To appear in *Optimization Methods & Software*
Vol. 00, No. 00, Month 20XX, 1–15

Using Automatic Differentiation for Compressive Sensing in Uncertainty Quantification

Mu Wang[a], Guang Lin[b] and Alex Pothen[a*]

[a]*Department of Computer Science, Purdue University, West Lafayette, IN 47907, U.S.;*
[b]*Department of Mathematics and School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, U.S.*

(*v5.0 released July 2015*)

This paper employs Automatic Differentiation (AD) in the compressive sensing-based generalized polynomial chaos (gPC) expansion, which computes a sparse approximation of the Quantity of Interest (QoI) using orthogonal polynomials as basis functions. An earlier approach without AD relies on an iterative procedure to refine the solution by approximating the gradient of the QoI. With AD, the gradient can be accurately evaluated, and a set of basis functions of the gPC expansion associated with new random variables can be efficiently identified. The computational complexity of the algorithm using AD is independent of the number of basis functions, whereas an earlier algorithm had complexity proportional to the square of this number. Our test problems include synthetic problems and a high-dimensional stochastic partial differential equation. With the new basis, the coefficient vector in the gPC expansion is sparser than the original basis. We demonstrate that introducing AD can greatly improve the performance by computing solutions two to ten times faster than an earlier approach. The accuracy of the gPC expansion is also improved; sparse gpC expansions are obtained without iterative refinement, even for high dimensions when an earlier approach fails.

Keywords: uncertainty quantification; generalized polynomial chaos expansion; compressive sensing; Hermite polynomials; automatic differentiation; reverse mode

AMS Subject Classification: 65K; 65H

1. Introduction

In this paper we introduce Automatic Differentiation (AD) to improve the efficiency and the effectiveness of compressive sensing-based generalized polynomial chaos expansion, which has become a powerful tool in uncertainty quantification in recent years. Uncertainty quantification (UQ) characterizes uncertainties in both computational and real world applications by illuminating the effect of the uncertainties on the quantity of interest (QoI). Let ω be an event in a complete probability space Ω, let $\boldsymbol{\xi}(\omega)$ be a d-dimensional random vector $\boldsymbol{\xi}(\omega) = \big(\xi_1(\omega), \xi_2(\omega), \cdots, \xi_d(\omega)\big)$ and let the QoI be $u(\boldsymbol{\xi})$. In the remainder of the paper, we will write $\boldsymbol{\xi}$ instead of $\boldsymbol{\xi}(\omega)$ and ξ_i instead of $\xi_i(\omega)$ for simplicity. The QoI $u(\boldsymbol{\xi})$ can be approximated by a generalized polynomial chaos (gPC)

*Corresponding author. Email: apothen@purdue.edu expansion [1, 2]:

$$u(\boldsymbol{\xi}) = \sum_{n=1}^{N} c_n \psi_n(\boldsymbol{\xi}) + \epsilon(\boldsymbol{\xi}), \tag{1}$$

where $\epsilon(\boldsymbol{\xi})$ is the truncation error. Here $\{c_n\}$ are coefficients, $\{\psi_n(\boldsymbol{\xi})\}$ are the basis functions of the gPC, and N is the number of basis functions. These polynomials are orthonormal with respect to the density of $\boldsymbol{\xi}$, i.e,

$$\int_{\mathbb{R}^d} \psi_i(\boldsymbol{\xi})\psi_j(\boldsymbol{\xi})\rho(\xi)d\boldsymbol{\xi} = \delta_{ij}, \tag{2}$$

where $\rho(\boldsymbol{\xi})$ is the probability density function (PDF) of $\boldsymbol{\xi}$ and δ_{ij} is the Kronecker delta. When $u(\boldsymbol{\xi})$ is in the Hilbert space (the space of square-integrable functions) with respect to the measure of $\boldsymbol{\xi}$, the approximation converges in L_2 space as the order of the polynomials goes to infinity.

The coefficients $\mathbf{c} = (c_1, \cdots, c_N)$ can be evaluated by a stochastic collocation approach [3, 4]. The first step is to generate samples of the input $\boldsymbol{\xi}^q, q = 1, 2, \cdots, M$, based on the PDF $\rho(\boldsymbol{\xi})$. Then the values of the QoI $u^q = u(\boldsymbol{\xi}^q)$ are evaluated using a prior computational model. The coefficients $\mathbf{c}$ in the gPC expansion can be computed using $\boldsymbol{\xi}^q$ and u^q, $q = 1, 2, \cdots, M$, by solving a linear system:

$$\boldsymbol{\Psi}\mathbf{c} = \mathbf{u} + \boldsymbol{\epsilon}. \tag{3}$$

In the equation, $\mathbf{u} = (u^1, u^2, \cdots, u^M)^T$ is the vector of M output samples, $\boldsymbol{\Psi}$ is an $M \times N$ matrix with $\boldsymbol{\Psi}_{ij} = \psi_j(\boldsymbol{\xi}^i)$, and $\boldsymbol{\epsilon}$ is a vector of error samples. In many application problems, the computation model for QoI $u(\boldsymbol{\xi})$ may be expensive to evaluate, and so the number of samples M will be few. When $M < N$, the linear system Eq. 3 is under-determined. The compressive sensing approach finds a sparse solution of $\mathbf{c}$ to this under-determined linear system efficiently. Several approaches have been proposed for solving Eq. 3 in UQ applications [5–11]. In this paper, we aim to enhance one of the methods, the iterative rotation method [11], by introducing Automatic Differentiation.

The remainder of this paper is organized as follows. A brief review of the iterative rotation method is given in Section 2. In Section 3, we describe an algorithm to enhance the sparsity in the gPC expansion, in which Automatic Differentiation is employed to compute a gradient matrix. To illustrate its performance, three numerical case studies are discussed in Section 4. Conclusions and future work are included in Section 5.

2. Brief Review of The Iterative Rotation Method

2.1 *Hermite Polynomial Chaos Expansion*

We assume that the QoI $u(\boldsymbol{\xi})$ depends on d-dimensional i.i.d Gaussian random variables, i.e, $\boldsymbol{\xi} \sim N(\mathbf{0}, \mathbf{I})$. Then the basis functions will be products of univariate orthonormal Hermite polynomials. That is, each multi-index $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_d), \alpha_i \in \mathbb{N}$ corresponds to a basis function:

$$\psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \psi_{\alpha_1}(\xi_1)\psi_{\alpha_2}(\xi_2)\cdots\psi_{\alpha_d}(\xi_d), \tag{4}$$

where ψ_{α_i} is the α_i-th order Hermite polynomial,

$$\psi_{\alpha_i}(\xi_i) = \frac{(-1)^{\alpha_i}}{\sqrt{\alpha_i!}} e^{\frac{\xi_i^2}{2}} \frac{d^{\alpha_i}}{d{\xi_i}^{\alpha_i}} e^{-\frac{\xi_i^2}{2}}. \tag{5}$$

One can check that the function $\psi_{\boldsymbol{\alpha}}$ is orthogonal w.r.t the Gaussian PDF $\rho(\boldsymbol{\xi})$.

$$\int_{\mathbb{R}^d} \psi_{\boldsymbol{\alpha}_i}(\boldsymbol{\xi})\psi_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi})\rho(\boldsymbol{\xi})d\boldsymbol{\xi} = \delta_{\boldsymbol{\alpha}_i\boldsymbol{\alpha}_j} = \delta_{\alpha_{i1}\alpha_{j1}}\delta_{\alpha_{i2}\alpha_{j2}}\cdots\delta_{\alpha_{id}\alpha_{jd}}, \tag{6}$$

where

$$\rho(\boldsymbol{\xi}) = \left(\frac{1}{\sqrt{2\pi}}\right)^d exp\left(-\frac{\xi_1^2+\xi_2^2+\cdots+\xi_d^2}{2}\right). \tag{7}$$

The sum $p \equiv \sum_{i=1}^{d} \alpha_i$ is the order of the basis function $\psi_{\boldsymbol{\alpha}}$. For a given dimension d and a maximum order p, the total number of basis functions N is $\binom{d+p}{p}$.

2.2 *Compressive Sensing-based gPC method*

Before we introduce compressive sensing, we need a quantitative definition of sparsity. The coefficient vector $\mathbf{c}$ is considered sparse when it contains few nonzero entries. This can be quantified by the number of nonzeros in $\mathbf{c}$

$$\text{nnz}(\mathbf{c}) = |\{i : c_i \neq 0\}|. \tag{8}$$

The vector $\mathbf{c}$ is called *s-sparse* if the number of nonzeros is not greater than s.

The compressive sensing-based gPC method tries to find a sparse vector $\mathbf{c}$, a solution to the linear system Eq. 3. However, finding a sparsest solution $\mathbf{c}$ to this system is intractable [12]. So we relax the definition of "sparse" to signify that the vector $\mathbf{c}$ has only few entries that significantly contribute to its l_1 norm. Hence $||\mathbf{c} - \mathbf{c}_s||_1$ is small, where the vector $\mathbf{c}_s$ consists of the s entries of $\mathbf{c}$ with the largest absolute values. Candes, Wakin and Boyd [13] have proved that in many cases, solving the problem with the l_1 norm yields similar results to finding a sparsest solution.

Hence we will approximate the solution vector $\mathbf{c}$ of Eq. 3 by solving the following optimization problem:

$$\mathbf{c} = arg \min_{\hat{\mathbf{c}}} ||\hat{\mathbf{c}}||_1, \ \ s.t. \ ||\Psi\hat{\mathbf{c}} - \mathbf{u}||_2 \leq \epsilon, \tag{9}$$

where $\epsilon = ||\boldsymbol{\epsilon}||_2$. In practice, the error term ϵ is not known *a priori*, and a cross-validation procedure is needed to estimate it. One such algorithm is given in [14]. Algorithm 1 describes the compressive sensing-based gPC method.

2.3 *Iterative Rotation for Enhancing Sparsity*

The sparsity of the vector $\tilde{\mathbf{c}}$ in the compressive sensing-based gPC can be further enhanced by finding another set of random variables $\boldsymbol{\eta} = (\eta_1, \eta_2, \cdots, \eta_d)^T$, which depends

Algorithm 1 Compressive sensing-based gPC

```
1: Generate $M$ sample points $\boldsymbol{\xi}^q, q = 1, 2, \cdots, M$, based on the distribution of $\boldsymbol{\xi}$.
2: Evaluate $M$ samples $u^q = u(\boldsymbol{\xi}^q)$ via the complete model.
3: Compute gPC basis functions $\{\psi_n\}_1^N$ from the PDF $\rho(\boldsymbol{\xi})$; compute the measurement
   matrix $\Psi$ by $\Psi_{ij} = \psi_j(\boldsymbol{\xi}^i)$; compute the estimated error $\epsilon$.
4: Solve the optimization problem:
       $\mathbf{c} = arg \min_{\hat{\mathbf{c}}} ||\hat{\mathbf{c}}||_1$, subject to $||\Psi\hat{\mathbf{c}} - \mathbf{u}||_2 \leq \epsilon$.
5: Construct the gPC expansion, $u(\boldsymbol{\xi}) \approx \sum_{n=1}^N c_n\psi_n(\boldsymbol{\xi})$.
```

on $\boldsymbol{\xi}$ [11]. Hence we write

$$u(\boldsymbol{\xi}) \approx \sum_{n=1}^{N} c_n\psi_n(\boldsymbol{\xi}) = \sum_{n=1}^{N} \tilde{c}_n\tilde{\psi}_n(\boldsymbol{\eta}(\boldsymbol{\xi})) \approx u(\boldsymbol{\eta}(\boldsymbol{\xi})). \tag{10}$$

We expect the vector $\tilde{\mathbf{c}}$ to be sparser than $\mathbf{c}$. That is, for a small s, we have

$$||\tilde{\mathbf{c}} - \tilde{\mathbf{c}}_s||_1 \leq ||\mathbf{c} - \mathbf{c}_s||_1. \tag{11}$$

In this case, the s leading entries are more significant in $\tilde{\mathbf{c}}$ than in $\mathbf{c}$, since the l_1 norm of the remaining entries are smaller. Denote the mapping from $\boldsymbol{\xi}$ to $\boldsymbol{\eta}$ by $g : \mathbb{R}^d \to \mathbb{R}^d$, $\boldsymbol{\eta} = g(\boldsymbol{\xi})$. We've assumed that $\boldsymbol{\xi} \sim N(\mathbf{0}, \mathbf{I})$; let us further assume $\boldsymbol{\eta}$ to be i.i.d Gaussian random variables also, i.e, $\boldsymbol{\eta} \sim N(\mathbf{0}, \mathbf{I})$ to reduce the complexity. Then, the mapping g is a linear transformation

$$\boldsymbol{\eta} = g(\boldsymbol{\xi}) = A\boldsymbol{\xi}, \tag{12}$$

where A is an orthonormal matrix. In order to compute the matrix A, we adopt the active subspace approach of Constantine, Dow and Wang [15, 16]. First, a "gradient matrix" G is defined and its eigen-decomposition is computed:

$$G = \mathbb{E}\Big[\nabla u(\xi) \cdot \nabla u(\xi)^T\Big] = U\Lambda U^T, UU^T = I, \tag{13}$$

where $\nabla u(\xi)$ is the gradient vector of $u(\xi)$, G is symmetric, and $U\Lambda U^T$ is the eigen-decomposition of G. The eigenvalue λ_i represents the expectation of the l_2 norm of the directional derivative along the direction specified by the corresponding eigenvector U_i. Consequently, when there is a large gap in the spectrum, the dependence of $u(\boldsymbol{\xi})$ on $\boldsymbol{\xi}$ is primarily determined by the projection of $\boldsymbol{\xi}$ on a subspace spanned by the eigenvectors corresponding to the dominant eigenvalues. The subspace determined by these eigenvectors is called the *active subspace*. In [15, 16], a low-rank approximation of $u(\boldsymbol{\xi})$ is constructed by restricting the dependence of $u(\boldsymbol{\xi})$ only on the active subspace.

Here, in the context of compressive sensing-based gPC, the mapping $\boldsymbol{\eta} = g(\boldsymbol{\xi}) = A\boldsymbol{\xi}$ can be defined by choosing $A = U^T$ [11]. Then the dependence of u on an eigenvector U_i is transformed to the dependence on the new random variable η_i. Since the first few variables η_i correspond to the active subspace of $u(\boldsymbol{\xi})$, we expect $\tilde{c}$ to be sparser than c.

In [11], the gradients are approximated by differentiating the current polynomial approximation the QoI.

$$G \approx \mathbb{E}\Bigg[\nabla\Big(\sum_{n=1}^{N} c_n\psi_n(\boldsymbol{\xi})\Big)\cdot\nabla\Big(\sum_{n'=1}^{N} c_{n'}\psi_{n'}(\boldsymbol{\xi})\Big)^T\Bigg]. \tag{14}$$

The algorithm begins with the solution $\mathbf{c}$ of the Algorithm 1. Then using Eq. 15 (see below), it approximates the gradient matrix G and finds a better solution $\tilde{\mathbf{c}}$ of the compressive sensing-based gPC. This procedure is iteratively repeated until a satisfactory solution is found. Algorithm 2 summarizes the iterative procedure.

Algorithm 2 Compressive sensing-based gPC with iterative rotations

```
1: Call Algorithm 1 to get the initial gPC coefficients $\mathbf{c}$.
2: $k = 0$, $\boldsymbol{\eta}^{(0)} = \boldsymbol{\xi}$, $\tilde{\mathbf{c}}^{(0)} = \mathbf{c}$.
3: Compute $G^{(k+1)}$ according to Eq. 15; then compute the eigen-decomposition of $G$ as :
       $G^{(k+1)} = U^{(k+1)}\Lambda^{(k+1)}\big(U^{(k+1)}\big)^T, U^{(k+1)}\big(U^{(k+1)}\big)^T = I.$
4: Use the relation $\eta^{(k+1)} = \big(U^{(k+1)}\big)^T\eta^{(l)}$ to compute samples $\big(\eta^{(k+1)}\big)^q = \big(U^{(k+1)}\big)^T\big(\eta^{(k)}\big)^q, q = 1, 2, \cdots, M$, and the new measurement matrix $\Psi^{(k+1)}$ as $\Psi_{ij}^{(k+1)} = \psi_j\big((\boldsymbol{\eta}^{(k+1)})^i\big)$. Update the estimated error $\epsilon^{(k+1)}$.
5: Solve the optimization problem:
       $\tilde{\mathbf{c}}^{(k+1)} = arg\min_{\hat{\mathbf{c}}} ||\hat{\mathbf{c}}||_1$, subject to $||\Psi^{(k+1)}\hat{\mathbf{c}} - \mathbf{u}||_2 \le \epsilon^{(k+1)}$.
6: Stop if $\Big| ||U^{(k+1)}||_1 - d \Big| \le \theta$ or maximum number of iterations is reached. Otherwise, set $k = k + 1$ and go to step 3.
```

In Algorithm 2, one termination condition is based on the l_1 norm of the rotation matrix $U^{(k+1)}$, which is satisfied when $U^{(k+1)}$ is close to the identity matrix. Another termination condition is to set a maximum number of iterations. In practice, these two conditions are used together. Also the estimated error $\epsilon^{(k+1)}$ is updated in each iteration. The gradient matrix G is computed from

$$G_{ij} = \sum_{n=1}^{N}\sum_{n'=1}^{N} c_n c_{n'}\mathbb{E}\Big[\frac{\partial\psi_n(\boldsymbol{\xi})}{\partial\xi_i}\cdot\frac{\partial\psi_{n'}(\boldsymbol{\xi})}{\partial\xi_j}\Big] = \boldsymbol{c}^T K_{ij}\boldsymbol{c}, \tag{15}$$

where K_{ij} is a "stiffness" matrix with entries

$$(K_{ij})_{kl} = \mathbb{E}\Big[\frac{\partial\psi_k(\boldsymbol{\xi})}{\partial\xi_i}\cdot\frac{\partial\psi_l(\boldsymbol{\xi})}{\partial\xi_j}\Big]. \tag{16}$$

The entity K itself can be considered as a 4-dimensional tensor, whose size depends on the number of basis functions N, and the dimension of the problem d. The orthogonality of the Hermite polynomials ψ_n allows K_{ij} to be pre-determined.

3. Enhancing Sparsity in gPC Expansion with Automatic Differentiation

In Algorithm 2, the key idea is to use the gradient matrix G to find a sparser basis for the gPC expansion. The iterative rotation procedure is introduced to refine the approximation of the gradient matrix G. The gradient of the gPC expansion $\sum_{n=1}^{N} c_n \psi_n(\boldsymbol{\xi})$ is used as an approximation of the gradient of the QoI $u(\boldsymbol{\xi})$.

However, in many problems the QoI $u(\boldsymbol{\xi})$ is obtained from a computational model, i.e, it is evaluated by executing a computer program. Then applying Automatic Differentiation (AD) can analytically compute the gradient of $u(\boldsymbol{\xi})$ at any point. Here we use the reverse mode of AD since the QoI $u(\boldsymbol{\xi})$ is a scalar function and we need its gradient $\nabla u(\boldsymbol{\xi})$.

Automatic Differentiation is a technique that augments a computer program to evaluate the objective function as well as its gradient (and higher order derivatives) [17]. AD decomposes the original program into a sequence of elemental function evaluations that reflects how the function is computed by using the independent variables and intermediate variables. The chain rule of calculus is applied to this sequence of elemental functions to compute derivatives. The major benefit of AD is that it evaluates the derivatives to machine precision accuracy with only a small constant factor of overhead relative to evaluating the objective function itself. AD has two major modes: forward mode and reverse mode. The forward mode of AD evaluates the directional derivative of the objective function; the reverse mode of AD evaluates the gradient of a scalar objective function. More details about AD can be found in the textbooks [17, 18]; here we use it as an external tool. For this problem, the independent variables are the random variables $\xi_1, \cdots, \xi_d$. The only dependent variable is the QoI $u(\boldsymbol{\xi})$. So, the reverse mode of AD can evaluate the gradient vector within a constant factor of the cost of the QoI evaluation, with the constant bounded by three [17, 18].

With the help of AD, we can analytically evaluate the gradients to replace the iterative rotation procedure. The gradient matrix G is defined as an expectation in Eq. 13, which can be computed via the Monte Carlo method. Recall that we need M sample points $\boldsymbol{\xi}^1, \boldsymbol{\xi}^2, \cdots, \boldsymbol{\xi}^M$ to evaluate the QoI as $u^q = u(\boldsymbol{\xi}^q)$, for $q = 1, 2, \cdots, M$. We use the same M sample points and apply reverse mode AD on the procedure for evaluating the QoI $u(\boldsymbol{\xi})$ to obtain also the gradients $\nabla u(\boldsymbol{\xi}^q), q = 1, 2, \cdots, M$. Then the gradient matrix G could be computed as:

$$G \approx \frac{1}{M} \sum_{q=1}^{M} \nabla u(\boldsymbol{\xi}^q) \cdot \Big(\nabla u(\boldsymbol{\xi}^q)\Big)^T. \tag{17}$$

However, since we directly evaluate the gradients on the sample points $\nabla u(\boldsymbol{\xi}^1), \cdots, \nabla u(\boldsymbol{\xi}^M)$, we can use the singular-value decomposition instead of the eigenvalue decomposition to achieve better numerical stability in the algorithm. The singular-value decomposition-based approach has been used in related work such as [15, 16]. A matrix C consisting of all the sampled gradients is defined as:

$$C = \frac{1}{\sqrt{M}} [\nabla u(\boldsymbol{\xi}^1) \cdots \nabla u(\boldsymbol{\xi}^M)]. \tag{18}$$

Then the singular-value decomposition of C is computed,

$$C = U\sqrt{\Lambda}V. \tag{19}$$

Computing the singular-value decomposition of C is numerically more stable than computing the matrix $G = CC^T$ and then its eigen-decomposition. Mathematically, both methods give the same transformation matrix U. Algorithm 3 describes the compressive sensing method that employs AD.

Introducing AD has several advantages. First of all, it evaluates the gradient of the QoI $u(\boldsymbol{\xi})$ analytically, and then the gradients at the sample points can be used to compute the transformation from $\boldsymbol{\xi}$ to η. This removes the iterative refinement needed to compute the gradient by differentiating a best polynomial approximation to the QoI available at each iteration. Hence we expect this methodology to provide a more accurate gPC expansion.

Secondly, since the QoI $u(\boldsymbol{\xi})$ is a scalar function, the overhead of using reverse mode AD to evaluate the gradient vector is only a constant factor more expensive than evaluating the QoI. To be more specific, we pay $O(M \cdot Eval(u))$ computation to evaluate the gradients using AD, where $Eval(u)$ is the cost of evaluating the quantity of interest $u(\boldsymbol{\xi})$. The number of samples M increases asymptotically only at a rate $\log(d)$ [19] for the Monte Carlo method to get a good approximation. This complexity does not depend on the order of the Hermite polynomials. On the other hand, the iterative rotation method needs $O(d^2N^2)$ operations in each iteration for Eq. 15. The "stiffness" matrix Eq. 16 that needs to be stored has size $O(d^2N^2)$. Considering that N increases exponentially with the order of the polynomials p, the complexity and storage requirements of the iterative rotation method prohibit it from scaling to high order polynomials.

Algorithm 3 Compressive sensing method with Automatic Differentiation

```
1: Given a given random vector $\xi$ and the quantity of interest $u$, collect $M$ samples $u^1, u^2, \cdots, u^M$.
2: While sampling, using AD to compute $\nabla u(\boldsymbol{\xi}^1), \nabla u(\boldsymbol{\xi}^2), \cdots, \nabla u(\boldsymbol{\xi}^M)$.
3: Construct $C = \frac{1}{\sqrt{M}}[\nabla u(\boldsymbol{\xi}^1) \cdots \nabla u(\boldsymbol{\xi}^M)]$. Then compute the singular value decomposition $C = U\sqrt{\Lambda}V$.
4: Define $\eta = U^T\xi$, and compute samples $\eta^q = U^T\xi^q, 1 \le q \le M$. Also, construct the new measurement matrix $\boldsymbol{\Psi}$ with $\Psi_{ij} = \psi_j(\eta^i)$, and compute the estimated error $\epsilon$.
5: Solve the optimization problem to obtain $\mathbf{c}$:
       $\mathbf{c} = \arg\min_{\mathbf{c}} ||\mathbf{c}||_h$, subject to $||\boldsymbol{\Psi}\mathbf{c} - u|| \le \epsilon$.
```

4. Numerical Results and Performance Evaluation

In this section, we evaluate the performance of the compressive sensing-based gPC with iterative rotation and AD. For the iterative rotation method, we use the implementation in Yang et al. [11] and set the maximum number of iterations to be 9. For the reverse mode of AD, we use the implementation in `ReverseAD` [20]. For both methods, we set a threshold δ to be 10^{-5} such that any coefficient smaller than δ in the gPC will be considered as zero. The number of nonzeros in the solution vector $\mathbf{c}$ indicates the sparsity, or effectiveness of the compressive sensing method. For accuracy of the method, we compare the mean and the variance of the QoI $u(\boldsymbol{\xi})$ and the gPC expansion $\sum_{n=1}^{N} c_n\psi_n(\boldsymbol{\xi})$. The mean and the variance of the QoI $u(\boldsymbol{\xi})$ are computed by drawing a large number (100,000) of samples from the distribution $\rho(\boldsymbol{\xi})$. The gPC expansion has the property that the first coefficient c_1 is the mean, and the sum of squares of the other coefficients $\sum_{n=2}^{N} c_n^2$ is the variance, of the random variable it represents.

4.1 *Synthetic function with compressibility*

Our first test set consists of two synthetic functions, which are both compressive, i.e, u can be represented with only a few nonzero coefficients under another basis $\boldsymbol{\eta}$. The first problem (F1) has equally weighted random variables:

$$u(\xi) = \frac{\sum_{i=1}^{d} \xi_i + 0.25\Big(\sum_{i=1}^{d} \xi_i\Big)^2 + 0.025\Big(\sum_{i=1}^{d} \xi_i\Big)^3}{d}, \tag{20}$$

and the second problem (F2) has random variables ξ_i with decreasing weights.

$$u(\xi) = \sum_{i=1}^{d} \frac{\xi_i}{d} + 0.25\Big(\sum_{i=1}^{d} \frac{\xi_i}{\sqrt{i \times d}}\Big)^2. \tag{21}$$

Both functions have closed form expressions for the gradient. We use them here only to show the effectiveness of AD. One can expect that AD is the technique to use in cases when the gradient does not have a closed form expression or it is not easily written down in a program.

We set the order of the gPC expansion p, i.e, the maximum order of $\psi_{\boldsymbol{\alpha}}$, to be 3. We set values of the dimension d to 12 and 30, which corresponds to the number of basis functions equal to 455 and 5456, respectively. We fix the number of samples M to be 360, and thus the problem is under-determined. The results of the two methods are summarized in Table 1. First, we see that by applying AD, the performance is improved as the runtime is reduced by a factor of three to four. Secondly, when $d = 12$, both methods get almost the same accuracy. That is, the random variables generated by the gPC expansion have the same mean and variance with the QoI $u(\boldsymbol{\xi})$. When $d = 30$, the AD-based method computes good approximations of the QoI $u(\boldsymbol{\xi})$ for both F1 and F2. But the iterative rotation method failed on F1. Figure 1 plots the error in the mean and variance of F1 for both methods with dimension d from 10 to 30 respectively. As the problem becomes more under-determined, the errors get larger. The error of the iterative rotation method has a big jump around $d = 21$, but the error of the AD-based method remains below 0.02 for d up to 30, which demonstrates the robustness of the proposed AD-based method. Figure 2 gives the errors in the mean and variance of F2. Both methods compute gPC expansions with very small errors.

Both synthetic functions F1 and F2 are within the space of the gPC expansion, and hence an exact gPC expansion without truncation error exists for both functions. Moreover, there is a transformation that can map the dependence of $u(\boldsymbol{\xi})$ onto only the first variable η_1 for F1, that is, $u(\boldsymbol{\xi}) = u(\eta_1(\boldsymbol{\xi}))$ where $\eta_1 = \frac{1}{\sqrt{d}} \sum_{i=1}^{d} \xi_i$. Then the synthetic function F1 becomes a third order polynomial in the variable η_1 alone. So the gPC expansion w.r.t the new variables η will only have four nonzero coefficients. Similarly, we can argue that the dependence of $u(\boldsymbol{\xi})$ for F2 can be transformed onto two variables η_1, η_2, and since F2 is a second order polynomial, so the exact gPC expansion will only have six nonzero coefficients. The results in Table 1 show that the method employing AD can find these exact gPC expansion solutions for both functions F1 and F2. The analytical gradient matrix computed by AD can capture this information, whereas the iterative rotation method fails to do so, especially when the dimension is high, i.e, $d = 30$.

Table 1. Numerical results for the compressive sensing-based gPC methods with AD and with iterative rotation on the two synthetic functions.

	$d = 12$			$d = 30$		
		F1	F2		F1	F2
QoI $u(\boldsymbol{\xi})$	Mean	0.2486	0.0638	Mean	0.2470	0.0326
	Variance	0.4696	0.0917	Variance	0.5673	0.0353
With AD	Time(s)	30.17	13.13	Time(s)	171.56	136.92
	nnz($\mathbf{c}$)	4	6	nnz($\mathbf{c}$)	4	6
	Mean	0.2500	0.0646	Mean	0.2500	0.0333
	Variance	0.4708	0.0917	Variance	0.5896	0.0355
Iterative Rotation	Time(s)	41.14	18.88	Time(s)	683.50	442.69
	nnz($\mathbf{c}$)	4	16	nnz($\mathbf{c}$)	513	241
	Mean	0.2500	0.0646	Mean	0.1514	0.0330
	Variance	0.4708	0.0917	Variance	0.2276	0.0353

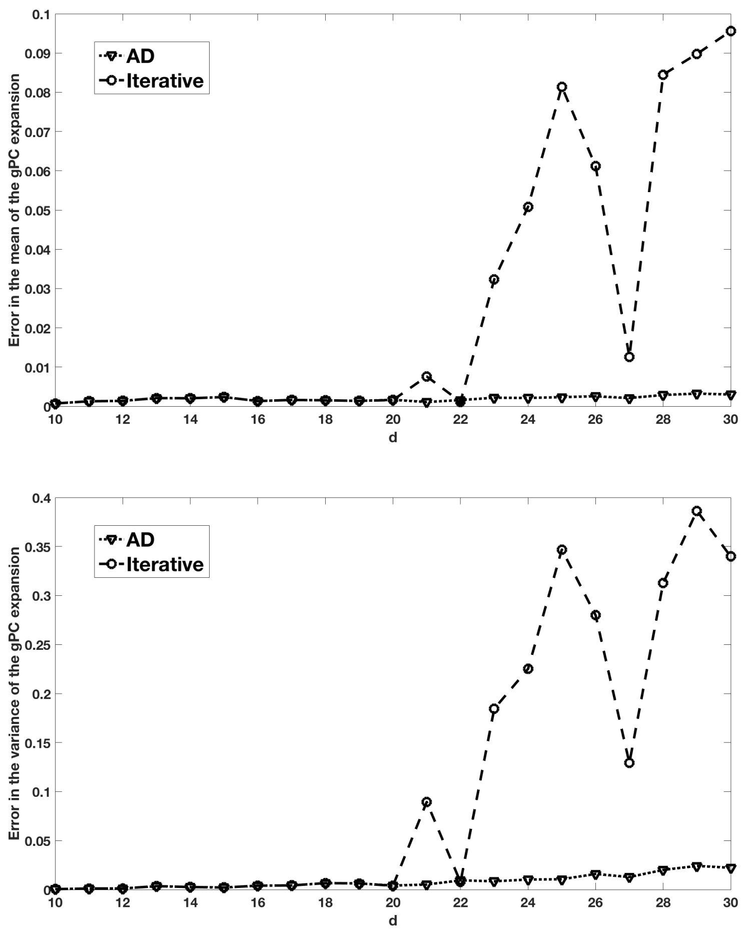


Figure 1. Errors in the mean (top) and variance (bottom) of both methods for problem F1 with dimensions from 10 to 30.

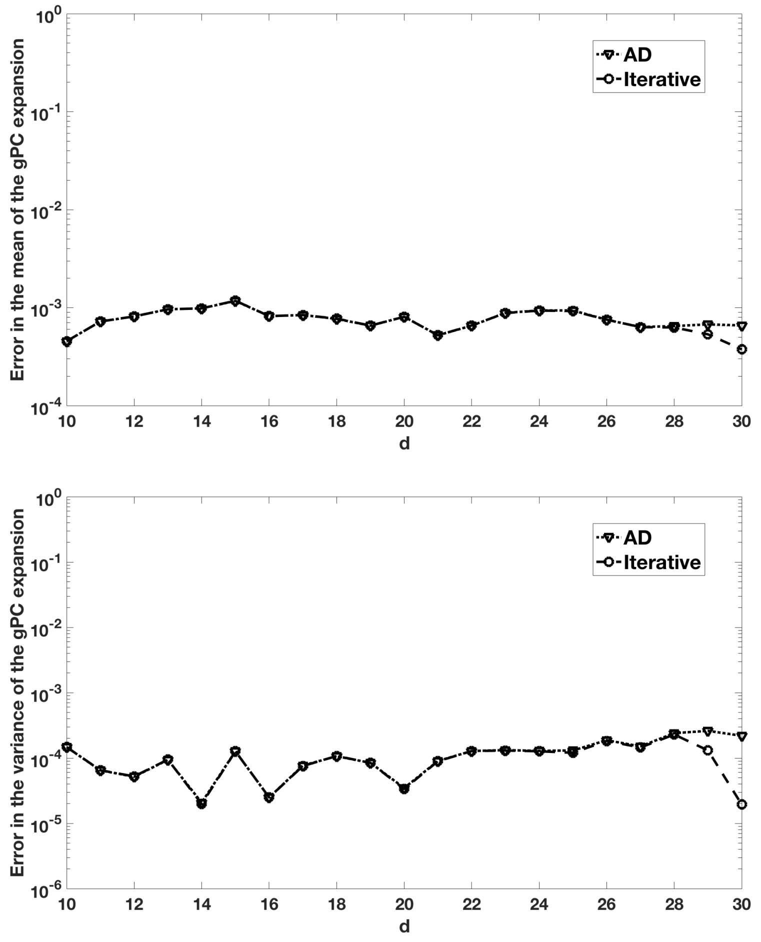


Figure 2. Errors in the mean (top) and variance (bottom) of both methods for problem F2 with dimensions from 10 to 30.

4.2 *Korteweg-de Vries equation*

To apply our new method to a more complicated and nonlinear differential equation, we consider the Korteweg-de Vries (KdV) equation with time-dependent additive noise:

$$\begin{aligned} &u_t(x,t;\boldsymbol{\xi}) - 6u(x,t;\boldsymbol{\xi})u_x(x,t;\boldsymbol{\xi}) + u_{xxx}(x,t;\boldsymbol{\xi}) = f(t;\boldsymbol{\xi}), x \in (-\infty,\infty), \\ &u(x,0;\boldsymbol{\xi}) = -2sech^2(x). \end{aligned} \tag{22}$$

Let

$$W(t;\boldsymbol{\xi}) = \int_0^t f(y;\boldsymbol{\xi})dy, \tag{23}$$

then the analytical solution of Eq. 22 can be written as:

$$u(x,t;\boldsymbol{\xi}) = W(t;\boldsymbol{\xi}) - 2sech^2\left(x - 4t + 6\int_0^t W(z;\boldsymbol{\xi})dz\right). \tag{24}$$

Table 2. Numerical results for the compressive sensing-based gPC methods with AD and with iterative rotation on the Korteweg-de Vries equation.

		$d = 10$			$d = 20$
		$p = 4$	$p = 5$		$p = 4$
QoI $u(\boldsymbol{\xi})$	Mean	-0.3976		Mean	-0.2998
	Variance	0.7314		Variance	0.7210
With AD	Time(S)	143.2	288.0	Time(S)	171.6
	Mean	-0.3870	-0.3918	Mean	-0.2822
	Variance	2.0586	0.9947	Variance	0.5333
Iterative Rotation	Time(s)	311.4	953.3	Time(s)	2,401
	Mean	-0.3935	-0.4003	Mean	-0.2611
	Variance	2.0765	0.9615	Variance	0.4478

In this experiment, we use a linear combination of random Gaussian fields defined by

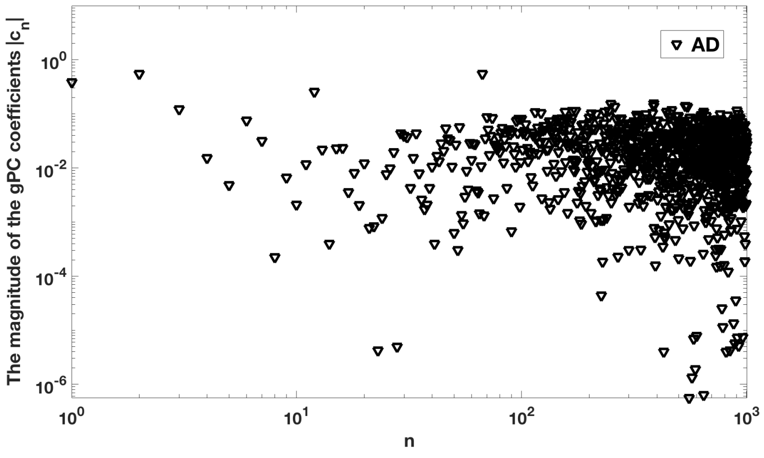
$$f(t;\boldsymbol{\xi}) = \sigma \sum_{i=1}^{d} \sqrt{\lambda_i}\phi_i(t)\xi_i, \tag{25}$$

where σ is a constant and $\{\lambda_i, \phi_i(t)\}_{i=1}^{d}$ are the leading eigenpairs of the Karhunen-Loeve (KL) expansion kernel [21]:

$$C(x, x') = exp\left(\frac{|x - x'|}{l_c}\right). \tag{26}$$

In this application we set $l_c = 0.25$, $\sigma = 0.1$ and choose the QoI to be $u(x, t; \boldsymbol{\xi})$ at $x = 6, t = 1$. We use three settings of the Hermite basis : $p = 4$, $d = 10$; $p = 4$, $d = 20$; and $p = 5$, $d = 10$. The number of basis functions are $N = 1,001$; $N = 10,626$; and $N = 3,003$; respectively, for the three settings. The results are summarized in Table 2.

We make the following observations about the results. Firstly, the method using AD is faster compared with the iterative rotation method, with the performance gap increasing with the number of dimensions. When $d = 10$, the speedup is between two and three, and when $d = 20$, the speedup is more than fourteen. Secondly, both methods give good approximations of the mean of the distribution of the QoI, but fail to provide good approximations to the variance. Part of the reason is that the QoI cannot be exactly represented by the basis functions (Hermite polynomials of order up to four), and thus the truncation error is not negligible. As shown in the table when $d = 10$, the error in the variance is reduced when the order of the Hermite polynomial increases from 4 to 5. Also for the same reason, the gPC coefficients computed by both methods are dense. Figure 3 plots the magnitudes of the gPC coefficients computed by the two methods for the setting $p = 4$ and $d = 10$. It shows similar distributions for the two solutions. Figure 4 plots the l_1 norms of the first s leading terms $||\mathbf{c}_s||_1$ and the remainder terms $||\mathbf{c} - \mathbf{c}_s||_1$ of the gPC coefficients under the same setting. The result shows that the gPC coefficients computed with AD have larger values for the ten largest terms in the expansion, while the l_1 norm of the remainder terms is smaller. This is because an accurate estimation of the gradients using AD helps to concentrate the dependence of u on a few basis functions, which further enhances the sparsity. For example, in the setting $d = 10$ and $p = 4$, only



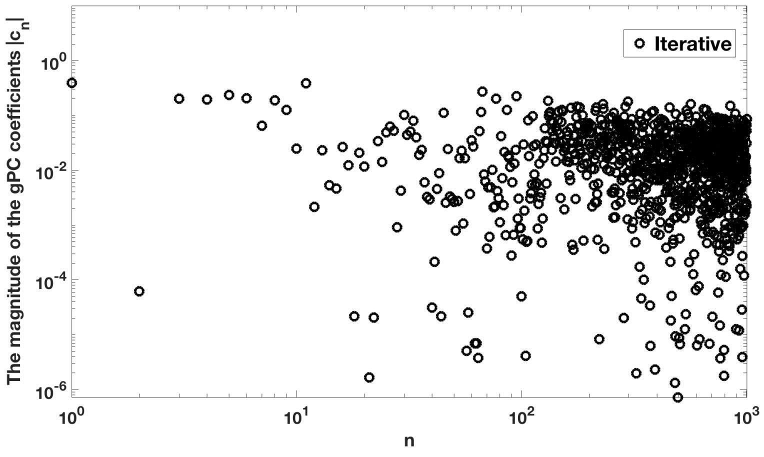


Figure 3. The magnitude of the gPC coefficients computed with AD (top) and iterative method (bottom) for the KdV equation with $p = 4$, $d = 10$.

the first two singular values (0.1117, 0.0149) are greater than 10^{-6}.

5. Conclusion and Future Work

In this paper we studied the effectiveness of Automatic Differentiation in compressive sensing-based uncertainty quantification. The iterative rotations method computes the gradients by differentiating the best polynomial approximation available at each iteration, and then it employs an iterative refinement process to achieve convergence, whereas the gradients are evaluated directly by reverse mode AD when a computational model for the QoI is available and it is differentiable. This replaces an $O(d^2N^2)$ computation per iteration with a global $O(M \cdot Eval(u))$ computation and removes the iterative refinement procedure (where $Eval(u)$ is the cost of computing the QoI $u(\boldsymbol{\xi})$). AD also evaluates the gradient matrix more accurately than the iterative rotation method. The numerical results demonstrate that with AD, we can increase the performance, sparsity, and accuracy of the Hermite polynomial expansion that represents the QoI.

These results are based on the property of AD that it computes the gradient within machine precision at a cost of a constant factor overhead of evaluating the QoI $u(\boldsymbol{\xi})$

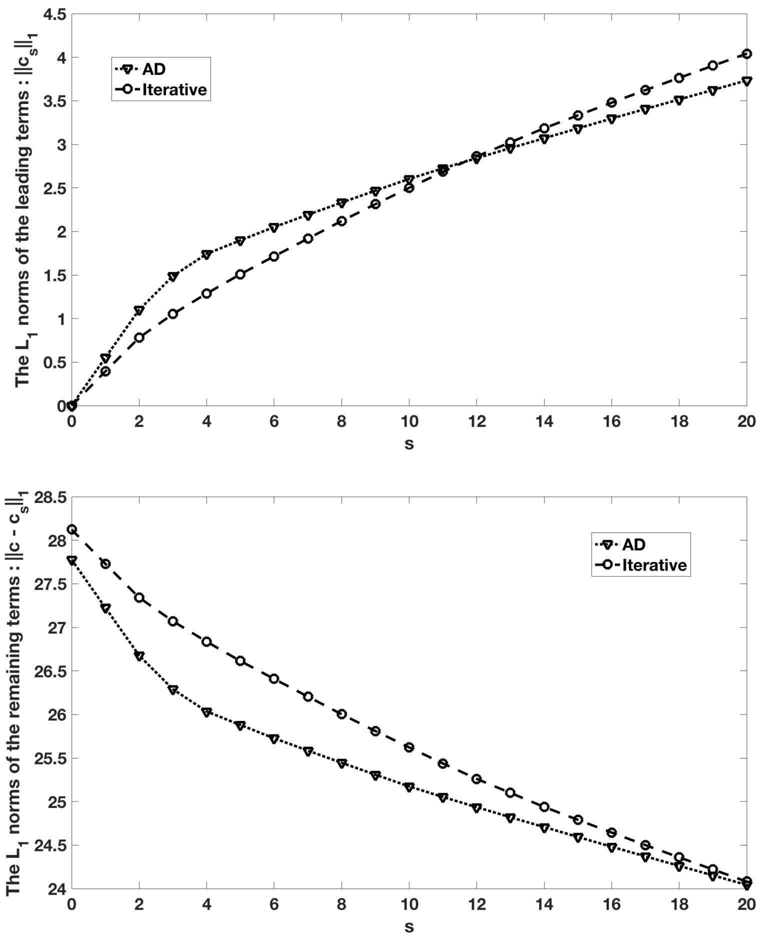


Figure 4. The l_1 norms of the first s leading terms $||\mathbf{c}_s||_1$ (top) and the remaining terms $||\mathbf{c} - \mathbf{c}_s||_1$ (bottom) of the gPC coefficients for the KdV equation with $p = 4$, $d = 10$.

itself. The limitation is that it requires the QoI $u(\boldsymbol{\xi})$ to be evaluated using a computer program, which is available only when there is a computational model for it. In this work, we restrict both $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ to be i.i.d Gaussian random variables, so that the transformation $\boldsymbol{\eta} = g(\boldsymbol{\xi})$ is a linear orthogonal transformation. When such constraints are relaxed, i.e, when $g(\boldsymbol{\xi})$ is nonlinear, we expect that introducing AD would be even more advantageous.

When AD is employed, we can consider how to improve the accuracy of gPC expansion while using fewer sample points. Some work in this direction has been done in [22]. We can also consider the possibility of high order gPC expansions, where we approximate the values of the QoI and also the derivatives, as:

$$\begin{aligned} u(\boldsymbol{\xi}) &\approx \sum_{n=1}^{N} c_n \psi_n(\boldsymbol{\xi}), \\ \frac{\partial u(\boldsymbol{\xi})}{\partial \xi_i} &\approx \sum_{n=1}^{N} c_n \frac{\partial \psi_n(\boldsymbol{\xi})}{\partial \xi_i}. \end{aligned}$$

Acknowledgments

We thank Dr. Paul Constantine for his comments on this manuscript. The work of A. Pothen and M. Wang was supported by the U.S. Department of Energy's Office of Advanced Scientific Computing Research (Office of Science) through grant DE-SC 00010205, and by NSF grant CCF-1552323. G. Lin's work was supported by NSF Grant DMS-1555072.

References

[1] Roger G Ghanem and Pol D Spanos. *Stochastic Finite Elements: a Spectral Approach.* Courier Corporation, 2003.

[2] Dongbin Xiu and George Em Karniadakis. The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, 2002.

[3] Dongbin Xiu and Jan S Hesthaven. High-order Collocation Methods for Differential Equations with Random Inputs. *SIAM Journal on Scientific Computing*, 27(3):1118–1139, 2005.

[4] Ivo Babuška, Fabio Nobile, and Raul Tempone. A Stochastic Collocation Method for Elliptic Partial Differential Equations with Random Input Data. *SIAM Journal on Numerical Analysis*, 45(3):1005–1034, 2007.

[5] Xiu Yang and George Em Karniadakis. Reweighted L1 Minimization Method for Stochastic Elliptic Differential Equations. *Journal of Computational Physics*, 248:87–108, 2013.

[6] Georgios Karagiannis, Bledar A Konomi, and Guang Lin. A Bayesian Mixed Shrinkage Prior Procedure for Spatial–Stochastic Basis Selection and Evaluation of gPC Expansions: Applications to Elliptic SPDEs. *Journal of Computational Physics*, 284:528–546, 2015.

[7] Huan Lei, Xiu Yang, Bin Zheng, Guang Lin, and Nathan A Baker. Quantifying the Tnfluence of Conformational Uncertainty in Biomolecular Solvation. *arXiv preprint arXiv:1408.5629*, 2014.

[8] Ji Peng, Jerrad Hampton, and Alireza Doostan. A Weighted L1-Minimization Approach for Sparse Polynomial Chaos Expansions. *Journal of Computational Physics*, 267:92–111, 2014.

[9] John D Jakeman, Michael S Eldred, and Khachik Sargsyan. Enhancing L1-minimization Estimates of Polynomial Chaos Expansions Using Basis Selection. *Journal of Computational Physics*, 289:18–34, 2015.

[10] Khachik Sargsyan, Cosmin Safta, Habib N Najm, Bert J Debusschere, Daniel Ricciuto, and Peter Thornton. Dimensionality reduction for complex models via Bayesian compressive sensing. *International Journal for Uncertainty Quantification*, 4(1), 2014.

[11] Xiu Yang, Huan Lei, Nathan A Baker, and Guang Lin. Enhancing Sparsity of Hermite Polynomial Expansions by Iterative Rotations. *Journal of Computational Physics*, 307:94–109, 2016.

[12] Alfred M Bruckstein, David L Donoho, and Michael Elad. From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images. *SIAM Review*, 51(1):34–81, 2009.

[13] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing Sparsity by Reweighted L1 Minimization. *Journal of Fourier Analysis and Applications*, 14(5-6):877–905, 2008.

[14] Alireza Doostan and Houman Owhadi. A non-adapted sparse approximation of PDEs with stochastic inputs. *Journal of Computational Physics*, 230(8):3015–3034, 2011.

[15] Paul G Constantine, Eric Dow, and Qiqi Wang. Active Subspace Methods in Theory and Practice: Applications to Kriging Surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[16] Paul G Constantine. *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies.* SIAM, 2015.

[17] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* SIAM, 2008.

[18] Uwe Naumann. *The Art of Differentiating Computer Programs: An Introduction to Algorithmic Differentiation*, volume 24. SIAM, 2012.

[19] Alex Gittens and Joel A Tropp. Tail Bounds for All Eigenvalues of a Sum of Random Matrices. *arXiv preprint arXiv:1104.4513*, 2011.

[20] Mu Wang. *https://github.com/wangmu0701/ReverseAD.*

[21] Dongbin Xiu. *Numerical Methods for Stochastic Computations: a Spectral Method Approach.* Princeton University Press, 2010.

[22] Ji Peng, Jerrad Hampton, and Alireza Doostan. On polynomial chaos expansion via gradient-enhanced L1-minimization. *Journal of Computational Physics*, 310:440–458, 2016.