Computer Science Technical Report TR-07-18 May 08, 2007

Haiyan Cheng and Adrian Sandu

" Efficient Uncertainty Quantification with Polynomial Chaos for Implicit Stiff Systems"

Computer Science Department
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060
Phone: (540)-231-2193
Fax: (540)-231-6075
Email: sandu@cs.vt.edu
http://eprints.cs.vt.edu

Efficient Uncertainty Quantification with Polynomial Chaos for Implicit Stiff Systems

Haiyan Cheng and Adrian Sandu

Abstract

The polynomial chaos method has been widely adopted as a computationally feasible approach for uncertainty quantification. Most studies to date have focused on non-stiff systems. When stiff systems are considered, implicit numerical integration requires the solution of a nonlinear system of equations at every time step. Using the Galerkin approach, the size of the system state increases from n to $S \times n$, where S is the number of the polynomial chaos basis functions. Solving such systems with full linear algebra causes the computational cost to increase from $O(n^3)$ to $O(S^3 n^3)$. The S^3-fold increase can make the computational cost prohibitive. This paper explores computationally efficient uncertainty quantification techniques for stiff systems using the Galerkin, collocation and collocation least-squares formulations of polynomial chaos. In the Galerkin approach, we propose a modification in the implicit time stepping process using an approximation of the Jacobian matrix to reduce the computational cost. The numerical results show a run time reduction with a small impact on accuracy. In the stochastic collocation formulation, we propose a least-squares approach based on collocation at a low-discrepancy set of points. Numerical experiments illustrate that the collocation least-squares approach for uncertainty quantification has similar accuracy with the Galerkin approach, is more efficient, and does not require any modifications of the original code.

1 Introduction

With the increase of complexity of physical models used in scientific and engineering studies, it becomes increasingly more important to quantify uncertainties associated with model predictions. In general, these uncertainties are divided into two categories: aleatory (random) and epistemic (subjective) [1]. The epistemic uncertainties come from the model itself, they can be in the initial conditions, boundary conditions or model parameters, etc. Many uncertainty quantification techniques have been developed to characterize, propagate and quantify epistemic uncertainties. These techniques have been successfully applied in many scientific applications [2, 3, 4, 5, 6, 7].

The traditional statistical approach for uncertainty quantification is the Monte Carlo method [8, 9]. With the brute force Monte Carlo implementa-

tion, one first generates an ensemble of random realizations with each parameter drawn from its uncertainty distribution. Deterministic solvers are then applied to each member to obtain an ensemble of results. The ensemble of results is post-processed to obtain the relevant statistical properties of the results such as the mean and standard deviation, as well as the probability density function (PDF). Since the estimation of the variance converges with the inverse square root of the number of runs, the Monte Carlo approach is computationally expensive. Although techniques such as Latin hypercube sampling [10], the quasi-Monte Carlo (QMC) method [11], and the Markov Chain Monte Carlo methods (MCMC) [12] can significantly improve the Monte Carlo convergence rate, their applications are limited. The advantages of the sampling-based methods are their conceptual simplicity and the relative ease of implementation, which allow them to be applied to almost any model regardless of its size and complexity. The disadvantages are the requirements for a large number of runs to obtain relatively accurate statistics.

In non-sampling techniques, uncertainties are represented by a spectral approximation that allows high order representations. The polynomial chaos approach is one of the most frequently used non-sampling approaches. The polynomial chaos approach originates from the homogeneous chaos concept defined by Wiener [13]. Ghanem and co-workers [14] have demonstrated that polynomial chaos is a feasible computational tool for scientific and engineering studies. Karniadakis and Xiu [15] generalized and expanded the concept by using orthogonal polynomials from the Askey-scheme class as the expansion basis. They further proposed that if the Wiener-Askey polynomial chaos expansion is chosen according to the probability distribution of the random input, then the chaos expansion can reach the optimal exponential convergence rate. Xiu has applied those expansions to stochastic differential equations extensively [16, 17, 18].

Consider a complete probability space $(\Omega, \mathcal{F}, P)$, in which Ω is a sample space, $\mathcal{F}$ a σ-algebra of subsets of Ω, and P a probability measure. A general second-order random process $X(\theta) \in L_2(\Omega, \mathcal{F}, P)$ can be represented as:

$$X(\theta) = \sum_{i=0}^{\infty} c^i \, \Phi^i(\xi(\theta))$$

where θ is a random event, and $\Phi^i(\xi(\theta))$ are polynomial functionals defined in terms of the multi-dimensional random variable $\xi(\theta) = (\xi_1(\theta), \ldots, \xi_d(\theta))$ with the joint probability density function of $w(\xi)$. The family $\{\Phi^i\}$ satisfies the orthogonality relations:

$$\langle \Phi^i, \Phi^j \rangle = 0 \quad \text{for} \quad i \neq j.$$

where the inner product on the Hilbert space of random functionals is the ensemble average $\langle \cdot, \cdot \rangle$:

$$\langle f, g \rangle = \int f(\xi) \, g(\xi) \, w(\xi) d\xi.$$

If the uncertainties in the model are independent random variables $\xi = (\xi_1, \cdots, \xi_n)$ with a joint probability distribution $w(\xi) = w^{(1)}(\xi_1) \cdots w^{(n)}(\xi_n)$, then a multi-dimensional orthogonal basis is constructed from the tensor products of one-dimensional polynomials $\{P_m^{(k)}\}_{m \geq 0}$ orthogonal with respect to the density $w^{(k)}$ [6]:

$$\Phi^i(\xi_1, \cdots \xi_n) = P_{i_1}^{(1)}(\xi_1) P_{i_2}^{(2)}(\xi_2) \cdots P_{i_n}^{(n)}(\xi_n).$$

In this case, the evaluation of n-dimensional scalar products is reduced to n independent one-dimensional scalar products.

In practice, we consider a truncated polynomial chaos expansion with S terms. We denote the number of random variables by n, and the maximum degree of the polynomials by p. S is given by [14]:

$$S = \frac{(n+p)!}{(n!\, p!)}. \tag{1}$$

With the growth of the polynomial order and the number of the random variables, the total number of terms in the expansion increases rapidly. Using polynomial chaos expansion, the original ODE is replaced by an ODE for the polynomial chaos coefficients, and the uncertainty information is embedded in these coefficients. For example, for the deterministic system

$$y' = f(y), \qquad t^0 \leq t \leq T, \qquad y(t^0) = y^0, \tag{2}$$

we expand the state vector y along the polynomial chaos basis

$$y = \sum_{i=1}^{S} y^i \Phi^i(\xi)$$

and insert it into the deterministic system to obtain

$$\sum_{i=1}^{S} (y^i)' \Phi^i(\xi) = f\left(\sum_{j=1}^{S} y^j \Phi^j(\xi) \right). \tag{3}$$

The evolution equation for the stochastic coefficients can then be obtained by Galerkin or collocation approaches as explained below.

In the Galerkin polynomial chaos approach, we project the system (3) on the space spanned by the orthogonal polynomials, i.e, we take the inner product of (3) with $\Phi^i(\xi)$ to obtain:

$$\left(y^i\right)' = \frac{1}{\langle \Phi^i, \Phi^i \rangle} \left\langle \Phi^i(\xi), f\left(\sum_{j=1}^{S} y^j \Phi^j(\xi) \right) \right\rangle, \qquad 1 \leq i \leq S. \tag{4}$$

This system has $n \times S$ components and their evolution is coupled. With this approach, the entire stochastic system needs to be evolved in time, but the integration is only executed once.

The collocation approach imposes the system (3) to be satisfied at a given set of points μ^j $(1 \leq j \leq Q)$:

$$\sum_{i=1}^{S} (y^i)' \Phi^i(\mu^j) = f\left(\sum_{i=1}^{S} y^i \Phi^i(\mu^j)\right), \qquad 1 \leq j \leq Q \tag{5}$$

With matrix A defined using the basis function values at the collocation points

$$A_{i,j} = \Phi^j(\mu^i), \qquad 1 \leq i \leq Q, \qquad 1 \leq j \leq S. \tag{6}$$

the collocation points in the system state space are:

$$Y^j(t) = \sum_{i=1}^{S} y^i(t) \Phi^i(\mu^j) = \sum_{i=1}^{S} A_{j,i}\, y^i(t), \qquad 1 \leq j \leq Q. \tag{7}$$

With this notation, the equation (5) becomes:

$$(Y^i)' = f(Y^i), \qquad 1 \leq i \leq Q. \tag{8}$$

There are Q independent integrations of the deterministic system (2):

$$Y^i(t^0) = \sum_{i=1}^{S} A_{j,i}\, y^i(t^0), \qquad t^0 \leq t \leq T. \tag{9}$$

After integration, we recover the stochastic solution coefficients at the final time T by solving the linear system of equations for y^i:

$$Y^j(T) = \sum_{i=1}^{Q} A_{j,i}\, y^i(T), \qquad 1 \leq j \leq Q, \tag{10}$$

which equavalents to:

$$y^j(T) = \sum_{i=1}^{Q} \left(A^{\#}\right)_{j,i} Y^i(T), \qquad 1 \leq j \leq S, \tag{11}$$

where $A^{\#}$ is the pseudo-inverse ($A^{\#} = A^{-1}$ if $Q = S$).

In this paper, we address the uncertainty quantification problem using polynomial chaos for stiff systems. We compare the prevalent uncertainty quantification techniques, and propose an improved algorithm to apply the Galerkin polynomial chaos method. We show that the Galerkin polynomial chaos and the collocation method are both superior to the Monte Carlo approach for stiff systems. With the improved techniques of applying Galerkin polynomial chaos, the computation time can be further reduced. In the general collocation approach, even if only Q collocation points are required to solve the exact system with Q unknown coefficients, we propose a least-squares collocation method that uses more points than the unknown coefficients, i.e., $Q > S$.

The rest of the paper is organized as follows: Section 2 describes the difficulties of applying the Galerkin polynomial chaos to implicit stiff systems. We implement different orderings of the system Jacobians, and propose a modification of the Jacobian to improve the computational efficiency. Section 3 discusses the collocation and Lagrange interpolation methods. We propose a least-squares stochastic collocation approach, and compare the numerical results against those obtained through the high order Lagrange interpolation using Smolyak algorithm. The conclusions are made in Section 4.

2 Application of the Galerkin Polynomial Chaos to Stiff Systems

We illustrate our approach using a stiff system arising from chemical kinetics. The simple Chapman-like mechanism from stratospheric chemistry is specified by the following equations:

$$
\begin{array}{lccccccccl}
r1) & O_2 & + & h\nu & \xrightarrow{k_1} & 2\,O & & & & \left(k_1 = 2.643 \times 10^{-10} \cdot \sigma 3\right) \\
r2) & O & + & O_2 & \xrightarrow{k_2} & O_3 & & & & \left(k_2 = 8.018 \times 10^{-17}\right) \\
r3) & O_3 & + & h\nu & \xrightarrow{k_3} & O & + & O_2 & & \left(k_3 = 6.120 \times 10^{-04} \cdot \sigma\right) \\
r4) & O & + & O_3 & \xrightarrow{k_4} & 2\,O_2 & & & & \left(k_4 = 1.576 \times 10^{-15}\right) \\
r5) & O_3 & + & h\nu & \xrightarrow{k_5} & O^{1D} & + & O_2 & & \left(k_5 = 1.070 \times 10^{-03} \cdot \sigma 2\right) \\
r6) & O^{1D} & + & M & \xrightarrow{k_6} & O & + & M & & \left(k_6 = 7.110 \times 10^{-11}\right) \\
r7) & O^{1D} & + & O_3 & \xrightarrow{k_7} & 2\,O_2 & & & & \left(k_7 = 1.200 \times 10^{-10}\right) \\
r8) & NO & + & O_3 & \xrightarrow{k_8} & NO_2 & + & O_2 & & \left(k_8 = 6.062 \times 10^{-15}\right) \\
r9) & NO_2 & + & O & \xrightarrow{k_9} & NO & + & O_2 & & \left(k_9 = 1.069 \times 10^{-11}\right) \\
r10) & NO_2 & + & h\nu & \xrightarrow{k_{10}} & NO & + & O & & \left(k_{10} = 1.289 \times 10^{-02} \cdot \sigma\right)
\end{array}
\tag{12}
$$

The model involves five variable chemical species $(O, O_{1D}, O_3, NO, NO_2)$ and two fixed species (M, O_2). The model predicts the concentration of the species at a future time from a given initial concentration. The deterministic model codes are generated with Kinetic Preprocessor (KPP) [19, 20] version

1.2. KPP parses the chemical mechanism, builds the derivative function and Jacobian to describe the chemical transformation, and offers support for treating sparsity. As the chemical system is stiff, meaning the lifetime of some species involved in the reactions are many orders of magnitude shorter than those of other species, a Rosenbrock numerical integrator is used for time integration [21]. For the stochastic model, we assume that the uncertainties only come from the initial condition. The uncertainties are uniformly distributed in the range $\pm 20\%$ of the reference solution. We expand the uncertain initial concentrations using polynomial chaos with the Legendre basis. After the expansion, each species contains S stochastic mode to represent the statistical information. We re-arrange the expanded five species into a single long state vector to perform the integration (two different orderings will be explored further in Section 2.2). After the time integration, we re-assemble the solution to extract the mean and standard deviation of the final result.

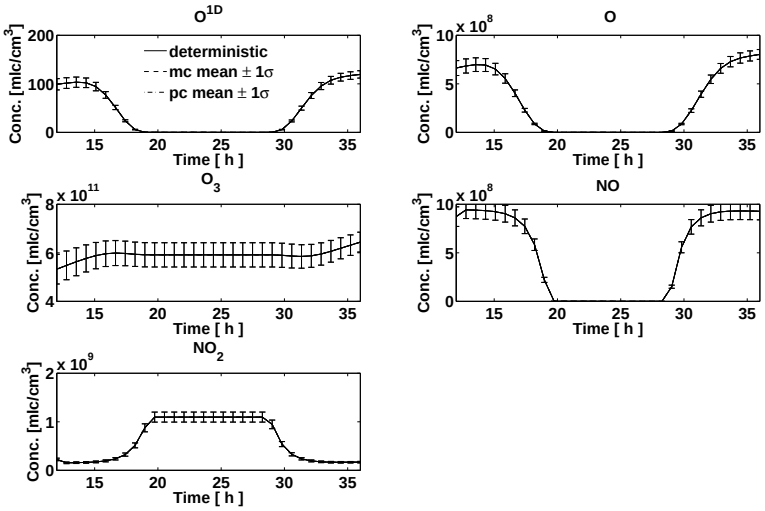


Figure 1: Comparison of the numerical solutions for deterministic, Monte Carlo and polynomial chaos approaches after a 24-hour integration

The results of the stochastic solutions are shown in Figure 1. We make a comparison for the 24-hour integration results among a deterministic run (solid line), a Monte Carlo run with 500 members (dashed line) and a Galerkin polynomial chaos order 3 result (dash dot line). The "error bar" result using the Galerkin polynomial chaos approach is comparable with the Monte Carlo result.

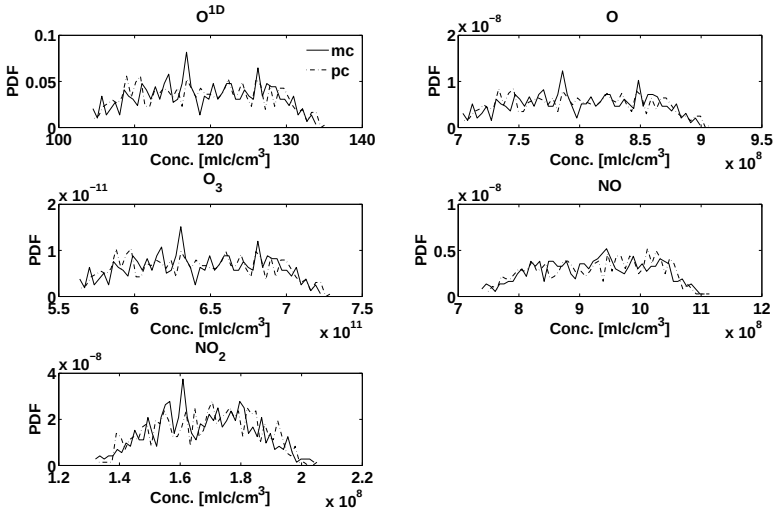


Figure 2: PDF comparison of Monte Carlo and Galerkin PC approaches

In most cases, the polynomial chaos result gives a smaller standard deviation compared to the Monte Carlo method. These results show that the Galerkin polynomial chaos approach works well for stiff systems.

Figure 2 shows a comparison of the probability density function of the Monte Carlo and Galerkin polynomial chaos results at the final integration time. We see that the PDF shapes of the two results match well, which means that the Galerkin polynomial chaos method can be used in place of the Monte Carlo method to capture the propagation behavior of the uncertainties in the initial conditions.

2.1 Difficulties of Applying the Galerkin Approach to Stiff Systems

Using the Galerkin approach, most of the computation time is consumed in the construction of the right hand side ODE function and its Jacobian matrix. The increasing computational cost is illustrated using the implicit Euler scheme applied to (2):

$$y^{n+1} = y^n + \Delta t f(y^{n+1}).$$

The implicit system is solved using the modified Newton formula which uses the Jacobian matrix evaluated at time t^n. We solve the equation

$$z - y^n - \Delta t f(z) = 0$$

for z using iterations of the form:

$$z^{[m]} = z^{[m-1]} - \left(I - \Delta t J(y^n)\right)^{-1} \left(z^{[m-1]} - y^n - \Delta t f(z)\right).$$
$$z^{[0]} = y^n, \qquad m = 1, 2, \cdots$$

To solve this system of equations with n unknowns, the complexity is $O(n^3)$ if full linear algebra is used. Using Galerkin polynomial chaos approach, the size of the stochastic system is $S \times n$, and the complexity of the linear algebra will be $O(S^3 n^3)$–an increase by a factor of S^3. In our experiment, the original state variable has length $n = 5$. With order 3 polynomial chaos expansion, the number of the stochastic modes is $S = 56$, as computed by (1). The deterministic Jacobian is a 5×5 matrix as shown in Figure 3 (a). However, in the stochastic system, by introducing the additional stochastic dimension, the system state is expanded to $n \times S = 5 \times 56 = 280$. The Jacobian size increases to 280×280, and the computational complexity associated with the full linear algebra solver increases by $56^3 \approx 176\,000$ times! The solution process can become infeasible.

In order to overcome this difficulty, we explore different orderings of the polynomial chaos coefficients, and propose an approximation to the Jacobian matrix.

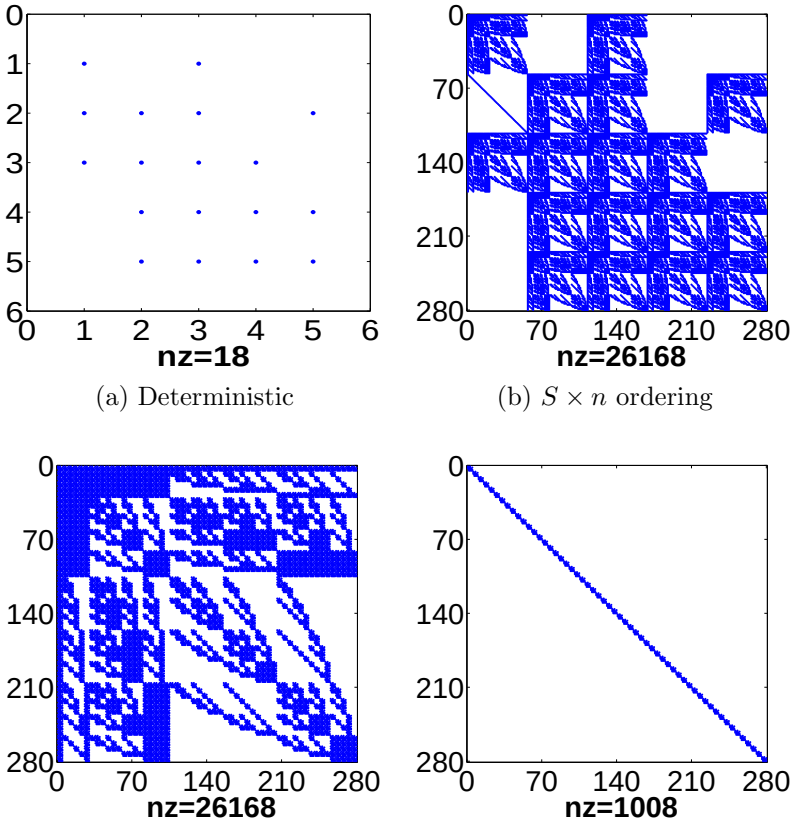
2.2 Orderings of the Polynomial Chaos Coefficients

There are two different ways to order the polynomial chaos coefficients into a one-dimensional vector. One method is to use the polynomial chaos index first, the other uses the system index first.

We refer to the first method as the $S \times n$ ordering of the Galerkin Jacobian. After the arrangement, the index (i, j) in the $S \times n$ space with $1 \le i \le S$ and $1 \le j \le n$ corresponds to $k = (j-1)n + i$ in the 1-dimentional vector. The corresponding Jacobian matrix of the $S \times n$ ordering has the same sparsity structure as the deterministic Jacobian matrix. Figure 3 (b) shows the Jacobian sparsity structure of the polynomial chaos order 3 model. Comparing with (a), we can see that each entry in the deterministic Jacobian matrix corresponds to a $S \times S$ block in the Galerkin Jacobian matrix.

The second ordering is referred to as the $n \times S$ ordering (the system index first, followed by the polynomial chaos index). The index (i, j) in the $n \times S$ space with $1 \le i \le n$ and $1 \le j \le S$ corresponds to index $k = (j-1)S + i$ in 1-dimensional ordering. The sparsity structure of the Jacobian matrix is shown in Figure 3 (c). The Jacobian is now a $S \times S$ block matrix, with each $n \times n$ block having the structure of the deterministic Jacobian.

Compared with using the Monte Carlo method, which requires solving a 5×5 system repeatedly, using Galerkin polynomial chaos, we solve one big linear system of dimension 280×280 once. However, it takes much longer to solve such a large system than solving the small system repeatedly.



(a) Deterministic (b) $S \times n$ ordering

(c) $n \times S$ ordering (d) Approximation

Figure 3: The Jacobian sparsity structure for (a) deterministic, and for (b) $S \times n$ ordering, (c) $n \times S$ ordering and (d) the approximated stochastic Jacobian.

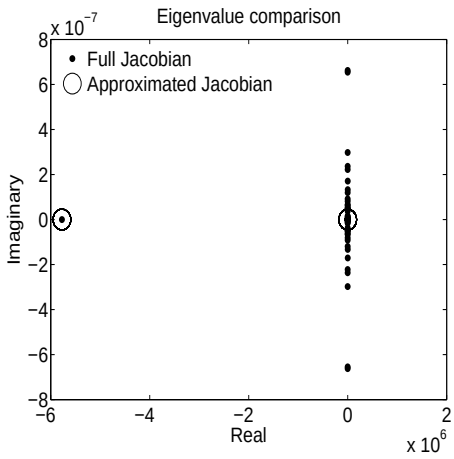


Figure 4: Eigenvalue comparison of the full Jacobian and the approximated Jacobian.

2.3 Approximation of the Jacobian Matrix

Using the exact Jacobian matrix requires its evaluation at every time step, which is computationally costly. An eigenvalue analysis of the Jacobian matrix provides additional information. We find that in our stochastic system (with n states, S stochastic modes), the eigenvalues of the full Galerkin Jacobian matrix are similar to the eigenvalues of the deterministic Jacobian matrix repeated S times, as shown in Figure 4.

We consider the $n \times S$ ordering. With the eigenvalue information, we approximate the stochastic Jacobian by a block diagonal matrix with the blocks on the diagonal being the $n \times n$ deterministic Jacobian repeated S times. Starting from the polynomial chaos stochastic formulation (4), we obtain the Galerkin Jacobian for each $n \times n$ block (i, j) with $1 \leq i, j \leq S$ as follows:

Let

$$Y = \sum_{k=1}^{S} y^k \Phi^k(\xi).$$

Then,

$$\begin{aligned} \frac{\partial (y^i)'}{\partial y^j} &= \frac{\left\langle \Phi^i, \frac{\partial f(Y)}{\partial y^j} \right\rangle}{\langle \Phi^i, \Phi^i \rangle} \\ &= \frac{\left\langle \Phi^i, \frac{\partial f(Y)}{\partial Y} \cdot \frac{\partial Y}{\partial y^j} \right\rangle}{\langle \Phi^i, \Phi^i \rangle} \end{aligned}$$

$$
\begin{aligned}
&= \frac{\left\langle \Phi^i, \frac{\partial f(Y)}{\partial Y} \cdot \Phi^j \right\rangle}{\langle \Phi^i, \Phi^i \rangle} \\
&= \frac{\left\langle \Phi^i, J(Y) \cdot \Phi^j \right\rangle}{\langle \Phi^i, \Phi^i \rangle} \\
&= \frac{1}{\langle \Phi^i, \Phi^i \rangle} \cdot \left\langle \Phi^i, J\Big(\sum_{k=1}^{S} y^k \Phi^k(\xi)\Big) \cdot \Phi^j \right\rangle .
\end{aligned}
$$

Justified by the above eigenvalue analysis, we use the approximation:

$$
J\left(\sum_{k=1}^{S} y^k \Phi^k(\xi)\right) \approx J(\bar{y}), \quad \text{with} \quad \bar{y} = \left\langle \sum_{k=1}^{S} y^k \Phi^k(\xi) \right\rangle .
$$

Then we have that the approximated Jacobian has a diagonal block structure:

$$
\frac{\partial (y^i)'}{\partial y^j} \approx \begin{cases} J(\bar{y}) & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases} .
$$

The approximated Jacobian can be used in implicit time stepping in two ways. First, the modified Newton method (13) can use the simplified Jacobian for fully implicit numerical methods. Second, we consider Rosenbrock-Wanner methods [22] that maintain the order of accuracy when inexact Jacobians are employed. The particular scheme we are using is the Ros-2 integrator [23, 24] formulated as follows:

$$
\begin{aligned}
y^{n+1} &= y^n + \frac{3}{2}k_1 + \frac{1}{2}k_2 \\
(I - \gamma \Delta t J) k_1 &= \Delta t f(y^n) \\
(I - \gamma \Delta t J) k_2 &= \Delta t f(y^n + k_1) - 2k_2
\end{aligned}
$$

in which the matrix J is the approximated Jacobian. The method is second-order consistent for $\gamma = 1 + 1/\sqrt{2}$.

The sparsity structure of the approximated Jacobian is shown in Figure 3 (d). As a result, instead of solving the large $Sn \times Sn$ system required by full Galerkin Jacobian at each time iteration, we repeatedly solve S smaller systems with the fixed deterministic Jacobian and corresponding stochastic mode on the right-hand side of the stochastic system. Since we have the same $n \times n$ matrix for all S systems, we can use the same LU decomposition for each stochastic mode.

2.4 Numerical Results

We now compare the accuracy of the solutions with the full and the approximated Jacobians. We use the numerical results obtained by Monte Carlo with 10 000 runs as our reference solution. We implement the $S \times n$ ordering, the $n \times S$ ordering, and the approximated diagonal Jacobian with the Ros-2 integrator and $Rtol = 1.0d - 2$, $Atol = 1.0d + 4$ (molec/cm^3) as tolerances.

Table 1 lists the error comparison for $S \times n$ ordering, $n \times S$ ordering and the approximated diagonal Jacobian against the Monte Carlo 10 000 reference runs. The single values in the last row are computed by taking the maximum entry of the relative errors in the covariance matrix. The relative errors are computed by dividing the norm of the error by the norm of the reference solution:

$$\text{relative error} = \frac{||\text{PC solution} - \text{reference solution}||}{||\text{reference solution}||}.$$

	$S \times n$ order	$n \times S$ order	Approximation
Mean	2.400e-04	2.400e-04	8.307e-04
Std	1.827e-03	1.827e-03	2.472e-03
Covariance	4.628e-04	4.628e-04	1.633e-03

Table 1: Error in PC order 3 solution compared with MC reference solution of 10 000 runs. The PC solutions are obtained with different orderings of the full Galerkin Jacobian and with the block diagonal approximation.

We can see from Table 1 that both $S \times n$ and $n \times S$ orderings have the same errors, while the approximated Jacobian has a slightly larger error compared with the reference solution.

Next we compare the run times for the different orderings. Table 2 shows the run time (in seconds) for polynomial chaos order 2 and order 3 compared with Monte Carlo method.

	$S \times n$ order	$n \times S$ order	Approximation
PC-order 3	15.80	17.30	10.50
PC-order 2	1.03	1.22	0.89
MC-10000	989.7		

Table 2: Run time comparison of PC order 2, PC order 3 and MC (time is given in seconds for FORTRAN implementation)

From the results in Table 1 and 2, we conclude that with a reduced run time, the approximation using the diagonal Jacobian generates a satisfactory numerical solution for our chemical system. A run time profiling check provides us additional information as where the most of the computing time is spent. The total CPU time is used in the function evaluation, Jacobian computation and

the linear algebra. The computation time we reduced is only in the Jacobian and the linear algebra part. The function evaluation on the right-hand side function of the equation (4) remains the main contributor to the total CPU time.

3 Collocation Methods

From the above discussion, we conclude that Monte Carlo implementation is straightforward, but requires many sample runs to generate the response in order to form a histogram of the final PDF. While polynomial chaos takes less run time, the derivation of the stochastic model is both time consuming and complicated for large nonlinear models. The increase of the number of uncertainties will greatly affect the size of the stochastic system.

The general collocation method uses polynomial chaos expansion (3). Instead of taking the Galerkin projection to derive the equations for the stochastic coefficients, the equations are imposed to hold at a given set of collocation points. The deterministic system needs to be integrated several times instead of running the larger system once. An algebraic equation system (11) is formed and solved to compute the coefficients. This approach is similar to the Stochastic Response Surface Method (SRSM) and the Deterministic Equivalent Modeling Method (DEMM) [25, 26].

The stochastic collocation method [27, 28, 29] is a hybrid of Monte Carlo method and polynomial chaos approach. It uses the traditional collocation method with random variables to represent the uncertainties. We present the stochastic collocation method with Lagrange interpolation in Section 3.2.

An important issue when applying the collocation approach is the selection of the collocation points. A necessary criterion for selecting these points is that the system coefficients matrix $A_{j,i}$ in equation (10) is well conditioned. We first test the random data sets by restricting the magnitude of the condition number when selecting the random collocation points. Although it is convenient to use the random numbers as the collocation points, an accurate solution generated by the randomly chosen collocation points can not be guaranteed for each run. Moreover, the alignment or clustering of the data points can easily cause the rank deficiency for the system matrix. Therefore, we consider other sets of collocation points as discussed next.

3.1 Low-Discrepancy Data Sets As Collocation Points

The low discrepancy sequences [30] have been explored in the area of the quasi-Monte Carlo integration with large number of variables. The commonly used data sets include Hammersley points [31], Halton points [32] and Sobol points [33]. The Hammersley points and Halton points are closely related. A comparison of these data sets has been made in [34] for numerical integration purposes. We use the Hammersley data set due to its uniformity.

The Hammersley-Sequence-Sampling (HSS) provides a way of generating samples of any distribution based on an evenly distributed set of points. It has been used in many applications [35]. The Hammersley points are generated as follows. In radix-R notation, any integer i can be represented as:

$$i = \sum_{j=0}^{v} i_j R^j$$

where R is an integer that is greater than 1, $v = floor\{log_R i\}$, and i_j is an integer between 0 and $R-1$ for each $j = 0, 1, \cdots, v$. The inverse radix number of i is given by:

$$\phi_R(i) = \sum_{j=0}^{v} \frac{i_j}{R^{j+1}}.$$

The S Hammersley sequence points m_i in a k-dimensional unit hypercube can be computed using:

$$m_i = 1 - [i/S, \phi_{r_1}(i), \phi_{R_2}(i), \cdots, \phi_{R_{k-1}}(i)]^T$$

where $i = 1, 2, \cdots, S$ and R_j is the jth prime number.

After the generation of the S Hammersley points, the Hammersley sampling sequence p_i can be generated by transforming the Hammersley points through the inverse Cumulative Density Function (CDF) of the uncertain parameter

$$p_i = F_p^{-1}(m_i).$$

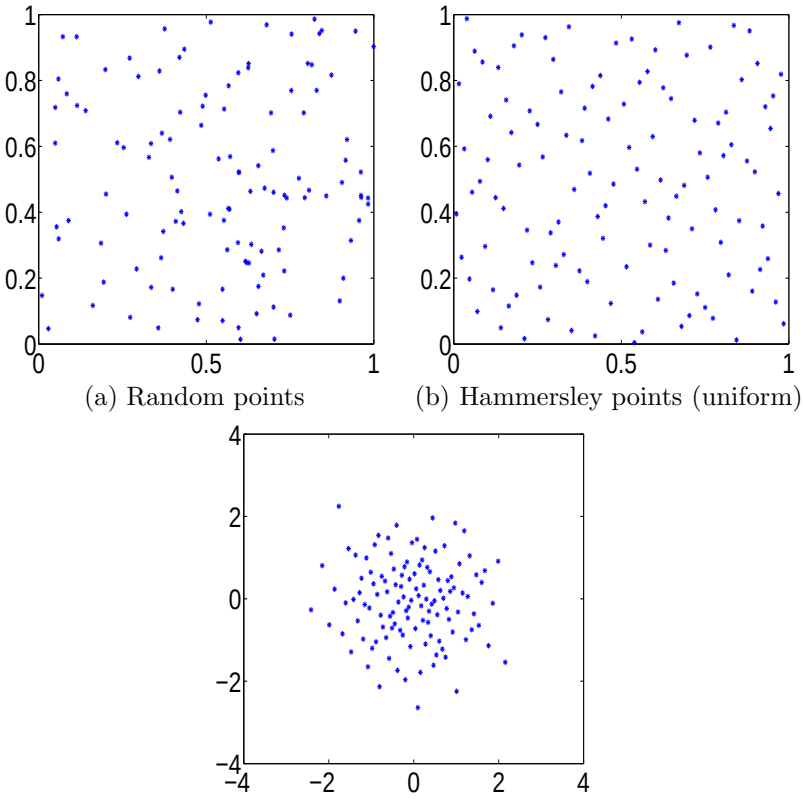
In general, for the same sample size, HSS sample is more representative of the random distribution than the random generated sample. In the case of a different distribution, such as the Gaussian distribution, the points are more clustered to represent the PDF of the associated random variable. Figure 5 (a) and 5 (b) show the comparison of the random points and the uniformly distributed Hammersley points. The Hammersley points for Gaussian distribution is shown in Figure 5 (c).

Another possible data set is the Smolyak sparse grids [36, 37] generated with the Smolyak algorithm. However, there are some restrictions of using the Smolyak data set as collocation points. We discuss the details in Section 3.2.

We present the numerical results comparison for general collocation method using different data sets in Section 3.3.

3.2 Smolyak Algorithm and Lagrange Interpolation

The stochastic collocation method is based on the polynomial interpolation in the multi-dimensional random space. Since we are not interpolating the



(a) Random points (b) Hammersley points (uniform)

(c) Hammersley points (Gaussian distribution)

Figure 5: Comparison of random points, uniform Hammersley points and Gaussian Hammersley points

derivative, but only the function itself, Lagrange interpolation is usually chosen [16]:

$$I(f)(x(\xi)) = \sum_{i=1}^{Q} f(x_i) L_i(x(\xi)).$$

By construction, the Lagrange polynomial associated with the i-th node x_i is defined in terms of all the nodes x_j with $j = 1, 2, \ldots, N+1$. The value of the Lagrange polynomial L_i associated with x_i equals to zero at all data points x_j except at x_i, where it equals 1.

The Chebyshev points are the optimal choice for 1-dimensional interpolation:

$$Y_j^i = -cos\frac{\pi \cdot (j-1)}{m_i - 1}, \qquad j = 1, \ldots, m_i$$

with $m_1 = 1$ and $m_i = 2^{i-1}$ for $i > 1$. Here i is the grid index (with grid i having m_i points) and j is the node index within the grid. The Chebyshev grids are naturally nested.

We construct the 1-dimensional interpolation as:

$$\mathcal{U}^i(f) = \sum_{k=1}^{m_i} f(Y_k^i) \cdot L_k^i,$$

where L_k^i are the 1-dimensional Lagrange polynomials associated with mode k on grid level i. The multi-dimensional interpolant based on the full tensor product is constructed as [16]:

$$\begin{aligned} \mathcal{I}(f) &\equiv (\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_N})(f) \\ &= \sum_{k_1=1}^{m_{i_1}} \cdots \sum_{k_N=1}^{m_{i_N}} f(Y_{k_1}^{i_1}, \cdots, Y_{k_N}^{i_N}) \cdot (L_{k_1}^{i_1} \otimes \cdots \otimes L_{k_N}^{i_N}). \end{aligned}$$

In higher dimensions, however, the full tensor product becomes infeasible, since the number of interpolation points grows exponentially fast with the increase in the number of dimensions. This is the so-called "curse of dimensionality".

The Smolyak algorithm [38] is designed to handle the "curse of dimensionality". Instead of using a full tensor product for multi-dimensional interpolation, the algorithm chooses the representative polynomials from the lower dimensions to form the higher-order interpolation polynomial.

There are two implementations of the Smolyak algorithm. One uses the Chebyshev points in 1-dimension, which generates the Clenshaw-Curtis formula. The other option uses Gaussian points (zeros of the orthogonal polynomials with respect to a weight ρ), which leads to the Gaussian formula. The optimality of these constructions over the traditional tensor product is discussed in [39].

Using the Smolyak algorithm, a polynomial interpolation of a d-dimensional function is given by a linear combination of the tensor product polynomials:

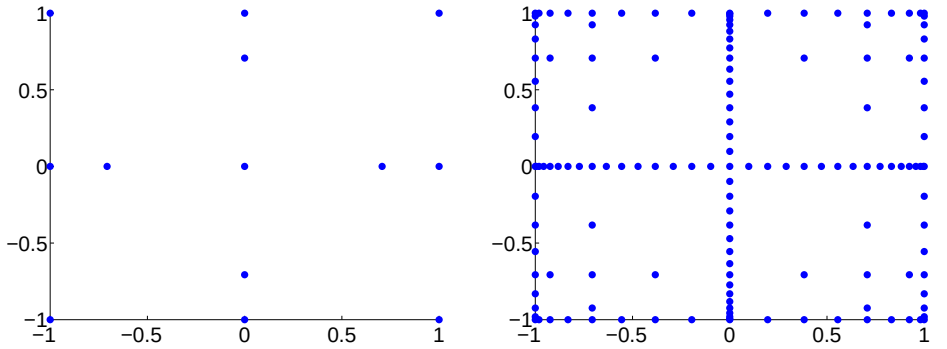
$$\mathcal{I}(f) \equiv A(q,d) = \sum_{q-d+1 \le |i| \le q} (-1)^{q-|i|_1} \binom{d-1}{q-|i|} \cdot (\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_d})$$

in which d is the number of dimensions, q is the order of the algorithm, and $q - d$ is called the level of interpolation.

The nested data set is used to guarantee that the higher degree polynomials reproduce all polynomials of the lower degrees.

Figure 6 illustrates the 2-dimensional Clenshaw-Curtise sparse grids for level 2 and level 5 constructions.

The Smolyak sparse grids can be used in both the general collocation method and the stochastic high-order interpolation approaches. However, there are two disadvantages of using the Smolyak sparse grids as the collocation points in the



(a) level 2 (13 points) (b) level 5 (145 points)

Figure 6: 2-D Smolyak points

general collocation approach. First, the number of the points is restricted by the algorithm, as can be seen in Table 3. For example, for order 3 PC expansion, we need to solve for 56 unknown coefficients, but with the 5 dimensional, level 1 construction, we obtain 11 Smolyak points, with the level 2 construction, 61 points are obtained. We end up with solving a least-squares problem. Second, the alignment of the data points can easily cause the system matrix to become rank deficient. In this case, a higher level of Smolyak construction is required. For example, for PC order 3 expansion, in order to avoid the rank deficiency problem, we have to use Smolyak level 3 construction, i.e. using 241 points to solve for 56 unknown coefficients.

Dimension	level	No. of Smolyak points
2	1	5
	2	13
	3	29
5	1	11
	2	61
	3	241

Table 3: Number of Smolyak points for different construction levels

3.3 Collocation Least-squares Method

As illustrated in (13), in order to solve for S unknowns, we need to generate S equations with Q chosen collocation points. If the number of collocation points is greater than the number of unknowns, i.e, $Q > S$, we obtain a least-squares system.

We assess the accuracy of the collocation least-squares method on a 2-dimensional problem and compare the results with the Lagrange interpolation

results using the Smolyak grids.

For collocation and collocation least-squares implementation, we use the Clenshaw-Curtis formulation. We implement the Lagrange interpolation with the Clenshaw-Curtis points in our stiff system with two uncertainties. We assume two independent uncertainties ξ_1 and ξ_2 in the initial concentrations as follows:

$$\begin{aligned} O_3 &= O_3 + 0.5 \times \xi_1 \times O_3 \\ NO &= NO + 0.25 \times \xi_2 \times (NO + NO_2) \\ NO_2 &= NO_2 + 0.25 \times \xi_2 \times (NO + NO_2) \end{aligned}$$

We compare the results of the Lagrange interpolation, the collocation approach, and the collocation least-squares approach for different numbers of collocation points. Figure 7 shows the error mesh plot for species O_3 in each implementation. All errors are computed by dividing the absolute point-wise error by the maximum value in the Galerkin polynomial chaos reference solution. For example, the O_3 collocation error is computed by the following formula:

$$O_3\, error(\xi_1, \xi_2) = \frac{|O_3^{colloc}(\xi_1, \xi_2) - O_3^{Galerkin}(\xi_1, \xi_2)|}{\max\limits_{\xi_1, \xi_2} |O_3^{Galerkin}(\xi_1, \xi_2)|}$$

We experimented with 10, 15, 20 and 40 points. Collocation least-squares with both Halton and Hammersley points generates smaller errors than the regular collocation approach. The Hammersley solution is more accurate than Halton solution.

From the results of Figure 7, we see that both Lagrange interpolation and collocation least-squares methods generate similar results with the Galerkin polynomial chaos approach. The run time of both approaches are much faster than the Galerkin polynomial chaos. We further compare the collocation least-squares approach for different numbers of collocation points. We find that for our 2-dimensional test problem, with the increase of the number of the collocation points, the error can be further reduced. But at the same time, the computation time is increased due to the increase of the number of the deterministic runs. We appreciate that a practical choice for the number of collocation points should be about 1.5 times the unknown coefficients considering the accuracy of the solution and the computation time.

For the stiff system with 5 uncertainties, we compare the numerical solution of the general collocation approach using different data sets and the Galerkin polynomial chaos solution against the Monte Carlo implementation. The numerical results are listed in Table 4. These results are obtained by a 24 hour integration using polynomial chaos of order 3 with a Legendre basis. In the table, the collocation with 56 points is the regular collocation that solves the exact system. The one with 85 and 120 points are the collocation least-squares

solution, in which we imposed the equation to hold at more points than the unknown coefficients. From the numerical test results, we conclude that although using more collocation points desensitizes the error with respect to the particular choice of the points in the computation, the error in solution does not decrease in general.

	Galerkin	HSS (56)	HSS (85)	HSS (120)	Smolyak (241)
Mean	2.60e-04	8.67e-05	2.94-04	3.02e-04	2.88e-04
Std	1.11e-03	9.80e-03	3.30e-04	1.40e-03	8.00e-04
Time	532.45	22.82	34.48	49.08	109.27

Table 4: Error and run time (in seconds) comparison of collocation and collocation least-squares with different collocation sets against MC reference run.

Between collocation least-squares and the Lagrange interpolation approach, we favor collocation least-squares method although both of them generate similar errors. The collocation least-squares method is easy to implement. The key is choosing good collocation points, like the ones we are considering (Hammersley points). The Lagrange interpolation approach requires the construction of both sparse grids and the high order Lagrange interpolation polynomials. Although by using Smolyak algorithm to replace the full tensor product, the computational cost is greatly reduced, the implementation is not a trivial work. We found that it is better to pre-compute the Lagrange sparse grids as well as the high order, multi-dimensional interpolation polynomials corresponding to the sparse grids, save them to a file and use them later on.

4 Conclusions

Stiff numerical integration requires the solution of a nonlinear system of equations at each step. The cost of full linear algebra grows from $O(n^3)$ in the deterministic case to $O(S^3n^3)$ in the Galerkin polynomial chaos case. The focus of this paper is to find alternative simulation methods to reduce this cost for stiff systems with uncertainties.

We implement and compare several uncertainty quantification techniques that include the traditional Monte Carlo method, the Galerkin polynomial chaos method, the collocation and collocation least-squares approaches, and finally the stochastic high-order interpolation approach.

We propose a block-diagonal approximation to the Jacobian matrix in the Galerkin polynomial chaos formulation, and use a special numerical integrator Ros-2 to accommodate the inexact Jacobian. The numerical results demonstrate that by using the modified approach, the Galerkin polynomial chaos run time can be reduced without compromising the accuracy.

We propose a least-squares collocation approach with the Hammersley data set as collocation points. This method generates satisfactory numerical results at lower computational cost than either the Galerkin polynomial chaos or the

Monte Carlo method. We implement a 2-dimensional test case using the stochastic collocation method with Lagrange interpolation on Smolyak sparse grids. We conclude that the stochastic collocation/interpolation approach has a similiar accuracy as the Galerkin polynomial chaos, but is considerably more efficient. Compared with the collocation/interpolation approach, the collocation least-squares approach is more flexible and easier to implement.

In the future, we will apply the collocation method with Hammersley collocation points in the 3D STEM atmospheric chemistry and transportation model to account for the combined effects of uncertainties in the initial conditions, boundary conditions, as well as in the emissions. These uncertainties will be propagated through the stiff chamical reactions using the techniques developed in this paper. This improved model that incorporates the uncertainties will provide more realistic predictions for policy decision making.

5 Acknowledgments

This work is supported by NSF CAREER ACI-0413872 and NSF ITR AP & IM 0205198.

References

[1] Willam L. Oberkamph, Jon C. Helton, and Kari Sentz. Mathematical representation of uncertainty. *AIAA Non-Deterministic Approaches Forum*, (2001-1645), April 2001.

[2] B. Debusschere, H. N. Najm, A. Matta, O. M. Knio, R. G. Ghanem, and O. P. Le Maitre. Protein labeling reactions in electrochemical microchannel flow: Numerical prediction and uncertainty propagation. *Physics of fluids*, 15(8):2238–2250, 2003.

[3] O. P. Le Maitre, O. M. Knio, H. N. Najm, and R. G. Ghanem. Uncertainty propagation using Wiener-Haar expansion. *Journal of Computational Physics*, 197(1):28–57, 2004.

[4] A. Matta. *Numerical simulation and uncertainty quantification in microfluidic system*. PhD thesis, Department of Civil Engineering, Johns Hopkins University, 2003.

[5] H. N. Najm, M. T. Reagan, O. M. Knio, R. G. Ghanem, and O. P. Le Maitre. Uncertainty quantification on reacting flow modeling. Technical Report 2003-8598, SANDIA, 2003.

[6] Adrian. Sandu, Corina. Sandu, and M. Ahmadian. Modeling multibody dynamics with uncertainties. Part I: Theoretical and computational aspects. *Multibody System Dynamics*, 2005.

[7] Corina Sandu, Adrian Sandu, and M. Ahmadian. Modeling multibody dynamic systems with uncertainties. Part II: Numerical applications. *Multibody System Dynamics*, 15(3):245–266, 2006.

[8] G. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, New York, 1996.

[9] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, 2001.

[10] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):237–240, 1979.

[11] H. G. Niederreiter. Quasi-monte carlo methods and pseudo-random numbers. *Bulletin of the American Mathematical Society*, 84(6):957–1041, 1978.

[12] R. L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32:1296–1308, 1984.

[13] N. Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60:897–936, 1938.

[14] R. G. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Dover publications, 1991.

[15] D. Xiu and G. Karniadakis. The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24:619–644, 2002.

[16] D. Xiu. Efficient collocational approach for parametric uncertainty analysis. *Communications in Computational Physics*, 2(2):293–309, April 2007.

[17] D. Xiu, R. G. Ghanem, and I. G. Kevrekidis. An equation-free, multiscale computational approach to uncertainty quantification for dynamical systems. *IEEE Computing in Science and Engineering Journal (CiSE)*, 7(3):16–23, 2005.

[18] D. Xiu and G. E. Karniadakis. Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos. *Computer Methods in Applied Mechanics and Engineering*, 191(43):4927–4948, 2002.

[19] V. Damian, A. Sandu, M. Damian, F. Potra, and G.R. Carmichael. The Kinetic PreProcessor KPP - a software environment for solving chemical kinetics. *Computers and Chemical Engineering*, 26:1567–1579, 2002.

[20] Adrian. Sandu and R. Sander. Technical notes: Simulating chemical system in Fortran90 and Matlab with the Kinetic PreProcessor KPP-2.1. *Atmospheric Chemistry & Physics*, 6:187–195, 2006.

[21] A. Sandu, J. G. Verwer, J. G. Blom, E. J. Spee, G. R. Carmichael, and F. A. Potra. Stiff ODE solvers for atmospheric chemistry problems II: Rosenbrock solvers. *Atmospheric Environment*, 31:3459–3472, 1997.

[22] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer-Verlag, Berlin, 1991.

[23] M. A. Botchev and J. G. Verwer. A new approximate matrix factorization for implicit time integration in air pollution modeling. *Journal of Computational and Applied Mathematics*, 157(2):309, 2003.

[24] J. G. Verwer, E. J. Spee, J. G. Blom, and W. Hundsdorfer. A second-order rosenbrock method applied to photochemical dispersion problems. *SIAM Journal on Scientific Computing*, 20(4):1456 – 1480, July 1999.

[25] S. S. Isukapalli, A Roy, and P. G. Georgopoulos. Efficient sensitivity/uncertainty analysis using the combined stochastic response surface method and automated differentiation: Application to environmental and biological systems. *Risk Analysis*, 20(5):591–602, October 2000.

[26] M. A. Tatang. *Direct Incorporation of Uncertainty in Chemical and Environmental Engineering Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

[27] I. Babuska, F. Nobile, and R. Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. Technical report, MOX, Dipartimento di Matematica, 2005.

[28] L. Mathelin and M. Hussaini. A stochastic collocation algorithm for uncertainty analysis. Technical Report NASA/CR-2003-212153, NASA Langley Research Center, 2003.

[29] D. Xiu and J. Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM Journal on Scientific Computing*, 27(3):1118–1139, 2005.

[30] Ladislav Kocis and William J. Whiten. Computational investigations of low-discrepancy sequences. *ACM Transactions on Mathematical Software (TOMS)*, 23(2):266–294, June 1997.

[31] J. M. Hammersley. Monte Carlo methods for solving multivariables problems. *ANNUAL NEW YORK ACADEMY OF SCIENCE*, 86:844–874, 1960.

[32] J. H. Halton and G. B. Smith. Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, December 1964.

[33] I. M. Sobol. The distribution of points in a cube and the approximate evaluation of integrals. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[34] William. J. Morokoff and Russel. E. Caflisch. Quasi-random sequences and their discrepancies. *SIAM Journal on Scientific Computing*, 15(6):1251–1279, November 1994.

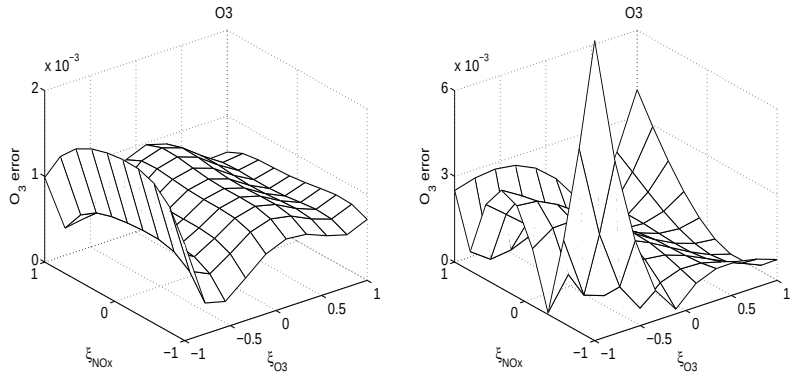
[35] L. G. Crespo and S. P. Kenny. Robust control design for systems with probabilistic uncertainty. Technical report, NASA, Langley Research Center, 2005.

[36] A. Klimke. *Uncertainty Modeling using Fuzzy Arithmetic and Sparse Grids*. PhD thesis, Universitt Stuttgart, Shaker Verlag, Aachen, 2006.

[37] F. Nobile, R. Tempone, and C. G. Webster. A sparse grid stochastic collocation method for elliptic partial differential equations with random input data. preprint.

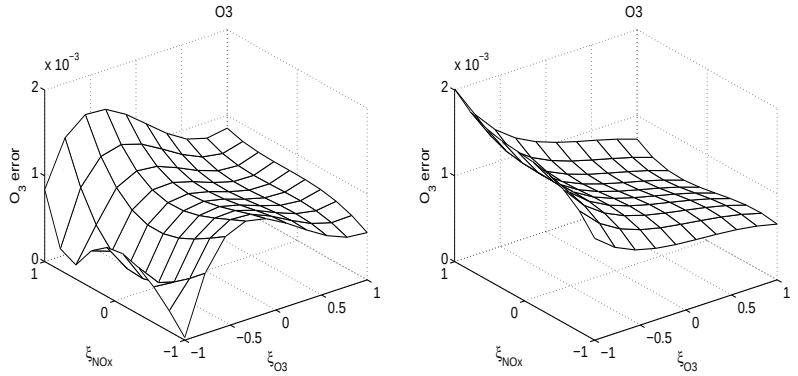
[38] S. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Math. Dokl.*, 4:240–243, 1963.

[39] V. Barthelmann, E. Novak, and K. Ritter. High dimensional polynomial interpolation on sparse grids. *Advanced Computational Mathematics*, 12(4):273–288, November 1999.



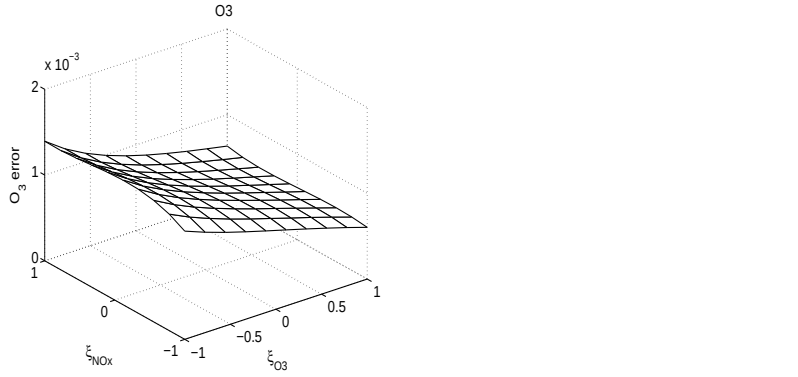
(a) Lagrange interpolation

(b) Collocation (10 pts)



(c) Collocation LS (15 pts)

(d) Collocation LS (20 pts)



(e) Collocation LS (40 pts)

Figure 7: O_3 errors for the 2-D problem. Results are shown for the Lagrange interpolation, collocation and collocation least-squares (using Hammersley points) with the Galerkin solution as reference.