(x,\omega)\right) = f(x), \quad x \in D = (0,1),\ \omega \in \Omega, \tag{74}$$

$$u(0,\omega) = 0, \quad u(1,\omega) = 0. \tag{75}$$

We will apply the DSM-MC, DSM-gSC, and DSM-gPC methods to solve this problem. When modeling a whole aquifer or a whole oil reservoir, the correlation length scale for random field $a(x,\omega)$ is in general significantly smaller than the size of the computational region. However, the correlation is typically large enough to fall outside the domain of stochastic homogenization



(a) $\beta = 2$ (b) $\beta = 1$

(c) $\beta = 0.5$ (d) $\beta = 1$

Figure 3. *Critical query number of the data driven stochastic method.*

techniques. In addition, typical sedimentation processes lead to fairly irregular structures and pore networks. A faithful model should assume only limited spatial regularity of $a(x, \omega)$. A covariance function that has been proposed is the following exponential two-point covariance function:

$$C(x, y) = \sigma^2 e^{-\frac{|x-y|^p}{\lambda}}, \quad x, y \in [0, 1]. \tag{76}$$

The parameters σ^2 and λ denote the variance and the correlation length, respectively. In this paper, we choose $p = 1$, $\sigma^2 = 1$, and $\lambda = 0.1$.

There are several ways to produce samples of $a(x, \omega)$, including the circulant embedding and the KL expansion. We use the KL expansion here. Let $k(x, \omega) = \log(a(x, \omega))$. We expand $k(x, \omega)$ in terms of a countable set of uncorrelated, zero mean random variables $\{\xi_n\}_{n=1}^{\infty}$ such that

$$k(x, \omega) = \sum_{n=1}^{\infty} \sqrt{\theta_n} \xi_n(\omega) \phi_n(x), \tag{77}$$

where we assume $E[k(x,\omega)] = 0$ and $\{\theta_n, \phi_n(x)\}_{n=1}^{\infty}$ are the eigenpairs of the covariance function (76). An important point to note is that for Gaussian random field $k(x,\omega)$ the random variables $\{\xi_n\}_{n=1}^{\infty}$ are a set of independent standard Gaussian variables, i.e., $\xi_n \in \mathcal{N}(0,1)$. The decay rate of the eigenvalues θ_n depends on the regularity of the covariance function (76) and the correlation length λ; see [36]. In our setting, i.e., $p = 1$ and $\lambda = 0.1$ in (76), the eigenpairs of the covariance function (76) have analytic expressions

$$\theta_n = \frac{2\lambda}{\lambda^2 w_n^2 + 1}, \quad n = 1, 2, \ldots,$$
$$\phi_n(x) = C_n(\sin(w_n x) + \lambda w_n \cos(w_n x)), \quad n = 1, 2, \ldots,$$

where $\{w_n\}$ are the real solutions of the transcendental equation $\tan(w) = \frac{2\lambda w}{\lambda^2 w^2 - 1}$ and C_n are normalization constants [16]. In practice we truncate the expansion (77) after a finite number of terms K and define the coefficient as

$$a(x,\omega) = e^{\sum_{n=1}^{K} \sqrt{\theta_n}\xi_n(\omega)\phi_n(x)}, \quad \xi_n \in \mathcal{N}(0,1). \tag{78}$$

We choose $K = 8$ in (78). The function class of the right-hand side is chosen to be $\mathfrak{F} = span\{1, \sin(i\pi x), \cos(i\pi x)\}_{i=1}^{15}$. We use the standard piecewise linear finite element method with mesh size $h = \frac{1}{256}$ to solve this elliptic problem. For the gSC or gPC method, the Hermite polynomials are used for the stochastic approximation. Since the coefficient $a(x,\omega)$ has eight independent random variables, we choose $r = 8$ and $p = 3$ in the orthonormal basis index (9). This gives rise to the multi-index set $\mathfrak{J} = \mathfrak{J}_8^3$, which results in a total number of 165 terms in the basis functions, i.e., $N_p = 165$. For the Monte Carlo method, we use 4×10^4 realizations in the offline stage to train the DSM basis.

Let $u_{DSM}(x,\omega)$ denote the data-driven solution and $u(x,\omega)$ the exact solution. To quantify the error, we define the relative error of mean and STD in $L^2(D)$ as

$$e_{mean} = \frac{||\bar{u}(x) - \bar{u}_{DSM}(x)||_{L^2(D)}}{||\bar{u}(x)||_{L^2(D)}}$$

and

$$e_{STD} = \frac{||STD(u) - STD(u_{DSM})||_{L^2(D)}}{||STD(u)||_{L^2(D)}}.$$

Convergence of the offline stage. We first test the convergence of the two-level data-driven basis updating procedure in the offline stage. The updating procedure of the data-driven basis in the ensemble representation, gSC representation, and gPC representation give similar results. We show only the results of the DSM in the gPC representation. Let $\mathcal{E}_\tau = \|\tau(x,\omega)\|_{L^2(D\times\Omega)}$ denote the L^2 norm of the maximum error, where $\tau(x,\omega)$ is defined in (43).

We list in Table 8 the decay of the maximum error. Initially, we solve (74) with $f(x) = 1$ to obtain the data-driven basis $\mathbf{A}$, which has six effective modes. It is clear that this stochastic basis is insufficient. The maximum residual error is 38.05%. Then we begin the two-level data-driven basis updating procedure. Every time, we add three modes to the data-driven basis

Table 8
Error decay of the maximum residual error in the DSM-gPC method.

DSM mode	$m = 6$	$m = 9$	$m = 12$	$m = 15$	$m = 18$	$m = 21$
$\mathcal{E}_\tau$	38.05%	17.28%	4.983%	2.014%	1.402%	0.798%

Table 9
Relative errors of statistical quantities computed by DSM and gPC.

Methods	Compare to u_{exact}		Compare to u_{gPC}		Time (sec.)
	e_{mean}	e_{STD}	e_{mean}	e_{STD}	
gPC $N_p = 165$	0.1149%	1.326%	NA	NA	2.1762
DSM $m = 6$	3.9066%	11.912%	3.8762%	10.982%	0.0047
DSM $m = 9$	1.281%	5.0201%	1.2568%	4.0736%	0.0105
DSM $m = 12$	0.3133%	1.7821%	0.2833%	0.4849%	0.0201
DSM $m = 15$	0.2465%	1.577%	0.2167%	0.2914%	0.0319
DSM $m = 18$	0.2025%	1.421%	0.1733%	0.1724%	0.0489
DSM $m = 21$	0.1482%	1.349%	0.1178%	0.1562%	0.0669

A. After six updates, our method converges (here the termination condition is $\mathcal{E}_\tau < 0.8\%$), and we obtain an optimal data-driven stochastic basis, which has 21 modes.

Error analysis of DSM. To understand the source of errors in the DSM method, we decompose the error into two terms. The first source of error $\mathcal{E}_1$ is the difference between the exact solution u_{exact} and the gPC approximate solution u_{gPC}. The second source of error $\mathcal{E}_2$ is the difference between the DSM solution u_{DSM} and the gPC solution u_{gPC}. More precisely, we have

$$\begin{aligned}\mathcal{E} = u_{exact} - u_{DSM} &= \{u_{exact} - u_{gPC}\} + \{u_{gPC} - u_{DSM}\} \\ &= \mathcal{E}_1 + \mathcal{E}_2.\end{aligned}$$

The error $\mathcal{E}_1$ is controlled by the multi-index set $\mathfrak{J}$. According to Cameron–Martin theorem [5], $\mathcal{E}_1$ converges in the $L^2(D \times \Omega)$ sense as $|\mathfrak{J}| \to \infty$ on the condition that the exact solution u_{exact} has a finite second moment, while the error $\mathcal{E}_2$ diminishes as $m \to N_p$. Thus, there is no need to increase m any further once $\mathcal{E}_2 \ll \mathcal{E}_1$.

Next, we test the effectiveness of the data-driven stochastic basis. We solve the (74) with the coefficient $a(x, \omega)$ given by (78) and $f(x) = \sin(1.2\pi x) + 4\cos(3.6\pi x)$. Here the "exact" solution is obtained by the Monte Carlo method with 10^6 realizations. The relative errors of mean and STD are tabulated in Table 9. The first and second columns are the comparisons between the gPC or DSM solution with the exact solution, while the third and fourth columns are the comparisons between the gPC and DSM-gPC solutions. Indeed, as the number of mode pairs in DSM increases, the $L^2(D)$ errors of mean and STD decrease, indicating the convergence of u_{DSM} to u_{gPC}. When $m = 21$, the error between the DSM solution and the exact solution is comparable with the error between the gPC solution and the exact solution.

Multiple query results in online stage. In the DSM-MC method, we use 40,000 realizations in the computation, and its offline training stage takes about 1,115 seconds. The DSM-gSC method uses 2,193 sparse grid points in the computation, and its offline training takes 283

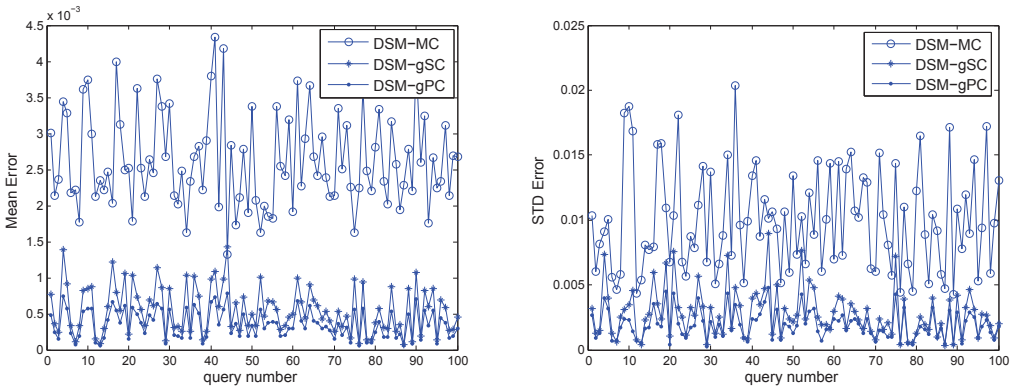


Figure 4. *Comparison of DSM in MC, gPC, and gSC representation. DSM basis $m = 21$.*

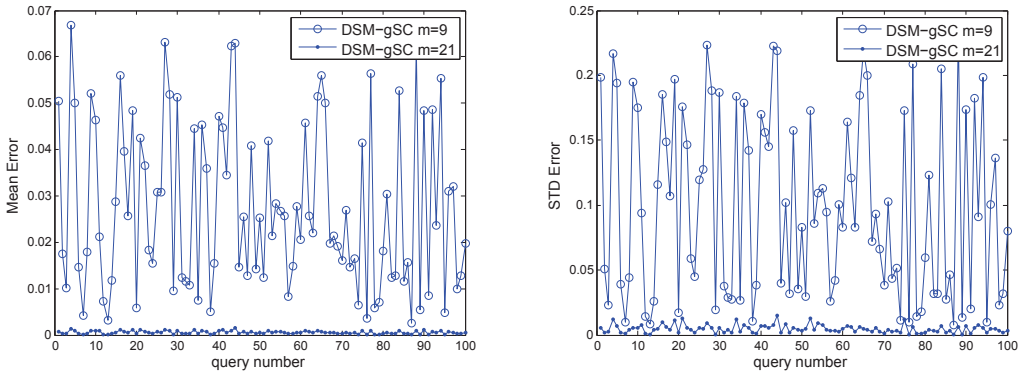


Figure 5. *Comparison of the sufficiency of DSM.*

seconds. The DSM-gPC method uses 165 Hermite polynomials in the polynomial chaos basis, and its offline training takes 365 seconds. We use $m = 21$ modes in the DSM basis in the DSM-MC, DSM-gSC, and DSM-gPC methods in the online stage to solve (74). We randomly generate 100 force functions, i.e., $f(x) \in \{c_i \sin(2\pi k_i x + \phi_i) + d_i \cos(2\pi l_i x + \varphi_i)\}_{i=1}^{100}$, where c_i, d_i, k_i, l_i, ϕ_i, and φ_i are random numbers. In Figure 4 we show the mean and STD comparison of DSM in the Monte Carlo, gSC, and gPC representations in the online stage. One can see that the stochastic basis generated by DSM-MC, DSM-gSC, and DSM-gPC is very effective in the sense it can be used to solve (74) with a large class of right-hand side force functions. We note that DSM-gSC and DSM-gPC are more accurate than DSM-MC. Taking into account the training time in the offline stage and the accuracy in the online stage, DSM-gSC gives the best performance. To further illustrate the necessity of using multiple trial functions to update the stochastic basis, we plot the numerical results obtained by nine modes in the DSM basis and the 21 modes in the DSM basis in Figure 5. We plot only the result of DSM-gSC, since the results of DSM-MC and DSM-gPC are similar. We can see that the computation using only nine modes gives significantly larger errors than those produced by the computation using 21 modes, indicating that the stochastic basis with nine modes is not sufficient to represent the

Table 10
The a posteriori error estimate of the first-order moment. $m = 21$.

M_{rc}	$\|\|E[u]\|\|_{L^2}$	$\|\|E[u_{DSM}]\|\|_{L^2}$	$\tau^1_{M_{rc}}$	$\tau^2_{M_{rc}}$	E_1	E_2
50	5.699e-2	5.694e-2	2.292e-4	2.271e-4	0.124%	0.406%
200	5.699e-2	5.694e-2	3.967e-4	3.156e-4	0.124%	0.281%
800	5.699e-2	5.694e-2	4.301e-5	6.312e-5	0.124%	0.145%

Table 11
The a posteriori error estimate of the second-order moment. $m = 21$.

M_{rc}	$\|\|E[u^2]\|\|_{L^2}$	$\|\|E[u^2_{DSM}]\|\|_{L^2}$	$\eta^1_{M_{rc}}$	$\eta^2_{M_{rc}}$	E_1	E_2
50	6.869e-3	6.810e-3	5.531e-5	7.046e-5	0.875%	0.123%
200	6.869e-3	6.810e-3	4.143e-5	3.705e-5	0.875%	0.518%
800	6.869e-3	6.810e-3	1.721e-5	1.832e-5	0.875%	0.722%

class of forcing functions that we consider here.

The a posteriori error estimate and error correction in the online stage. We use the same notation as in section 3.5. To calculate the second-order moment, we denote $r^2(x,\omega) = u^2(x,\omega) - u^2_{DSM}(x,\omega)$ and use $\bar{r}^2(x)$ and $\tilde{r}^2(x)$ to denote the corresponding sample mean and sample STD. Let $\eta^1_{M_{rc}} = ||\bar{r}^2(x)||$ and $\eta^2_{M_{rc}} = ||\frac{\tilde{r}^2(x)}{\sqrt{M_{rc}}}||$ denote their $L^2(D)$ norm, where M_{rc} is the number of Monte Carlo samples. We solve (74) only once with coefficient $a(x,\omega)$ given by (78) and $f(x) = \sin(1.2\pi x) + 4\cos(3.6\pi x)$ as an example. Here the "exact" solution is obtained by the Monte Carlo method with 10^6 realizations. In Tables 10–13, E_1 stands for the relative error between the data-driven solution and the exact solution without using the Monte Carlo error correction, while E_2 stands for the same relative error with the Monte Carlo error correction.

Tables 10 and 11 show the a posteriori error estimate of the first- and second-order moments, respectively. Since the data-driven basis with 21 modes is approximately a complete basis, several hundreds of Monte Carlo simulations indicate the convergence of the data-driven solution, i.e., $\tau^1_{M_{rc}} + 2\tau^2_{M_{rc}} \ll ||E[u_{DSM}]||_{L^2}$ and $\eta^1_{M_{rc}} + 2\eta^2_{M_{rc}} \ll ||E[u^2_{DSM}]||_{L^2}$. In this case, the error correction is not necessary. In practical computations, the exact solution is unknown. This example shows that when the data-driven stochastic basis spans the SPDE solution space, this *a posteriori error* estimate is very effective.

Tables 12 and 13 show the error correction of the first- and second-order moments, respectively. We choose the first 10 modes in the data-driven basis to produce an insufficient basis. The first several hundred Monte Carlo simulations indicate that the $L^2(D)$ norm of the residual error cannot be neglected. One can see that every time we increase the realization number M_{rc} by a factor of 4, the norms of the sample STD $\tau^2_{M_{rc}}$ and $\eta^2_{M_{rc}}$ decrease by a factor of 2. When the ratios $\tau^2_{M_{rc}}/\tau^1_{M_{rc}}$ and $\eta^2_{M_{rc}}/\eta^1_{M_{rc}}$ gradually become less than some predefined threshold, we obtain a good error correction for the data-driven solution. In this example, the relative error of the second-order moment will be 4.675% if we use the insufficient data-driven basis to solve the SPDE. However, when we use several hundred Monte Carlo simulations to provide an error correction, this error becomes less than 1%.

Table 12
The error correction of the first-order moment. $m = 10$.

M_{rc}	$\|\|E[u]\|\|_{L^2}$	$\|\|E[u_{DSM}]\|\|_{L^2}$	$\tau^1_{M_{rc}}$	$\tau^2_{M_{rc}}$	E_1	E_2
50	5.699e-2	5.623e-2	2.567e-3	1.017e-3	1.816 %	3.701 %
200	5.699e-2	5.623e-2	1.101e-3	4.264e-4	1.816 %	0.736 %
800	5.699e-2	5.623e-2	9.596e-4	2.077e-4	1.816 %	0.235 %
3200	5.699e-2	5.623e-2	9.617e-4	1.075e-4	1.816 %	0.165 %
12800	5.699e-2	5.623e-2	1.015e-3	5.381e-5	1.816 %	0.144 %

Table 13
The error correction of the second-order moment. $m = 10$.

M_{rc}	$\|\|E[u^2]\|\|_{L^2}$	$\|\|E[u^2_{DSM}]\|\|_{L^2}$	$\eta^1_{M_{rc}}$	$\eta^2_{M_{rc}}$	E_1	E_2
50	6.869e-3	6.551e-3	5.765e-4	2.137e-4	4.675%	4.483%
200	6.869e-3	6.551e-3	2.901e-4	1.009e-4	4.675%	0.793%
800	6.869e-3	6.551e-3	2.988e-4	4.155e-5	4.675%	0.446%
3200	6.869e-3	6.551e-3	3.288e-4	3.166e-5	4.675%	0.185%
12800	6.869e-3	6.551e-3	3.216e-4	1.586e-5	4.675%	0.059%

5.2. DSM for 2D elliptic SPDE. In this section, we apply our data-driven method to solve the following 2D stochastic elliptic problem with a random coefficient:

$$-\nabla \cdot (a(x,y,\omega)\nabla u(x,y,\omega)) = f(x,y), \quad (x,y) \in D, \quad \omega \in \Omega, \tag{79}$$

$$u(x,y,\omega) = 0, \qquad (x,y) \in \partial D, \quad \omega \in \Omega, \tag{80}$$

where $D = [0,1] \times [0,1]$. The random coefficient is defined as

$$a(x,y,\omega) = e^{\sum_{i=1}^{4} \sqrt{\lambda_i}\xi_i\varphi_i(x,y)}, \quad \xi_i \in \mathcal{N}(0,1), \tag{81}$$

where the ξ_i's are independent random variables, $\lambda_i = 1/i^2$, and $\varphi_i(x,y) = \sin(2\pi i x)\cos(2\pi(5-i)y)$, $i = 1,\ldots,4$. In the offline stage, the function class of the right-hand side in the preconditioning DSM method is chosen to be $\mathfrak{F} = \{\sin(2\pi k_i x + \phi_i)\cos(2\pi l_i y + \varphi_i)\}_{i=1}^{20}$, where k_i and l_i are random wavenumbers and ϕ_i and φ_i are random phases. We use this random training strategy to reduce the computational cost. The FEM is used for the spatial discretization. We first partition the domain D into squares with mesh size $h = \frac{1}{128}$ and then further partition them into triangular meshes. Taking into account the slow convergence of DSM-MC, we only discuss and compare DSM-gSC and DSM-gPC here. Hermite polynomials are used to approximate the stochastic solution. Since the coefficient $a(x,y,\omega)$ has four independent random variables, we choose $r = 4$ and $p = 3$ in the orthonormal basis index (9), which results in a total number of 35 terms in the basis functions, i.e., $N_p = 35$.

Multiple query results in the online stage. The DSM-gSC method uses 201 sparse grid points in the computation, and its offline training takes 362 seconds. The DSM-gPC method uses 35 Hermite polynomials in the polynomial chaos basis, and its offline training takes 1,261 seconds. Finally, both the DSM-gSC and DSM-gPC methods produce $m = 13$ modes. In the online stage we use them to solve (79). We randomly generate 100 force functions of the form $f(x,y) \in \{\sin(k_i\pi x + l_i\pi y)\cos(m_i\pi x + n_i\pi y)\}_{i=1}^{100}$, where k_i, l_i, m_i, and n_i are

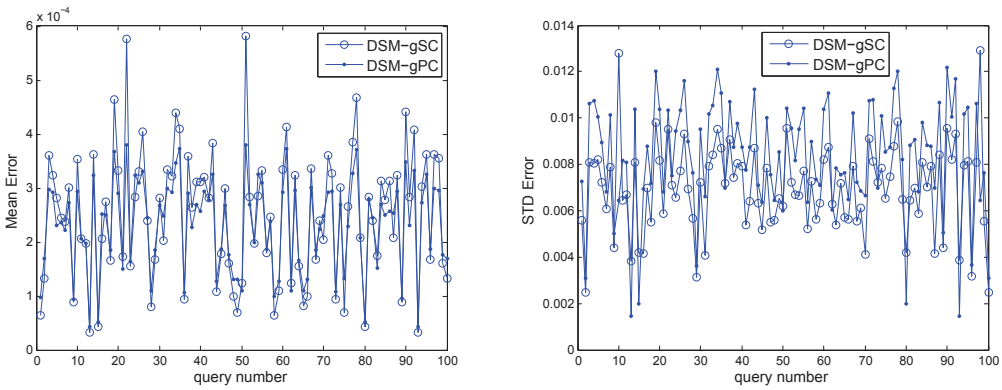


Figure 6. *Comparison of DSM in the gSC and gPC representations. DSM basis $m = 13$.*

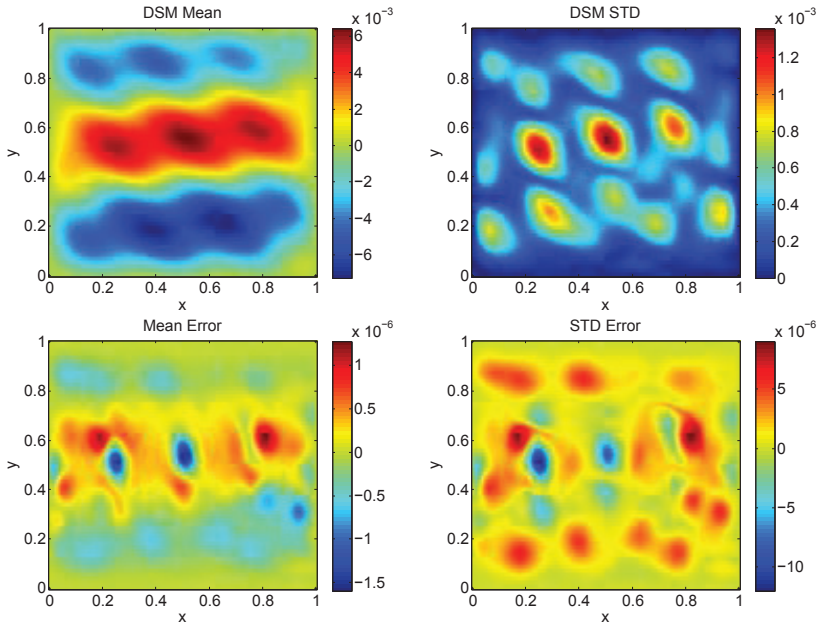


Figure 7. *Comparison of the mean and STD in the 2D SPDE with Gaussian random variables.*

random numbers. In Figure 6, we show the mean and STD comparison of DSM in gSC and gPC representations in the online stage. One can see that both the DSM-gSC basis and the DSM-gPC basis are very effective in the sense they can be used to solve (79) with a large class of right-hand side force functions. Here the "exact" solution is obtained by the stochastic collocation method with 1,305 sparse grid points. In Figure 7 we show one of the query results, where $f(x, y) = \sin(1.3\pi x + 3.4\pi y)\cos(4.3\pi x - 3.1\pi y)$. The mean and STD of the solution obtained by the stochastic collocation method and DSM-gSC as well as their errors are given. In this example, the relative error of mean and STD are 0.01% and 0.65%, respectively. The DSM-gPC has similar results (not shown here).

Compare the DSM, gPC, and gSC. Let n denote the total query number. The computa-

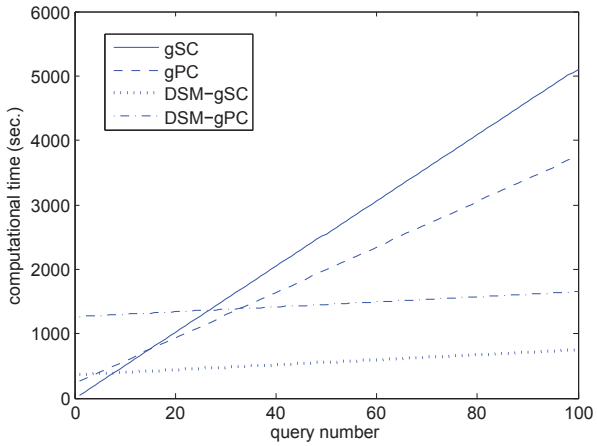


Figure 8. *The computation time comparison. All times are in seconds.*

tional cost of gSC will be $t_{gSC}(n) = 51.80n$. For the gPC method with $N_p = 35$, we generate the stiffness matrix S and factorize it, which takes 226.57 seconds. In the online stage, one step of forward and backward substitutions takes 35.32 seconds. The computational cost of gPC will be $t_{gPC}(n) = 226.57 + 35.32n$. In the online stage of DSM, the size of the stiffness matrix S will be smaller than that of the gPC method. One step of forward and backward substitutions takes 3.89 seconds. Recall that the offline training of DSM-gSC and DSM-gPC takes 362 seconds and 1,261 seconds, respectively. The computational cost of DSM-gSC and DSM-gPC will be $t_{DSM-gSC}(n) = 362 + 3.89n$ and $t_{DSM-gPC}(n) = 1261 + 3.89n$. We plot the total computational time in Figure 8. Simple calculation shows that if we need to solve the original SPDE with more than five (or eight) different forcing functions, the DSM-gSC will be superior to the gPC (or gSC) solver. Similar results can be obtained for the DSM-gPC solver.

Compare offline stage of DSM-gSC and DSM-gPC. The online stage of DSM-gSC and DSM-gPC have the same accuracy and computation time; however, the offline time is different. Let N_h denote the physical degree of freedom, N_p denote the number of polynomials, and J denote the number of stochastic collocation (sparse grid) points. In the DSM-gPC method we solve a coupled deterministic PDE system with $N_h N_p$ unknowns to obtain the expansion coefficients, while in the DSM-gSC method we solve J uncoupled deterministic equations with N_h unknowns. Each solution corresponds to a collocation point and has its own weight. We use gSC and gPC to solve (79) with coefficient $a(x, y, \omega)$ given by (81) and $f(x, y) = \sin(1.3\pi x + 3.4\pi y)\cos(4.3\pi x - 3.1\pi y)$. We fix $N_h = 128^2$ and choose the level of the sparse grid from 2 to 5, i.e., $J = 9$, 33, 81, and 201 in the gSC method, and choose $N_p = 5$, 15, 25, and 35 in the gPC method. The "exact" solution is obtained by the level nine quadrature rule, which has 2,129 sparse grid points. Table 14 indicates when the solution is smooth the gSC method is very effective. When we increase the number of polynomials N_p, the gPC method and the DSM-gPC method become very expensive or even infeasible. The main cost comes from solving the linear equation system due to the memory requirement in a direct method

Table 14
Relative errors of statistical quantities computed by gSC and gPC.

gSC method				gPC method			
J	e_{mean}	e_{STD}	time (sec.)	N_p	e_{mean}	e_{STD}	time (sec.)
9	0.1636%	5.268%	3.529	5	0.6921%	15.05%	9.934
33	0.0230%	1.338%	9.628	15	0.0385%	1.841%	58.67
81	0.0047%	0.2582%	21.84	25	0.0051%	0.3292%	183.8
201	0.0003%	0.0259%	51.80	35	0.0020%	0.1902%	417.3

and the computational time in an iterative method. However, the DSM-gSC method is still very effective due to its nonintrusive nature.

To further demonstrate the effectiveness of DSM-gSC, we consider (79) with another random coefficient given by

$$a(x,y,\omega) = 8.01 + \sum_{i=1}^{8} \xi_i(\omega)\sin(\pi i x)\cos(\pi(9-i)y), \tag{82}$$

where the ξ_i's are independent uniform random variables on $[0,1]$. In the offline stage, the function class of the right-hand side in the preconditioning DSM is chosen to be $\mathfrak{F} = \{\sin(2\pi k_i x + \phi_i)\cos(2\pi l_i y + \varphi_i)\}_{i=1}^{20}$, where k_i and l_i are random wavenumbers, while ϕ_i and φ_i are random phases. As before, we use this random training strategy to reduce the computational cost. The FEM is used for the spatial discretization with $h = \frac{1}{128}$. Legendre polynomials are used for the stochastic space approximation. Since the coefficient $a(x,y,\omega)$ has eight independent random variables, we choose $r = 8$ and $p = 3$ in the orthonormal basis index (9), which results in a total number of 165 basis functions, i.e., $N_p = 165$. In this case, the gPC method is too expensive to compute within the limit of our computational resources.

We use the level four sparse grid in the stochastic collocation method to train the data-driven stochastic basis, which has 609 points. We have used higher-level sparse grid points for the convergence study and found out the relative errors of mean and STD between the solutions obtained by 609 sparse grids and higher-level sparse grids are smaller than 0.1%. The offline training time of the DSM-gSC method takes 674 seconds. The DSM-gSC method gives $m = 10$ modes. In the online stage we use them to solve (79). We randomly generate 100 force functions of the form $f(x,y) \in \{\sin(k_i\pi x + l_i\pi y)\cos(m_i\pi x + n_i\pi y)\}_{i=1}^{100}$, where k_i, l_i, m_i, and n_i are random numbers. In Figure 9 we show the mean and STD comparison of DSM in gSC representation in the online stage. Here the reference solution is obtained by the stochastic collocation method with 2177 sparse grids. One can see that the DSM-gSC basis is very effective in the sense they can be used to solve (79) with a large class of right-hand side force functions. In Figure 10 we show one of the query results with $f(x,y) = \sin(5.3\pi x + 2.3\pi y)\cos(6.4\pi x - 4.1\pi y)$. The mean and STD of the solution obtained by the stochastic collocation method and DSM-gSC as well as their errors are given. In this example, the relative error of the mean and STD are 0.019% and 0.64%, respectively.

5.3. An advantage of the DSM-MC method. When the input dimension of the random variables is high or if the stochastic solution is not sufficiently smooth, the DSM-gSC method will be very expensive or even infeasible. Although the Monte Carlo method has a slow

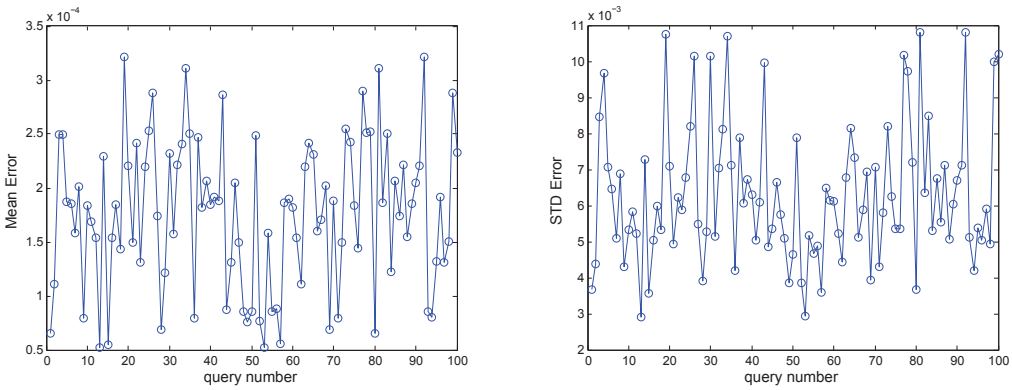


Figure 9. *DSM in gSC representation for large N_p. $N_p = 165$. DSM basis $m = 10$.*

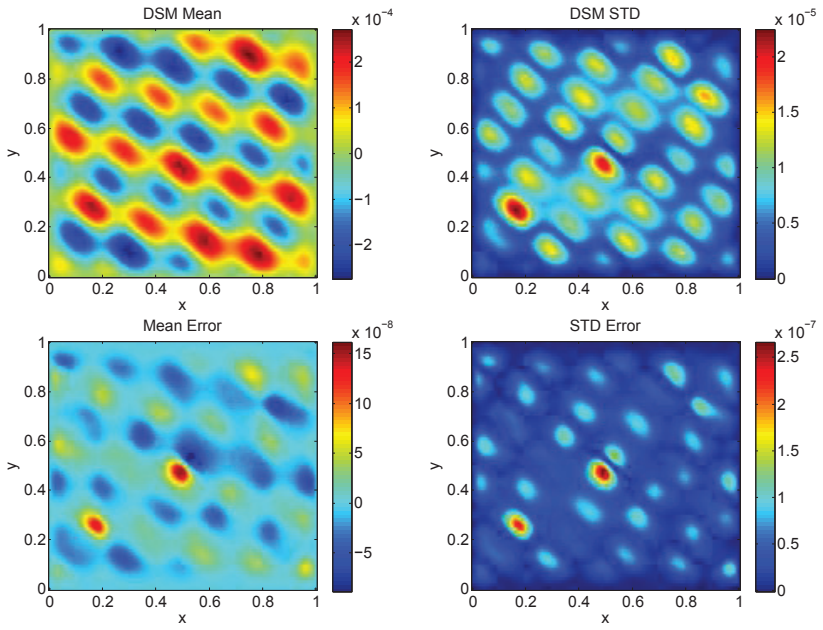


Figure 10. *Comparison of the mean and STD in the 2D SPDE with uniform random variables.*

convergence rate, its computational error does not depend on the dimension of the problem or the regularity of the stochastic solution. Therefore, the DSM-MC method will be the method of choice. In some engineering applications, the dimension of the input random space could be very large, but the effective dimension of the output random space may be small due to the fast decay of the eigenvalues of the covariance kernel. To demonstrate the effectiveness of DSM-MC in this scenario, we consider the 1D elliptic SPDE (74) with a random coefficient

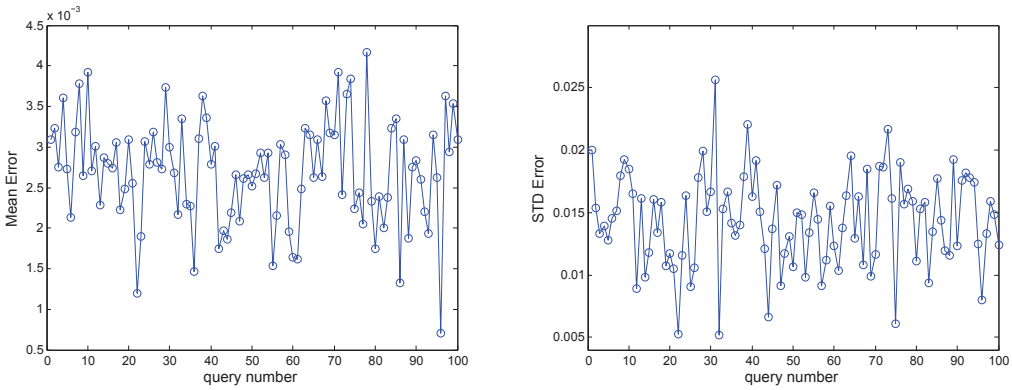


Figure 11. *DSM in MC representation for large input random space. DSM basis* $m = 16$.

given below:

$$a(x,\omega) = \sum_{m=1}^{M} m^{-\beta}\xi_m(\sin(2\pi m x + \eta_m) + 1), \tag{83}$$

where the ξ_m's and η_m's are independent uniform random variables on $[0,1]$. We choose $M = 15$ and $\beta = 0$ in the coefficient (83). We run 4×10^4 realizations of Monte Carlo samples in the offline stage to train the DSM basis. The offline training takes 641 seconds and gives 16 modes in the DSM basis. In the online stage, we randomly generate 100 force functions, i.e., $f(x) \in \{c_i \sin(k_i \pi x + \phi_i) + d_i \cos(l_i \pi x + \varphi_i)\}_{i=1}^{100}$, where c_i, k_i, ϕ_i, d_i, l_i, and φ_i are random numbers. For each query we run 10^6 realizations of Monte Carlo samples to obtain the exact solution, which takes 790 seconds. We use the DSM-MC solver to obtain the DSM solution, which takes 0.01 second. As we can see, the computational savings are huge. In Figure 11 we plot the mean and STD comparison of the DSM solution and the exact solution in the online stage. This example demonstrates that the DSM-MC basis could be very effective when the input stochastic dimension is high but the effective dimension of the output stochastic solution is small.

6. Concluding remarks. In this paper, we proposed and developed a data-driven stochastic method (DSM) to study the multiquery problem of SPDEs. Our method consists of an offline stage and an online stage. In the offline stage, a data-driven stochastic basis $\{A_i(\omega)\}_{i=1}^m$ is computed using the KL expansion and a two-level optimization approach based on multiple trial functions. In the online stage, we expand the SPDE solution under the data-driven stochastic basis and solve a set of coupled deterministic PDEs to obtain the coefficients. By exploring the low-dimensional structure of the solution, our DSM offers considerable computational saving over some traditional methods. Depending on the numerical representations of the data-driven stochastic basis $\{A_i(\omega)\}$, three versions of DSM have been proposed, i.e., DSM-MC, DSM-gSC, and DSM-gPC. They have their own advantages and disadvantages. The DSM-gSC and DSM-gPC methods depend on the multi-index of the orthonormal polynomial basis. These methods are very accurate but could be expensive when the dimension

of the input random variables is large. Under the same computational condition, the DSM-gSC can handle a larger multi-index of the orthonormal polynomial basis than the DSM-gPC method due to its nonintrusive nature. Since the stochastic basis of DSM-gSC and DSM-gPC is expanded in certain basis, we also propose an a posteriori error estimate and error correction based on the Monte Carlo method. This further improves the accuracy of the DSM method. The DSM-MC method has an advantage over DSM-gSC and DSM-gPC in the sense that its accuracy does not depend on the dimension of the input random variables or the regularity of the stochastic solution. This advantage is particularly attractive when the dimension of the input random variables is large but the effective dimension of the output stochastic solution is small. Numerical examples have been presented for both 1D and 2D elliptic PDEs with random coefficients to demonstrate the accuracy and efficiency of the proposed method.

We should point out that data-driven philosophy can be extended to the cases where the right-hand side function involves randomness, i.e., $f(x,\omega)$ and the time-dependent SPDEs. For the time-dependent SPDEs, we have recently designed a dynamically biorthogonal stochastic method to solve time-dependent SPDEs [6, 7]. An important feature of the dynamically biorthogonal stochastic method is that it offers considerable computational savings even for a single query since we compute the reduced sparse basis on the fly. When the dimension of the SPDE solution space is large, the current version of the DSM method does not offer much computational saving. We are currently developing a multiscale version of DSM to handle this class of problems.

Appendix. KL expansion of the stochastic solution in gPC basis. In this appendix, we will derive the KL expansion of the stochastic solution represented in the gPC basis. Since the stochastic basis $\mathbf{H}(\boldsymbol{\xi})$ in the gPC representation (36) is orthonormal, we can calculate the KL expansion of the solution without calculating its covariance function.

We assume the solution is given by $u(x,\omega) = \mathbf{V}(x)\mathbf{H}(\boldsymbol{\xi})^T$. By some stable orthogonalization procedures, such as the modified Gram–Schmidt process, $\mathbf{V}(x)$ has the decomposition $\mathbf{V}(x) = \mathbf{Q}(x)\mathbf{R}$, where $\mathbf{Q}(x) = (q_1(x), q_2(x), \ldots, q_{N_p}(x))$ are orthonormal bases on $L^2(D)$ and $\mathbf{R}$ is an N_p-by-N_p upper triangular matrix. $\mathbf{R}\mathbf{R}^T$ is a positive definite symmetric matrix and has the SVD $\mathbf{R}\mathbf{R}^T = \mathbf{W}\Lambda_{\mathbf{U}}\mathbf{W}^T$. Here $\Lambda_{\mathbf{U}} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{N_p})$ is a diagonal matrix, and $\mathbf{W}$ is an orthonormal matrix. Then, the covariance function can be rewritten as

$$Cov_u(x,y) = \mathbf{Q}(x)\mathbf{W}\Lambda_{\mathbf{U}}\mathbf{W}^T\mathbf{Q}^T(y).$$

It is easy to see that the eigenfunctions of covariance function $Cov_u(x,y)$ are

$$\mathbf{U}(x) = \mathbf{Q}(x)\mathbf{W}\Lambda_{\mathbf{U}}^{\frac{1}{2}}, \tag{84}$$

and the stochastic basis $A_i(\omega)$ in the KL expansion can be expressed in terms of the polynomial chaos basis,

$$A_i(\omega) = \sum_{\boldsymbol{\alpha}\in\mathfrak{J}} A_{\boldsymbol{\alpha}i}\mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)), \quad i = 1, \ldots, N_p, \tag{85}$$

where the N_p-by-N_p matrix $\mathbf{A} = (A_{\boldsymbol{\alpha}i})$ is given by

$$\mathbf{A} = \mathbf{R}^T\mathbf{W}\Lambda_{\mathbf{U}}^{-\frac{1}{2}}. \tag{86}$$

From (84)–(86) we know that $\mathbf{U}(x) = (u_1(x), u_2(x), \ldots, u_{N_p}(x))$ are orthogonal bases on $L^2(D)$, i.e.,

$$\int_D \mathbf{U}(x)^T \mathbf{U}(x) dx = \Lambda_{\mathbf{U}}, \tag{87}$$

and $\mathbf{A}$ is an N_p-by-N_p orthonormal matrix. In the gPC representation, we need only use (84) and (86) to compute the KL expansion of the stochastic solution. There is no need to form the covariance function, which greatly reduces the computational cost.

In practical computations, we sort the sequence of $\mathbf{U}(x)$ and the columns of $\mathbf{A}$ so that the eigenvalues in $\Lambda_{\mathbf{U}}$ are in a descending order. According to the eigenvalue decay property of the diagonal matrix $\Lambda_{\mathbf{U}}$, we can truncate the representation (84) and (86) into m terms and obtain the truncated KL expansion

$$u(x,\omega) \approx \sum_{i=1}^{m} \sum_{\boldsymbol{\alpha}\in\mathfrak{J}} u_i(x) A_{\boldsymbol{\alpha} i} \mathbf{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi}(\omega)) = \mathbf{U}(x)\mathbf{A}^T\mathbf{H}^T, \tag{88}$$

where $\mathbf{U}(x) = [u_1(x), \ldots, u_m(x)]$ and $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_m]$ is an N_p-by-m matrix, satisfying $\mathbf{A}^T\mathbf{A} = \mathbf{I}_m$.

REFERENCES

[1] I. Babuška, F. Nobile, and R. Tempone, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM J. Numer. Anal., 45 (2007), pp. 1005–1034.

[2] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera, *An 'empirical interpolation' method: Application to efficient reduced basis discretization of partial differential equations*, C. R. Acad. Sci. Paris Ser. I Math., 339 (2004), pp. 667–672.

[3] L. Borcea, G. Papanicolaou, C. Tsogka, and J. Berryman, *Imaging and time reversal in random media*, Inverse Problems, 18 (2002), pp. 1247–1279.

[4] H. J. Bungartz and M. Griebel, *Sparse grids*, Acta Numer., 13 (2004), pp. 147–269.

[5] R. H. Cameron and W. T. Martin, *The orthogonal development of non-linear functionals in series of Fourier-Hermite functionals*, Ann. Math., 48 (1947), pp. 385–392.

[6] M. L. Cheng, T. Y. Hou, and Z. W. Zhang, *A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations* I: *Derivation and algorithms*, J. Comput. Phys., 242 (2013), pp. 843–868.

[7] M. L. Cheng, T. Y. Hou, and Z. W. Zhang, *A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations* II: *Adaptivity and generalizations*, J. Comput. Phys., 242 (2013), pp. 753–776.

[8] M. L. Cheng, *Adaptive Methods Exploring Intrinsic Sparse Structures of Stochastic Partial Differential Equations*, Doctoral thesis, California Institute of Technology, Pasadena, CA, 2013.

[9] A. J. Chorin, *Hermite expansion in Monte-Carlo simulations*, J. Comput. Phys., 8 (1971), pp. 472–482.

[10] A. J. Chorin and X. M. Tu, *Implicit sampling for particle filters*, Proc. Natl. Acad. Sci. USA, 106 (2009), pp. 17249–17254.

[11] M. Dashti and A. M. Stuart, *Uncertainty quantification and weak approximation of an elliptic inverse problem*, SIAM J. Numer. Anal., 49 (2011), pp. 2524–2542.

[12] A. Doostan and H. Owhadi, *A non-adapted sparse approximation of PDEs with stochastic inputs*, J. Comput. Phys., 230 (2011), pp. 3015–3034.

[13] P. Dostert, Y. Efendiev, T. Y. Hou, and W. Luo, *Coarse gradient Langevin algorithms for dynamic data integration and uncertainty quantification*, J. Comput. Phys., 217 (2006), pp. 123–142.

[14] Y. Efendiev, T. Hou, and W. Luo, *Preconditioning of Markov chain Monte Carlo simulations using coarse-scale models*, SIAM J. Sci. Comput., 28 (2006), pp. 776–803.
[15] W. Feller, *An Introduction to Probability Theory and Its Application, Volume* 1, 3rd ed., John Wiley and Sons, New York, 1968.
[16] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Springer-Verlag, New York, 1991.
[17] M. B. Giles, *Multilevel Monte Carlo path simulation*, Oper. Res., 56 (2008), pp. 607–617.
[18] W. Hackbusch, *A sparse matrix arithmetic based on H-matrices. Part* I: *Introduction to H-matrices*, Computing, 62 (1999), pp. 89–108.
[19] W. Hackbusch and B. N. Khoromskij, *A sparse matrix arithmetic. Part* II: *Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47.
[20] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.
[21] T. Y. Hou, W. Luo, B. Rozovskii, and H. Zhou, *Wiener chaos expansions and numerical solutions of randomly forced equations of fluid mechanics*, J. Comput. Phys., 216 (2006), pp. 687–706.
[22] K. Karhunen, *Über lineare methoden in der Wahrscheinlichkeitsrechnung*, Ann. Acad. Sci. Fennicae. Ser. A. I. Math.-Phys., 37 (1947), pp. 1–79.
[23] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer-Verlag, Berlin, Heidelberg, 1992.
[24] L. Y. Li, H. A. Tchelepi, and D. X. Zhang, *Perturbation-based moment equation approach for flow in heterogeneous porous media: Applicability range and analysis of high-order terms*, J. Comput. Phys., 188 (2003), pp. 296–317.
[25] M. Loève, *Probability Theory, Vol.* II, 4th ed., Grad. Texts in Math. 46, Springer-Verlag, New York, 1978.
[26] W. Luo, *Wiener Chaos Expansion and Numerical Solutions of Stochastic Partial Differential Equations*, Ph.D. thesis, California Institute of Technology, Pasadena, CA, 2006.
[27] X. Ma and N. Zabaras, *An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations*, J. Comput. Phys., 228 (2009), pp. 3084–3113.
[28] O. Le Maitre, *Uncertainty propagation using Wiener-Haar expansions*, J. Comput. Phys., 197 (2004), pp. 28–57.
[29] A. J. Majda, I. Timofeyev, and E. V. Eijnden, *A mathematical framework for stochastic climate models*, Comm. Pure Appl. Math., 54 (2001), pp. 891–974.
[30] A. J. Majda and M. Branicki, *Lessons in uncertainty quantification for turbulent dynamical systems*, Discrete Contin. Dyn. Syst., 32 (2012), pp. 3133–3221.
[31] N. C. Nguyen, *A multiscale reduced-basis method for parametrized elliptic partial differential equations with multiple scales*, J. Comput. Phys., 227 (2008), pp. 9807–9822.
[32] B. K. Øksendal, *Stochastic Differential Equations: An Introduction with Applications*, 6th ed., Springer, Berlin, 2003.
[33] H. Owhadi, C. Scovel, T. J. Sullivan, M. McKerns, and M. Ortiz, *Optimal uncertainty quantification*, SIAM Rev., 55 (2013), pp. 271–345.
[34] G. Rozza, D. B. P. Huynh, and A. T. Patera, *Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. Application to transport and continuum mechanics*, Arch. Comput. Methods Eng., 15 (2008), pp. 229–275.
[35] T. P. Sapsis and P. F. J. Lermusiaux, *Dynamically orthogonal field equations for continuous stochastic dynamical systems*, Phys. D, 238 (2009), pp. 2347–2360.
[36] C. Schwab and R. A. Todor, *Karhunen-Loève approximation of random fields by generalized fast multipole methods*, J. Comput. Phys., 217 (2006), pp. 100–122.
[37] X. Wan and G. E. Karniadakis, *An adaptive multi-element generalized polynomial chaos method for stochastic differential equations*, J. Comput. Phys., 209 (2005), pp. 617–642.
[38] N. Wiener, *The homogeneous chaos*, Amer. J. Math., 60 (1938), pp. 897–936.
[39] X.-H. Wu, L. Bi, and S. Kalla, *Effective parametrization for reliable reservoir performance predictions*, Int. J. Uncertain. Quantif., 2 (2012), pp. 259–278.

[40] D. Xiu and G. E. Karniadakis, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644.
[41] D. Xiu and G. E. Karniadakis, *Modeling uncertainty in flow simulations via generalized polynomial chaos*, J. Comput. Phys., 187 (2003), pp. 137–167.
[42] D. Xiu and J. S. Hesthaven, *High-order collocation methods for differential equations with random inputs*, SIAM J. Sci. Comput., 27 (2005), pp. 1118–1139.