Calculation of Generalized Polynomial-Chaos Basis Functions and Gauss Quadrature Rules in Hierarchical Uncertainty Quantification

Zheng Zhang, *Student Member, IEEE*, Tarek A. El-Moselhy, Ibrahim (Abe) M. Elfadel, *Senior Member, IEEE*, and Luca Daniel, *Member, IEEE*

***Abstract*—Stochastic spectral methods are efficient techniques for uncertainty quantification. Recently they have shown excellent performance in the statistical analysis of integrated circuits. In stochastic spectral methods, one needs to determine a set of orthonormal polynomials and a proper numerical quadrature rule. The former are used as the basis functions in a generalized polynomial chaos expansion. The latter is used to compute the integrals involved in stochastic spectral methods. Obtaining such information requires knowing the density function of the random input *a-priori*. However, individual system components are often described by surrogate models rather than density functions. In order to apply stochastic spectral methods in hierarchical uncertainty quantification, we first propose to construct physically consistent closed-form density functions by two monotone interpolation schemes. Then, by exploiting the special forms of the obtained density functions, we determine the generalized polynomial-chaos basis functions and the Gauss quadrature rules that are required by a stochastic spectral simulator. The effectiveness of our proposed algorithm is verified by both synthetic and practical circuit examples.**

***Index Terms*—Uncertainty quantification, stochastic circuit simulation, density estimation, generalized polynomial chaos, Gauss quadrature, surrogate model.**

I. Introduction

DUE to significant manufacturing process variation, it has become necessary to develop efficient uncertainty quantification tools for the fast statistical analysis of electronic circuits and systems [1]–[18]. Monte Carlo simulators [6]–[8] have been utilized in statistical circuit analysis for decades. Recently, stochastic spectral methods [19]–[24] have emerged as a promising technique for the uncertainty quantification of integrated circuits [1]–[5]. Such methods approximate the stochastic solution by a truncated generalized polynomial chaos expansion [25]–[27], which converges much faster than Monte Carlo when the parameter dimensionality is not high.

This work was funded by the MIT-SkolTech program and by the Cooperative Agreement between the Masdar Institute of Science and Technology, Abu Dhabi, UAE and the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA (Reference No. 196F/002/707/102f/70/9374).

Z. Zhang and L. Daniel are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA (e-mail: z_zhang@mit.edu, luca@mit.edu).

T. El-Moselhy was with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA. He is now with the D. E. Shaw Group, 1166 Avenue of the Americas, New York, NY 10036, USA (e-mail: tarek.moselhy@gmail.com).

I. Elfadel is with the Masdar Institute of Science and Technology, United Arab Emirates (e-mail: ielfadel@masdar.ac.ae).

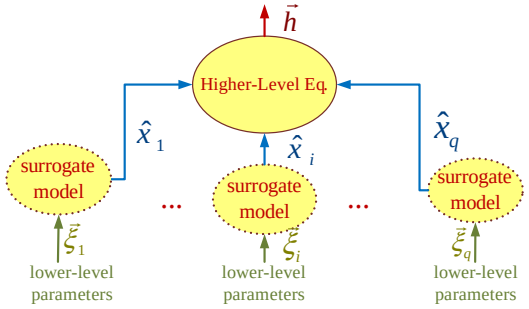


Fig. 1. Demonstration of hierarchical uncertainty quantification.

This work is motivated by the need for hierarchical uncertainty quantification based on stochastic spectral methods. Consider Fig. 1 that demonstrates the uncertainty quantification of a complex system. In this system, there exist q readily obtained surrogate models:

$$\hat{x}_i = f_i(\vec{\xi}_i), \text{ with } \vec{\xi}_i \in \mathbb{R}^{d_i}, \ i = 1, \cdots, q \tag{1}$$

where $\hat{x}_i$ is a variable dependent on multiple lower-level random parameters. In transistor-level simulation, $\hat{x}_i$ is a device-level parameter (e.g., threshold voltage of a transistor) influenced by some geometric and process variations. In a statistical behavior-level simulator [8], $\hat{x}_i$ is the performance metric of a small circuit block (e.g., the frequency of a voltage-controlled oscillator) affected by some device-level parameters $\vec{\xi}_i$. Typical surrogate models include linear (quadratic) response surface models [8], [28]–[32], truncated generalized polynomial chaos representations [1], [2], smooth or non-smooth functions, stochastic reduced-order models [11], [14], [33], and some numerical packages that can rapidly evaluate $f_i(\vec{\xi}_i)$ (e.g., computer codes that implement a compact statistical device model). By solving a system-level equation, the output $\vec{h}$ can be obtained.

Instead of simulating the whole system starting from the bottom-level random parameters $\vec{\xi}_i$'s, uncertainty quantification can be performed in a hierarchical way. By treating $\hat{x}_i$'s as the random inputs for the higher-level equation, the output $\vec{h}$ can be computed more efficiently. This treatment can dramatically reduce the parameter dimensionality and problem size. Related work in this direction includes the statistical analysis of phase-lock loops [8] and the statistical timing analysis of digital VLSI [16]–[18]. In all existing works, Monte Carlo was

utilized to perform the higher-level simulation.

When $\hat{x}_i$'s are mutually independent, it is possible to further speed up the hierarchical uncertainty quantification flow by using stochastic spectral methods [1]–[5], [19]–[24]. In this case, high-accuracy results may be obtained by fast simulation if $\vec{h}$ smoothly depends on $\hat{x}_i$'s (even if $f_i(\vec{\xi}_i)$ is non-smooth). In order to apply available stochastic spectral methods, we need to determine a set of orthonormal polynomials as the basis functions for generalized polynomial chaos expansions. Sometimes we also need a proper numerical quadrature rule (such as Monte Carlo, Gauss quadrature, etc.) [3]. In this paper, we will consider Gauss quadrature since it is widely used in stochastic spectral methods. Both tasks require the probability density function of each random input $\hat{x}_i$. Existing techniques typically assume that the random inputs have some well-known distributions (e.g., Gaussian, uniform, Gamma and Beta distributions), and make use of available quadrature rules and orthogonal polynomials (e.g., Hermite, Legendre, Laguerre and Jacobi polynomials) [1]–[5], [20]–[27]. This assumption obviously does not hold in our case: the probability density function of $\hat{x}_i$ is not readily available from its surrogate model.

Therefore, a question is how to determine the generalized polynomial-chaos basis functions and Gauss quadrature rules from a general surrogate model. This paper aims to partly answer this key question in hierarchical uncertainty quantification. Our method is based on the ideas of changing variables and monotone interpolation [34]–[37]. Specifically, we represent the random input as a linear function of a new parameter, and treat such parameter as a new random input. Using two monotone interpolation schemes, physically consistent closed-form cumulative density functions and probability density functions can be constructed for the new random input. Due to the special forms of the obtained density functions, we can easily determine a proper Gauss quadrature rule and the basis functions for a generalized polynomial chaos expansion.

We focus on the general framework and verify our method by using both synthetic and performance-level circuit surrogate models. Our method can be employed to handle a wide variety of surrogate models, including device-level models for SPICE-level simulators [1]–[3], circuit-level performance models for behavior-level simulation [8], as well as gate-level statistical models for the timing analysis of digital VLSI [16]–[18]. In this paper we will focus only on the derivation of the basis functions and Gauss quadrature rules and refer the reader interested to the extensive literature on how to use them in a stochastic spectral simulator (see [1]–[5], [20]–[27] and the references therein).

II. Related Work and Background Review

A. Related Work on Density Estimation

Let x be a random variable, both kernel density estimation [38]–[40] and asymptotic probability extraction [29], [30] aim to approximate its probability density function $\rho(x)$.

Kernel Density Estimation: with N samples for x, kernel density estimation approximates its probability density function by using a set of kernel functions. The probability density function generated by kernel density estimation is non-negative, and the resulting cumulative density function is bounded in $[0, 1]$. However, kernel density estimation is seldom used in circuit modeling due to several shortcomings. First, the approximated probability density function is not compact: one has to store all samples as the parameters of a density function, which is inefficient for reuse in a stochastic simulator. Second, it is not straightforward to generate samples from the approximated probability density function. Third, the accuracy of kernel density estimation highly depends on the specific forms of the kernel functions (although Gaussian kernel seems suitable for the examples used in this work) as well as some parameters (e.g., the smoothing parameter). This paper will not construct the closed-form probability density function by kernel density estimation, instead we will use the *numerical* results from kernel density estimation as a reference for accuracy comparison.

Asymptotic Probability Extraction: if x is a linear quadratic function of some lower-level Gaussian parameters $\vec{\xi}$, asymptotic probability extraction [29], [30] can efficiently approximate $\rho(x)$ by moment matching. It has become the mainstream algorithm used in statistical circuit yield analysis and optimization. Asymptotic probability extraction and its variant [31] treat $\rho(x)$ as the impulse response of a linear time-invariant system, then approximates $\rho(x)$ using asymptotic waveform evaluation [41]. Several shortcomings have limited the application of asymptotic probability extraction and its variants:

1) Some assumptions of asymptotic probability extraction may not hold: i) $\vec{\xi}$ are assumed Gaussian variables, whereas in reality $\vec{\xi}$ can be non-Gaussian; ii) x may not be a linear quadratic function of $\vec{\xi}$; iii) the statistical moments of x may be unbounded, and thus asymptotic waveform evaluation [41] cannot be applied.

2) The density functions from moment matching may be physically inconsistent. The cumulative density function may have oscillations and the probability density function may be negative, as shown by [29], [30] and the recent work [42], as well as by our experiments in Section VI-D. This is because the impulse response of a linear system is not guaranteed non-negative when generated by asymptotic waveform evaluation [41]. Negative probability density functions cannot be used for the stochastic simulation of a physical model.

3) The resulting density function may blow up, as shown in Section VI-D and in [42]. There are two reasons for that. First, inaccurate moment computation can cause positive poles for a linear system, leading to an unbounded time-domain response. Second, asymptotic waveform evaluation is numerically unstable, which is well known in interconnect macromodeling. This is one of the important reasons why the model order reduction community has switched to implicit moment matching by Krylov-subspace projection.

B. Generalized Polynomial-Chaos Basis Function and Gauss Quadrature

In order to apply stochastic spectral methods, one normally needs a set of generalized polynomial-chaos basis functions to approximate the stochastic solution of a physical model.

Very often, a proper quadrature rule such as Gauss quadrature method [43] is also required to set up a deterministic equation in intrusive solvers such as stochastic Galerkin [4], [5] and stochastic testing [1]–[3], or to recover the coefficients of each basis function for the solution in non-intrusive (i.e., sampling-based) solvers such as stochastic collocation [20]–[23].

Basis Function Construction. Given $\rho(x)$ (the probability density function of x), the generalized polynomial-chaos basis functions of x are a set of orthonormal polynomials

$$\int_{\mathbb{R}} \phi_i(x)\phi_j(x)\rho(x)dx = \delta_{i,j} \tag{2}$$

where integers i and j denote the polynomial degrees, and $\delta_{i,j}$ is a Delta function. In order to obtain $\phi_i(x)$'s, a set of orthogonal polynomials $\pi_i(x)$'s are first constructed via the well-known three-term recurrence relation [44]:

$$\begin{aligned} &\pi_{i+1}(x) = (x-\gamma_i)\,\pi_i(x) - \kappa_i\pi_{i-1}(x), \quad i = 0,\ 1, \cdots \\ &\pi_{-1}(x) = 0, \quad \pi_0(x) = 1, \end{aligned} \tag{3}$$

where

$$\gamma_i = \frac{\int_{\mathbb{R}} x\pi_i^2(x)\rho(x)dx}{\int_{\mathbb{R}} \pi_i^2(x)\rho(x)dx},\ \kappa_{i+1} = \frac{\int_{\mathbb{R}} \pi_{i+1}^2(x)\rho(x)dx}{\int_{\mathbb{R}} \pi_i^2(x)\rho(x)dx},\quad i = 0, 1, \cdots, \tag{4}$$

and $\kappa_0 = 1$. Here $\pi_i(x)$ is a degree-i polynomial with leading coefficient 1. After that, the first $\hat{n}+1$ basis functions are obtained by normalization:

$$\phi_i(x) = \frac{\pi_i(x)}{\sqrt{\kappa_0\kappa_1\cdots\kappa_i}},\ \text{for } i = 0, 1, \cdots, \hat{n}. \tag{5}$$

The obtained univariate basis functions can be easily extended to the multivariate cases, as detailed in Section II-A of [3].

Gauss-Quadrature Rule. When computing an integral with Gauss quadrature [43] one typically uses the expression

$$\int_{\mathbb{R}} g(x)\rho(x)dx \approx \sum_{j=1}^{\hat{n}+1} g(x^j)w^j \tag{6}$$

which provides an exact result if $g(x)$ is a polynomial of degree $\leq 2\hat{n}+1$. The quadrature points x^j's and weights w^j's depend on $\rho(x)$. Define a symmetric tridiagonal matrix

$$\mathbf{J} = \begin{bmatrix} \gamma_0 & \sqrt{\kappa_1} & & & \\ \sqrt{\kappa_1} & \gamma_1 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \gamma_{\hat{n}-1} & \sqrt{\kappa_{\hat{n}}} \\ & & & \sqrt{\kappa_{\hat{n}}} & \gamma_{\hat{n}} \end{bmatrix}, \tag{7}$$

and let its eigenvalue decomposition be $\mathbf{J} = \mathbf{U}\Sigma\mathbf{U}^T$, where $\mathbf{U}$ is a unitary matrix. Denote the (i,j) entry of $\mathbf{U}$ by $u_{i,j}$, then x^j is the j-th diagonal element of Σ, and the corresponding weight w^j is $u_{1,j}^2$ [43]. Using tensor product or sparse grids, 1-D Gauss quadrature rules can be easily extended to multi-dimensional cases [20]–[23].

Remark 2.1: The main bottleneck of the above procedures lies in (4), which requires computing a set of integrals. This step can be non-trivial if $\rho(x)$ is not in a proper form. When such integrals are not accurately computed, the constructed basis functions can be erroneous. Furthermore, κ_i's may become negative, making computing the Gauss-quadrature points and weights impossible.

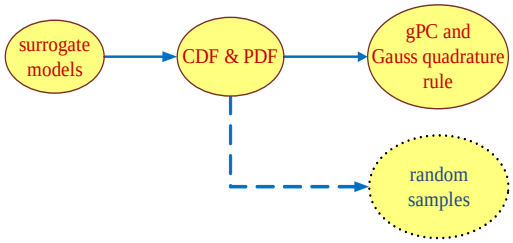


Fig. 2. Construct generalized polynomial-chaos (gPC) bases and Gauss quadrature rules from surrogate models. Here CDF and PDF means "cumulative density function" and "probability density function", respectively.

III. The Proposed Framework

In this paper, $\hat{x}_i$'s in Fig. 1 are assumed mutually independent. With this assumption, we can consider each surrogate model independently. For simplicity, let

$$\hat{x} = f(\vec{\xi}),\ \text{with } \vec{\xi} \in \mathbb{R}^d \tag{8}$$

represent a general surrogate model. We employ the linear transformation

$$x = \frac{\hat{x}-a}{b} \tag{9}$$

to define a new random input x, which aims to improve the numerical stability. Once we obtain the cumulative density function and probability density function of x (denoted as $p(x)$ and $\rho(x)$, respectively), then the cumulative density function and probability density function of $\hat{x}$ can be obtained by

$$\hat{p}(\hat{x}) = p\left(\frac{\hat{x}-a}{b}\right) \text{ and } \hat{\rho}(\hat{x}) = \frac{1}{b}\rho(\frac{\hat{x}-a}{b}) \tag{10}$$

respectively.

As shown in Fig. 2, we first construct the density functions of x *in a proper way*, then we determine the generalized polynomial-chaos bases of x and a proper Gauss quadrature rule based on the obtained density functions. With the obtained cumulative density function, random samples of x could be easily obtained for higher-level Monte Carlo-based simulation, however such task is not the focus of this paper. Our proposed framework consists of the following steps.

- **Step 1.** Use N Monte Carlo samples (or readily available measurement/simulation data) to obtain the discrete cumulative density function curve of $\hat{x} = f(\vec{\xi})$. Since $f(\vec{\xi})$ is a surrogate model, this step can be extremely efficient.
- **Step 2.** Let $\delta > 0$ be a small threshold value, $\hat{x}_{\min}$ and $\hat{x}_{\max}$ be the minimum and maximum values of $\hat{x}$ from the Monte Carlo analysis (or available data), respectively. We set a=$\hat{x}_{\min} - \delta$, b=$\hat{x}_{\max} + \delta - a$, then N samples of x in the interval $(0, 1)$ are obtained by the

linear transformation (9). The obtained samples provide a discrete cumulative density function for x.

- **Step 3.** From the obtained cumulative density function curve of x, pick $n \ll N$ points (x_i, y_i) for $i = 1, \cdots, n$. Here x_i denotes the value of x, and y_i the corresponding cumulative density function value. The data are monotone: $x_i < x_{i+1}$ and $0 = y_1 \leq \cdots \leq y_n = 1$.
- **Step 4.** Use a monotone interpolation algorithm in Section IV to construct a closed-form function $p(x)$ to approximate the cumulative density function of x.
- **Step 5.** Compute the first-order derivative of $p(x)$ and use it as a closed-form approximation to $\rho(x)$.
- **Step 6.** With the obtained $\rho(x)$, we utilize the procedures in Section V to construct the generalized polynomial-chaos basis functions and Gauss quadrature points/weights for x.

Many surrogate models are described by truncated generalized polynomial chaos expansions. The cost of evaluating such models may increase dramatically when the lower-level parameters $\vec{\xi}$ have a high dimensionality (which may occasionally happen), although the surrogate model evaluation is still much faster than the detailed simulation. Fortunately, in practical high-dimensional stochastic problems, normally only a small number of parameters are important to the output and most cross terms will vanish [45]–[47]. Consequently, a highly sparse generalized polynomial chaos expansion can be utilized for fast evaluation. Furthermore, when the coupling between the random parameters are weak, quasi-Monte Carlo [48] can further speed up the surrogate model evaluation.

In Step 3, we first select $(x_1, y_1) = (0, 0)$ and $(x_n, y_n) = (1, 1)$. The n data points are selected such that

$$|x_{i+1} - x_i| \leq \frac{1}{m} \text{ and } |y_{i+1} - y_i| \leq \frac{1}{m}, \tag{11}$$

where m is an integer used to control n. This constraint ensures that the interpolation points are selected properly such that the behavior around the peak of $\rho(x)$ is well captured. In practical implementation, for $k = 2, \cdots, n-1$, the point (x_k, y_k) is selected from the cumulative density function curve subject to the following criteria:

$$\sqrt{(y_{k-1} - y_k)^2 + (x_{k-1} - x_k)^2} \approx \frac{1}{m}. \tag{12}$$

For $x \notin [x_1, x_n]$, we set $\rho(x)$=0. This treatment introduces some errors in the tail regions. Approximating the tail regions is non-trivial, but such errors may be ignored if rare failure events are not a major concern (e.g., in the yield analysis of some analog/RF circuits).

Remark 3.1: Similar to standard stochastic spectral simulators [1]–[5], [20]–[27], this paper assumes that $\hat{x}_i$'s are mutually independent. It is more difficult to handle correlated and non-Gaussian random inputs. Not only is it difficult to construct the density functions, but also it is hard to construct the basis functions even if the multivariate density function is given [19], [49]. How to handle correlated non-Gaussian random inputs remains an open and important topic in uncertainty quantification [19]. Some of our progress in this direction will be reported in [50].

The most important parts of our algorithm are Step 4 and Step 6. In Section IV we will show how we guarantee that the obtained density functions are physically consistent. Step 6 will be detailed in Section V, with emphasis on an efficient analytical implementation.

IV. Implementation of the Density Estimator

This section presents the numerical implementation of our proposed density estimation. Our implementation is based on two monotone interpolation techniques, which are well studied in the mathematical community but have not been applied to uncertainty quantification. Since we approximate the cumulative density function $p(x)$ in the interval $x \in [x_1, x_n]$, in both methods we set $p(x)$=y_1=0 for $x < x_1$ and $p(x)$=y_n=1 for $x > x_n$, respectively.

A. Method 1: Piecewise Cubic Interpolation

Our first implementation uses a piecewise cubic interpolation [34], [35]. With the monotone data from Step 3 of Section III, we construct $p(x)$ as a cubic polynomial:

$$p(x) = c_k^1 + c_k^2(x - x_k) + c_k^3(x - x_k)^2 + c_k^4(x - x_k)^3 \tag{13}$$

for $x \in [x_k, x_{k+1}]$, $0 < k < n$. If y_k=y_{k+1}, we simply set c_k^1=y_k and c_k^2=c_k^3=c_k^4= 0. Otherwise, the coefficients are selected according to the following formula [35]

$$\begin{array}{l} c_k^1 = y_k, \;\; c_k^2 = \dot{y}_k, \\ c_k^3 = \frac{s_k - \dot{y}_{k+1} - 2\dot{y}_k}{\Delta x_k}, \; c_k^4 = \frac{2s_k - \dot{y}_{k+1} - \dot{y}_k}{(\Delta x_k)^2} \end{array} \tag{14}$$

where Δx_k=x_{k+1}−x_k, s_k=$\frac{y_{k+1}-y_k}{\Delta x_k}$. This formula ensures that $p(x)$ and $p'(x)$ are continuous, $p(x_k) = y_k$ and $p'(x_k) = \dot{y}_k$. Here $p'(x)$ denotes the 1st-order derivative of $p(x)$.

The key of this implementation is how to compute $\dot{y}_k$ such that the interpolation is accurate and $p(x)$ is non-decreasing. The value of $\dot{y}_k$ is decided by two steps. First, we compute the first-order derivative $\dot{y}(x_k)$ by a parabolic method:

$$\dot{y}(x_k) = \begin{cases} \frac{s_1(2\Delta x_1 + \Delta x_2) - s_2 \Delta x_1}{x_3 - x_1}, & \text{if } k = 1 \\ \frac{s_{n-1}(2\Delta x_{n-1} + \Delta x_{n-2}) - s_{n-2}\Delta x_{n-1}}{x_n - x_{n-2}}, & \text{if } k = n \\ \frac{s_k \Delta x_{k-1} + s_{k-1}\Delta x_k}{x_{k+1} - x_{k-1}}, & \text{if } 2 < k < n-1. \end{cases} \tag{15}$$

This parabolic method has a 2nd-order accuracy [35]. Second, $\dot{y}_k$ is obtained by perturbing $\dot{y}(x_k)$ (if necessary) to enforce the monotonicity of $p(x)$. The monotonicity of $p(x)$ is equivalent to $p'(x) \geq 0$, which is a 2nd-order inequality. By solving this inequality, a feasible region for $\dot{y}_k$, denoted by $\mathcal{A}$, is provided in [34]. Occasionally we need to project $\dot{y}(x_k)$ onto $\mathcal{A}$ to get $\dot{y}_k$ if $\dot{y}(x_k) \notin \mathcal{A}$. In practice, we use the simpler projection method suggested by [35]:

$$\dot{y}_k = \begin{cases} \min\left(\max\left(0, \dot{y}(x_k)\right), 3s_{\min}^k\right), & \text{if } s_k s_{k-1} > 0 \\ 0, & \text{if } s_k s_{k-1} = 0 \end{cases} \tag{16}$$

with s_0=s_1, s_n=s_{n-1} and $s_{\min}^k$=$\min(s_k, s_{k-1})$. The above procedure projects $\dot{y}(x_k)$ onto a subset of $\mathcal{A}$, and thus the monotonicity of $p(x)$ is guaranteed.

Algorithm 1 piecewise cubic density estimation

1: Evaluate the model (8) to obtain N samples of $\hat{x}$;
2: Shift and scale $\hat{x}$ to obtain N samples for x;
3: Pick n data points (x_k, y_k), under constraint (11);
4: Calculate $\dot{y}(x_k)$ using the parabolic method (15);
5: **for** $k = 1, \cdots, n$ **do**
6: **if** $y_k = y_{k+1}$, set $c_k^1{=}y_k$ and $c_k^2{=}c_k^3{=}c_k^4{=}0$;
7: **else**
8: Compute $\dot{y}_k$ according to (16);
9: Compute the coefficients in (14).
10: **end**
11: **end for**

Algorithm 2 piecewise rational quadratic density estimation

1: Evaluate the model (8) to obtain N samples of x;
2: Shift and scale $\hat{x}$ to obtain N samples for x;
3: Pick n data points (x_k, y_k), under constraint (11);
4: **for** $k = 1, \cdots, n$ **do**
5: Calculate $\dot{y}_k$ using the formula in (20);
6: **if** $y_k = y_{k+1}$
7: set $\alpha_k^1 = y_k$, $\beta_k^1 = 1$ and other coefficients to zero;
8: **else**
9: compute the coefficients of $N(x)$ and $D(x)$ using (19).
10: **end**
11: **end for**

Once $p(x)$ is constructed, the probability density function of x can be obtained by

$$\rho(x) = p'(x) = c_k^2 + 2c_k^3(x - x_k) + 3c_k^4(x - x_k)^2 \qquad (17)$$

for $x_k \leq x \leq x_{k+1}$. Note that for $x \notin [x_1, x_n]$, $p'(x) = 0$.

Calculating $p'(x)$ may amplify the interpolation errors. However, the error is acceptable since the constructed $p(x)$ is smooth enough and $p'(x)$ is continuous. The pseudo codes of Algorithm 1 summarize the steps of this approach.

B. Method 2: Piecewise Rational Quadratic Interpolation

Our second implementation is based on a piecewise rational quadratic interpolation [36], [37]. In this implementation, we approximate the cumulative density function of x by

$$p(x) = \frac{N(x)}{D(x)} = \frac{\alpha_k^1 + \alpha_k^2 x + \alpha_k^3 x^2}{\beta_k^1 + \beta_k^2 x + \beta_k^3 x^2} \qquad (18)$$

for $x \in [x_k, x_{k+1}]$. The coefficients are selected by the following method: when $x_k = x_{k+1}$, we set $\alpha_k^1 = y_k$, $\beta_k^1 = 1$ and all other coefficients to zero; otherwise, the coefficients are decided according to the formula

$$\begin{array}{l} \alpha_k^1 = y_{k+1}x_k^2 - w_k x_k x_{k+1} + y_k x_{k+1}^2, \\ \alpha_k^2 = w_k(x_k + x_{k+1}) - 2y_{k+1}x_k - 2y_k x_{k+1}, \\ \alpha_k^3 = y_{k+1} - w_k + y_k, \\ \beta_k^1 = x_k^2 - v_k x_k x_{k+1} + x_{k+1}^2, \\ \beta_k^2 = v_k(x_k + x_{k+1}) - 2x_k - 2x_{k+1},\ \beta_k^3 = 2 - v_k, \\ \text{with } w_k = \frac{y_{k+1}\dot{y}_k + y_k\dot{y}_{k+1}}{s_k} \text{ and } v_k = \frac{\dot{y}_k + \dot{y}_{k+1}}{s_k} \end{array} \qquad (19)$$

where s_k is defined the same as in piecewise cubic interpolation. In this interpolation scheme, the sufficient and necessary condition for the monotonicity of $p(x)$ is very simple: $\dot{y}_k \geq 0$. In order to satisfy this requirement, the slope $\dot{y}_k$ is approximated by the geometric mean

$$\dot{y}_k = \begin{cases} (s_1)^{\frac{x_3 - x_1}{x_3 - x_2}} (s_{3,1})^{\frac{x_1 - x_2}{x_3 - x_2}}, & \text{if } k = 1 \\ (s_{n-1})^{\frac{x_n - x_{n-2}}{x_{n-1} - x_{n-2}}} (s_{n,n-2})^{\frac{x_{n-1} - x_n}{x_{n-1} - x_{n-2}}}, & \text{if } k = n \\ (s_{k-1})^{\frac{x_{k+1} - x_k}{x_{k+1} - x_{k-1}}} (s_k)^{\frac{x_k - x_{k-1}}{x_{k+1} - x_{k-1}}}, & \text{if } 1 < k < n \end{cases} \qquad (20)$$

with $s_{k_1,k_2} = \frac{y_{k_1} - y_{k_2}}{x_{k_1} - x_{k_2}}$. Similarly, the probability density function of x can be approximated by

$$\rho(x) = p'(x) = \frac{N'(x)D(x) - D'(x)N(x)}{D^2(x)}, \qquad (21)$$

for $x \in [x_k, x_{k+1}]$.

Note that in piecewise cubic interpolation, a projection procedure is not required, since the monotonicity of $p(x)$ is automatically guaranteed. The pseudo codes of this density estimation method are provided in Algorithm 2.

C. Properties of $p(x)$

It is straightforward to show that the obtained density functions are physically consistent: 1) $p(x)$ is differentiable, and thus its derivative $p'(x)$ always exists; 2) $p(x)$ is monotonically increasing from 0 to 1, and the probability density function $\rho(x)$ is non-negative.

We can easily draw a random sample from the obtained $p(x)$. Let $y \in [0, 1]$ be a sample from a uniform distribution, then a sample of x can be obtained by solving $p(x) = y$ in the interval $y \in [y_k, y_{k+1}]$. This procedure only requires computing the roots of a cubic (or quadratic) polynomials, resulting in a unique solution $x \in [x_k, x_{k+1}]$. This property is very useful in uncertainty quantification. Not only are random samples used in Monte Carlo simulators, but also they can be used in stochastic spectral methods. Recently, compressed sensing has been applied to high-dimensional stochastic problems [45]–[47]. In compressed sensing, random samples are normally used to enhance the restricted isometry property of the dictionary matrix [51].

Finally, it becomes easy to determine the generalized polynomial-chaos basis functions and a proper quadrature rule for x due to the special form of $\rho(x)$. This issue will be discussed in Section V.

Remark 4.1: Our proposed density estimator only requires some interpolation points from a discrete cumulative density function curve. The interpolation points actually can be obtained by any appropriate approach. For example, kernel density estimation will be a good choice if we know a proper kernel function and a good smoothing parameter based on *a-priori* knowledge. When the surrogate model is a linear quadratic function of Gaussian variables, we may first employ asymptotic probability extraction [29] to generate a physically inconsistent cumulative density function. After that, some monotone data points (with y_i's bounded by 0 and 1) can be selected to generate a piecewise cubic or piecewise rational quadratic cumulative density function. The new cumulative density function and probability density function become

physically consistent and can be reused in a stochastic simulator.

V. Determine Basis Functions and Gauss Quadrature Rules

This section shows how to calculate the generalized polynomial-chaos bases and the Gauss quadrature points/weights of x based on the obtained density function.

A. Proposed Implementation

One of the many usages of our density estimator is to fast compute a set of generalized polynomial-chaos basis functions and Gauss quadrature points/weights by analytically computing the integrals in (4). Let $\pi_i^2(x) = \sum_{k=0}^{2i} \tau_{i,k} x^k$, then we have

$$\begin{aligned} \int_{\mathbb{R}} x\pi_i^2(x)\rho(x)dx &= \sum_{k=0}^{2i} \tau_{i,k} M_{k+1}, \\ \int_{\mathbb{R}} \pi_i^2(x)\rho(x)dx &= \sum_{k=0}^{2i} \tau_{i,k} M_k \end{aligned} \tag{22}$$

where M_k denotes the k-th statistical moments of x. By exploiting the special form of our obtained density function, the statistical moments can be computed as

$$M_k = \int_{-\infty}^{+\infty} x^k\rho(x)dx = \int_{x_1}^{x_n} x^k\rho(x)dx = \sum_{j=1}^{n-1} I_{j,k} \tag{23}$$

where $I_{j,k}$ denotes the integral in the j-th piece:

$$I_{j,k} = \int_{x_j}^{x_{j+1}} x^k\rho(x)dx = F_{j,k}(x_{j+1}) - F_{j,k}(x_j). \tag{24}$$

Here $F_{j,k}(x)$ is a continuous analytical function under the constraint $\frac{d}{dt}F_{j,k}(x) = x^k\rho(x)$ for $x \in [x_j, x_{j+1}]$. The key problem of our method is to construct $F_{j,k}(x)$. When $\rho(x)$ is obtained from Alg. 1 or Alg. 2, we can easily obtain the closed form of $F_{j,k}(x)$, as will be elaborated in Section V-B and Section V-C.

Remark 5.1: This paper directly applies (4) to compute the recurrence parameters γ_i and κ_i. As suggested by [44], modified Chebyshev algorithm [52] can improve the numerical stability when constructing high-order polynomials. Modified Chebyshev algorithm indirectly computes γ_i and κ_i by first evaluating a set of modified moments. Again, if we employ the $\rho(x)$ obtained from our proposed density estimators, then the calculation of modified moments can also be done analytically to further improve the accuracy and numerical stability.

B. Construct $F_{j,k}(x)$ using the Density Function from Alg. 1

When $\rho(x)$ is constructed by Alg. 1, $x^k\rho(x)$ is a polynomial function of at most degree $k+2$ inside the interval $[x_j, x_{j+1}]$. Therefore, the analytical form of $F_{j,k}(x)$ is

$$F_{j,k}(x) = \mathbf{a}_{j,k}x^{k+3} + \mathbf{b}_{j,k}x^{k+2} + \mathbf{c}_{j,k}x^{k+1} \tag{25}$$

with

$$\begin{aligned} \mathbf{a}_{j,k} &= \tfrac{3c_j^4}{k+3}, \ \mathbf{b}_{j,k} = \tfrac{2c_j^3 - 6c_j^4 x_j}{k+2}, \\ \mathbf{c}_{j,k} &= \tfrac{c_j^2 - 2c_j^3 x_j + 3c_j^4 x_j^2}{k+1}. \end{aligned}$$

C. Construct $F_{j,k}(x)$ using the Density Function from Alg. 2

If $\rho(x)$ is constructed by Alg. 2, for any $x \in [x_j, x_{j+1}]$ we rewrite $x^k\rho(x)$ as follows

$$\begin{aligned} x^k\rho(x) &= \tfrac{x^k\left[N'(x)D(x) - D'(x)N(x)\right]}{D^2(x)} \\ &= \tfrac{d}{dx}\left(\tfrac{x^kN(x)}{D(x)}\right) - \tfrac{kx^{k-1}N(x)}{D(x)}. \end{aligned}$$

Therefore, $F_{j,k}(x)$ can be selected as

$$F_{j,k}(x) = \frac{x^kN(x)}{D(x)} - \tilde{F}_{j,k}(x), \text{ with } \frac{d}{dx}\tilde{F}_{j,k}(x) = \frac{kx^{k-1}N(x)}{D(x)}.$$

In order to obtain $\tilde{F}_{j,k}(x)$, we perform a long division:

$$\frac{kx^{k-1}N(x)}{D(x)} = \tilde{P}_{j,k}(x) + \frac{\tilde{R}_{j,k}(x)}{D(x)} \tag{26}$$

where $\tilde{P}_{j,k}(x)$ and $\tilde{R}_{j,k}(x)$ are both polynomial functions, and $\tilde{R}_{j,k}(x)$ has a lower degree than $D(x)$. Consequently,

$$\tilde{F}_{j,k}(x) = \tilde{F}^1_{j,k}(x) + \tilde{F}^2_{j,k}(x) \tag{27}$$

where $\tilde{F}^1_{j,k}(x)$ and $\tilde{F}^2_{j,k}(x)$ are the integrals of $\tilde{P}_{j,k}(x)$ and $\frac{\tilde{R}_{j,k}(x)}{D(x)}$, respectively. It is trivial to obtain $\tilde{F}^1_{j,k}(x)$ since $\tilde{P}_{j,k}(x)$ is a polynomial function.

The closed form of $\tilde{F}^2_{j,k}(x)$ is decided according to the coefficients of $D(x)$ and $\tilde{R}_{j,k}(x)$, as is summarized below.

Case 1: if $\beta_j^3 \neq 0$, then $\tilde{R}_{j,k}(x) = \tilde{r}^0_{j,k} + \tilde{r}^1_{j,k}x$. Let us define $\Delta_j := 4\beta_j^1\beta_j^3 - \beta_j^2$, then we can select $\tilde{F}^2_{j,k}(x)$ according to the formula in (28).

Case 2: if $\beta_j^3 = 0$ and $\beta_j^2 \neq 0$, then $\tilde{R}_{j,k}(x) = \tilde{r}^0_{j,k}$ is a constant. In this case, we select

$$\tilde{F}^2_{j,k}(x) = \frac{\tilde{r}^0_{j,k}}{\beta_j^2} \ln\left|\beta_j^2 x + \beta_j^1\right|. \tag{29}$$

Case 3: if $\beta_j^3 = \beta_j^2 = 0$, then $\tilde{R}_{j,k}(x) = 0$. In this case we set $\tilde{F}^2_{j,k}(x) = 0$.

Remark 5.2: Occasionally, the projection procedure (16) in Alg. 1 may cause extra errors at the end points of some intervals. If this problem happens we recommend to use Alg. 2. On the other hand, if high-order basis functions is required we recommend Alg. 1, since the moment computation with the density from Alg. 2 is numerically less stable (due to the long-term division and the operations in (28).

VI. Numerical Examples

This section presents the numerical results on a synthetic example and the statistical surrogate models from two practical analog/RF circuits. The surrogate models of these practical circuits are extracted from transistor-level simulation using the fast stochastic circuit simulator developed in [1]–[3]. All experiments are run in Matlab on a 2.4GHz 4-GB RAM laptop.

In the following experiments, we use the density functions from kernel density estimation as the "reference solution" because: 1) as a standard technique, kernel density estimation is most widely used in mathematics and engineering; 2) kernel density estimation guarantees that the generated probability density function is non-negative, whereas asymptotic probability extraction cannot; 3) Gaussian kernel function seems to

$$\tilde{F}^2_{j,k}(x) = \begin{cases} \frac{\tilde{r}^1_{j,k}}{2\beta^3_j} \ln\left|\beta^3_j x^2 + \beta^2_j x + \beta^1_j\right| + \frac{2\beta^3_j \tilde{r}^0_{j,k} - \beta^2_j \tilde{r}^1_{j,k}}{\beta^3_j \sqrt{\Delta_j}} \arctan \frac{2\beta^3_j x + \beta^2_j}{\sqrt{\Delta_j}}, & \text{if } \Delta_j > 0 \\ \frac{\tilde{r}^1_{j,k}}{2\beta^3_j} \ln\left|\beta^3_j x^2 + \beta^2_j x + \beta^1_j\right| - \frac{2\beta^3_j \tilde{r}^0_{j,k} - \beta^2_j \tilde{r}^1_{j,k}}{\beta^3_j \sqrt{-\Delta_j}} \arctan \frac{2\beta^3_j x + \beta^2_j}{\sqrt{-\Delta_j}}, & \text{if } \Delta_j < 0 \\ \frac{\tilde{r}^1_{j,k}}{2\beta^3_j} \ln\left|\beta^3_j x^2 + \beta^2_j x + \beta^1_j\right| - \frac{2\beta^3_j \tilde{r}^0_{j,k} - \beta^2_j \tilde{r}^1_{j,k}}{\beta^3_j \left(2\beta^3_j x + \beta^2_j\right)}, & \text{if } \Delta_j = 0 \end{cases} \quad (28)$$

be a good choice for the examples in this paper. However, it is worth noting that the density functions from kernel density estimation are not efficient for reuse in higher-level stochastic simulation. We plot the density functions of $\hat{x}$ (the original random input) instead of x (the new random input after a linear transformation) since the original one is physically more intuitive. To verify the accuracy of the computed generalized polynomial-chaos bases and Gauss quadrature points/weights, we define a symmetric matrix $\mathbf{V}_{\hat{n}+1} \in \mathbb{R}^{(\hat{n}+1)\times(\hat{n}+1)}$, the (i,j) entry of which is

$$v_{i,j} = \sum_{k=1}^{\hat{n}+1} w^k \phi_{i-1}\left(x^k\right) \phi_{j-1}\left(x^k\right).$$

Here x^k and and w^k are the computed k-th Gauss quadrature point and weight, respectively. Therefore $v_{i,j}$ approximates the inner product of $\phi_{i-1}(x)$ and $\phi_{j-1}(x)$, defined as $\int_{\mathbb{R}} \phi_{i-1}(x)\,\phi_{j-1}(x)\,\rho(x)dx$, by $\hat{n}+1$ quadrature points. Let $\mathbf{I}_{\hat{n}+1}$ be an identity matrix, then we define an error:

$$\epsilon = ||\mathbf{I}_{\hat{n}+1} - \mathbf{V}_{\hat{n}+1}||_\infty \quad (30)$$

which is close to zero when our constructed basis functions and Gauss-quadrature points/weights are accurate enough.

A. Synthetic Example

As a demonstration, we first consider the following synthetic example with four random parameters $\vec{\xi} = [\xi_1, \cdots, \xi_4]$:

$$\hat{x} = f(\vec{\xi}) = \xi_1 + 0.5\exp(0.52\xi_2) + 0.3\sqrt{2.1 \times |\xi_4|} + \sin(\xi_3)\cos(3.91\xi_4)$$

where ξ_1, ξ_2 and ξ_3 are all standard Gaussian random variables, and ξ_4 has a uniform distribution in the interval $[-0.5, 0.5]$. This model is strongly nonlinear with respect to $\vec{\xi}$ due to the exponential, triangular and square root functions. It is also non-smooth at $\xi_4 = 0$ due to the third term in the model. This model is designed to challenge our algorithm. Using this surrogate model, 10^6 samples of x are easily created to generate the cumulative density function curve within 1 second.

Density Estimation: we set $m = 45$ and select 74 data points from the obtained cumulative density function curve using the constraint in (12). After that, both Alg. 1 and Alg. 2 are applied to generate $p(x)$ and $\rho(x)$ as approximations to the cumulative density function and probability density function of x, respectively. The CPU times cost by our proposed density estimators are in millisecond scale, since only simple algebraic operations are required. After scaling by (10), the cumulative density function and probability density function of the original random input $\hat{x}$ ($\hat{p}(\hat{x})$ and $\hat{\rho}(\hat{x})$, respectively) from both

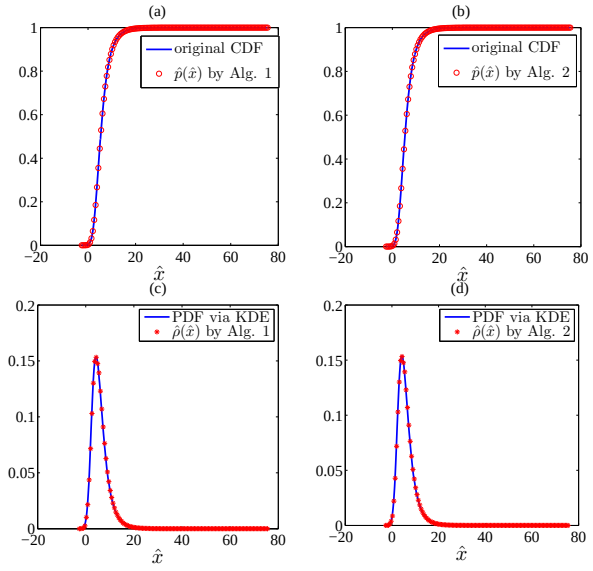


Fig. 3. Cumulative density function (CDF) and probability density function (PDF) approximation of $\hat{x}$ for the synthetic example. The reference PDF is generated by kernel density estimation (KDE).

algorithms are compared with the original cumulative density function and probability density function in Fig. 3. Clearly, $\hat{p}(\hat{x})$ is indistinguishable with the original cumulative density function (from Monte Carlo simulation); and $\hat{\rho}(\hat{x})$ overlaps with the original probability density function (estimated by kernel density estimation using Gaussian kernels). Note that the results from kernel density estimation are not efficient for reuse in higher-level stochastic simulation, since all Monte Carlo samples are used as parameters of the resulting density function.

It is clearly shown that the generated $\hat{p}(\hat{x})$ [and thus $p(x)$] is monotonically increasing from 0 to 1, and that the generated $\hat{\rho}(\hat{x})$ [and thus $\rho(x)$] is non-negative. Therefore, the obtained density functions are physically consistent.

Basis Function: Using the obtained density functions and the proposed implementation in Section V, a set of orthonormal polynomials $\phi_k(x)$'s are constructed as the basis functions at the cost of milliseconds. Fig. 4 show the first five generalized polynomial-chaos basis functions. Note that although the computed basis functions from two methods are graphically indistinguishable, they are actually slightly different since Alg. 1 and Alg. 2 generate different representations for $\rho(x)$.

Gauss Quadrature Rule: setting $\hat{n} = 4$, five Gauss quadrature points and weights are generated using the method presented in Section V. Table I shows the results from two kinds of approximated density functions. Clearly, since the probability density functions from Alg. 1 and Alg. 2 are different, the

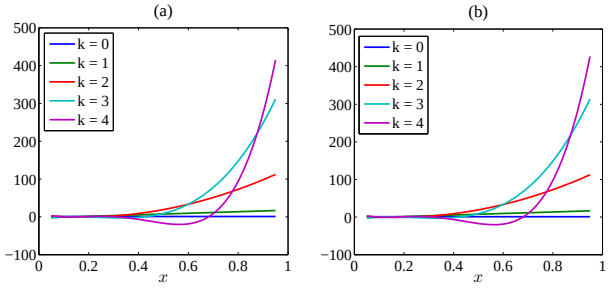


Fig. 4. Computed generalized polynomial-chaos basis functions $\phi_k(x)$ ($k = 0, \cdots, 4$) for the synthetic example. (a) uses the probability density function from Alg. 1, and (b) uses the probability density function from Alg. 2.

TABLE I
COMPUTED GAUSS QUADRATURE POINTS AND WEIGHTS FOR THE SYNTHETIC EXAMPLE.

with $\rho(x)$ from Alg. 1		with $\rho(x)$ from Alg. 2	
x^k	w^k	x^k	w^k
0.082620	0.311811	0.084055	0.332478
0.142565	0.589727	0.144718	0.576328
0.249409	0.096115	0.252980	0.089027
0.458799	0.002333	0.463207	0.002150
0.837187	0.000016	0.835698	0.000016

resulting quadrature points/weights are also slightly different. The results from both probability density functions are very accurate. Using the probability density function from Alg. 1, we have $\epsilon = 2.24 \times 10^{-14}$, and the error (30) is 7.57×10^{-15} if $\rho(x)$ from Alg. 2 is employed.

B. Colpitts Oscillator

We now test our proposed algorithm on a more practical example, the Colpitts oscillator circuit shown in Fig. 5. The design parameters of this circuit are R_1=2.2 kΩ, R_2=R_3=10 kΩ, C_2=100 pF, C_3=0.1μF, and α=0.992 for the BJT. The oscillation frequency is mainly determined by the values of L_1, C_1 and C_2. In this circuit, L_1=150 + $\mathcal{N}(0, 9)$ nH and C_1=100 + $\mathcal{U}(-10, 10)$ pF are random variables with Gaussian and uniform distributions, respectively. We construct a surrogate model using generalized polynomial chaos expansions and the stochastic shooting Newton solver in [2]. The oscillation frequency $f_{\rm osc}$ is expressed as

$$\hat{x} = f_{\rm osc} = f(\vec{\xi}) = \frac{1}{\sum_{k=1}^{10} T_k \psi_k(\vec{\xi})} \qquad (31)$$

where the denominator is a 3rd-order generalized polynomial chaos representation for the period of the oscillator, with $\psi_k(\vec{\xi})$ being the k-th multivariate generalized polynomial-chaos basis function of $\vec{\xi}$ and T_k the corresponding coefficient. Although the period is a polynomial function of $\vec{\xi}$, the frequency is not, due to the inverse operation. In order to extract the cumulative density function curve, 5×10^5 samples are utilized to evaluate the surrogate model (31) by Monte Carlo, which costs 225 seconds of CPU times on our Matlab platform.

Density Estimation: 106 data points on the obtained cumulative density function curve are used to construct $p(x)$ and $\rho(x)$, which costs only several milliseconds. After scaling

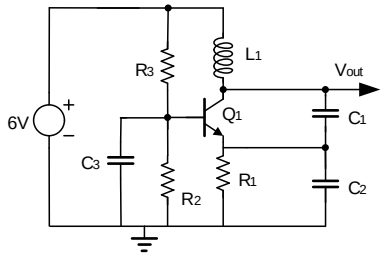


Fig. 5. Schematic of the Colpitts oscillator.

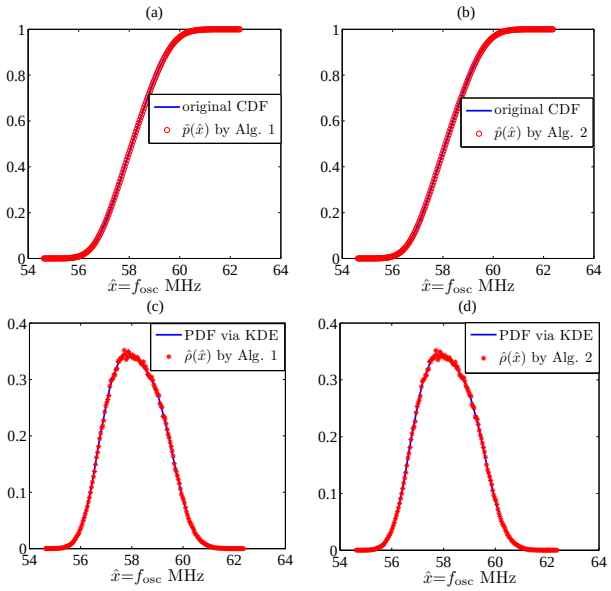


Fig. 6. Cumulative density function (CDF) and probability density function (PDF) approximation for the frequency of the Colpitts oscillator. The reference PDF is generated by kernel density estimation (KDE).

the constructed closed-form cumulative density functions and probability density functions from Alg. 1 and Alg. 2, the approximated density functions of the oscillation frequency are compared with the Monte Carlo results in Fig. 6. The constructed cumulative density functions by both methods are graphically indistinguishable with the result from Monte Carlo. The bottom plots in Fig. 6 also show a good match between our obtained $\hat{\rho}(\hat{x})$ with the result from kernel density estimation. Again, important properties of the density functions (i.e., monotonicity and boundedness of the cumulative density function, and non-negativeness of the probability density function) are well preserved by our proposed density estimation algorithms.

Basis Function: Using the obtained density functions and the proposed implementation in Section V, a set of orthonormal polynomials $\phi_k(x)$'s are constructed as the basis functions at the cost of milliseconds. Fig. 7 shows several generalized polynomial-chaos basis functions of x. Again, the basis functions resulting from our two density estimation implementations are only slightly different.

Gauss Quadrature Rule: the computed five Gauss quadrature points and weights are shown in Table II. Again the results from two density estimations are slightly different.

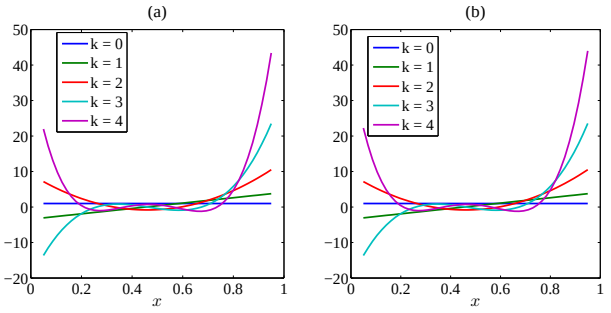


Fig. 7. Computed generalized polynomial-chaos basis functions $\phi_k(x)$ ($k = 0, \cdots, 4$) for the Colpitts oscillator. (a) uses the probability density function from Alg. 1, and (b) uses the probability density function from Alg. 2.

TABLE II
COMPUTED GAUSS QUADRATURE POINTS AND WEIGHTS FOR THE COLPITTS OSCILLATOR.

with $\rho(x)$ from Alg. 1		with $\rho(x)$ from Alg. 2	
x^k	w^k	x^k	w^k
0.170086	0.032910	0.170935	0.032456
0.309764	0.293256	0.310016	0.292640
0.469034	0.441303	0.468658	0.439710
0.632232	0.217359	0.631249	0.218274
0.788035	0.016171	0.786226	0.016820

The results from both probability density functions are very accurate. Using $\rho(x)$ from Alg. 1, we have $\epsilon = 1.3 \times 10^{-13}$, and the error is 1.45×10^{-13} if we use $\rho(x)$ from Alg. 2.

C. Low-Noise Amplifier

In this example we consider the statistical behavior of the total harmonic distortion at the output node of the low-noise amplifier shown in Fig. 8. The device ratios of the MOSFETs are W_1/L_1=W_2/L_2=500/0.35 and W_3/L_3=50/0.35. The linear components are R_1=50Ω, R_2=2 kΩ, C_1=10 pF, C_L=0.5 pF, L_1=20 nH and L_3=7 nH. Four random parameters are introduced to describe the uncertainties: ξ_1 and ξ_2 are standard Gaussian variables, ξ_3 and ξ_4 are standard uniform-distribution parameters. These random parameters are mapped to the physical parameters as follows: temperature T=300 + 40ξ_1 K influences transistor threshold voltage; $V_{\rm T}$=0.4238 + 0.1ξ_2 V represents the threshold voltage under zero $V_{\rm bs}$; R_3=0.9 + 0.2ξ_3 kΩ and L_2=0.8 + 1.2ξ_4 nH. The supply voltage is $V_{\rm dd}$=1.5 V, and the periodic input is $V_{\rm in} = 0.1\sin(4\pi \times 10^8 t)$ V.

The surrogate model for total harmonic distortion analysis is constructed by a numerical scheme as follows. First, the parameter-dependent periodic steady-state solution at the output is solved by the non-Monte Carlo simulator in [2], and is expressed by a truncated generalized polynomial chaos representation with K basis functions:

$$V_{\rm out}(\vec{\xi}, t) = \sum_{k=1}^{K} v_k(t)\psi_k(\vec{\xi})$$

where $v_k(t)$ is the time-dependent coefficient of the generalized polynomial chaos expansion for the periodic steady-state solution and is actually solved at a set of time points during the entire period $[0, T]$. Next, $v_k(t)$ is expressed by a truncated Fourier series:

$$v_k(t) = \frac{a_k^0}{2} + \sum_{j=1}^{J} \Big(a_k^j \cos(j\omega t) + b_k^j \sin(j\omega t) \Big)$$

with $\omega = \frac{2\pi}{T}$. The coefficients a_k^j and b_k^j

$$a_k^j = \frac{2}{T}\int_0^T v_k(t)\cos(j\omega t)dt, \;\; b_k^j = \frac{2}{T}\int_0^T v_k(t)\sin(j\omega t)dt$$

are computed by a Trapezoidal integration along the time axis. Finally, the parameter-dependent total harmonic distortion is obtained as

$$\hat{x} = \text{THD} = f(\vec{\xi}) = \sqrt{\frac{\sum_{j=2}^{J}\left[\left(a^j(\vec{\xi})\right)^2 + \left(b^j(\vec{\xi})\right)^2\right]}{\left(a^1(\vec{\xi})\right)^2 + \left(b^1(\vec{\xi})\right)^2}} \quad (32)$$
$$\text{with } a^j(\vec{\xi}) = \sum_{k=1}^{K} a_k^j \phi_k(\vec{\xi}), \;\; b^j(\vec{\xi}) = \sum_{k=1}^{K} a_k^j \phi_k(\vec{\xi}).$$

We set $J = 5$ in the Fourier expansion, which is accurate enough for this low-noise amplifier. We use a 3rd-order

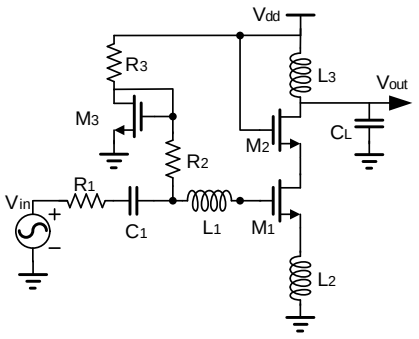


Fig. 8. Schematic of the low-noise amplifier.

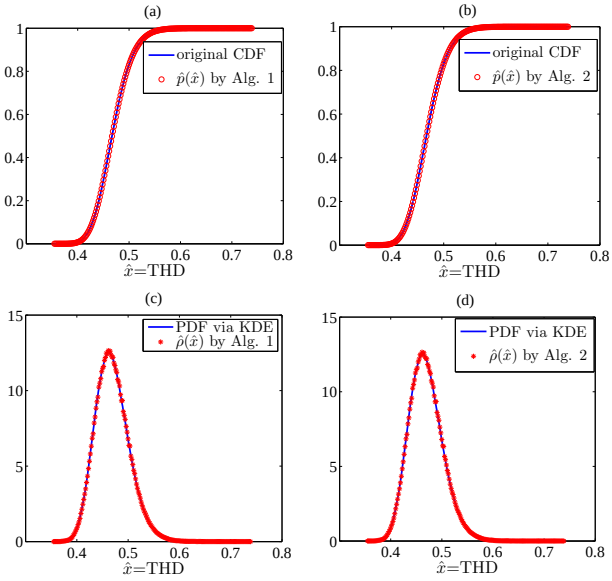


Fig. 9. Cumulative density function (CDF) and probability density function (PDF) for the total harmonic distortion (THD) of the low-noise amplifier. The reference PDF is generated by kernel density estimation (KDE).

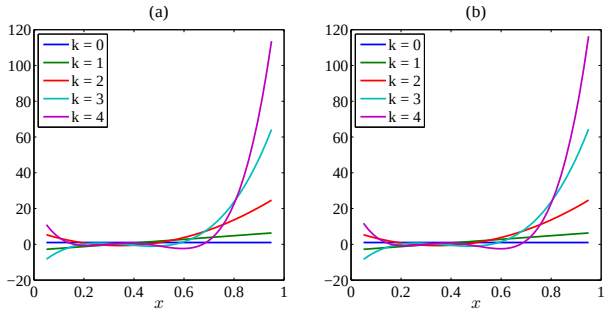


Fig. 10. Computed generalized polynomial-chaos basis functions $\phi_k(x)$ ($k = 0, \cdots, 4$) for the low-noise amplifier. (a) uses the probability density function from Alg. 1, and (b) uses the probability density function from Alg. 2.

TABLE III
COMPUTED GAUSS QUADRATURE POINTS AND WEIGHTS FOR THE LOW-NOISE AMPLIFIER.

with $\rho(x)$ from Alg. 1		with $\rho(x)$ from Alg. 2	
x^k	w^k	x^k	w^k
0.131542	0.056766	0.140381	0.073309
0.251826	0.442773	0.261373	0.470691
0.385311	0.4432588	0.395704	0.400100
0.550101	0.066816	0.561873	0.055096
0.785055	0.001056	0.798122	0.000803

generalized polynomial chaos expansion, leading to K=35. This surrogate model is evaluated by Monte Carlo with 5×10^5 samples at the cost of 330 seconds.

Density Estimation: 114 points are selected from the obtained cumulative density function curve to generate $p(x)$ and $\rho(x)$ by Alg. 1 and Alg. 2, respectively, which costs only several milliseconds. After scaling, Fig. 9 shows the closed-form density functions for the total harmonic distortion of this low-noise amplifier, which matches the results from Monte Carlo simulation very well. The generated $p(x)$ monotonically increases from 0 to 1, and $\rho(x)$ is non-negative. Therefore, the obtained density functions are physically consistent.

Basis Function: Using the obtained density functions, several orthonormal polynomials of x are constructed. Fig. 10 shows the first five basis functions of x. Again, the basis functions resulting from our two density estimation implementations look similar since the density functions from both methods are only slightly different.

Gauss Quadrature Rule: Five Gauss quadrature points and weights are computed and listed in Table III. Again the results from two density estimations are slightly different due to the employment of different density estimators. When the density functions from piecewise cubic and piecewise rational quadratic interpolations are used, the the errors defined in (30) are 3.11×10^{-14} and 4.34×10^{-14}, respectively.

D. Comparison with Asymptotic Probability Extraction

Finally we test our examples by the previous asymptotic probability extraction algorithm [29], [30]. Since our surrogate models are not in linear quadratic forms, we slightly modify asymptotic probability extraction: as done in [18] we use Monte Carlo to compute the statistical moments. All other procedures are exactly the same with those in [29], [30].

As shown in Fig. 11, asymptotic probability extraction produces some negative probability density function values for the synthetic example and the Colpitts oscillator. The probability density functions of the low-noise amplifier are also slightly below 0 in the tail regions, which is not clearly visible in the plots. Compared with the results from our proposed algorithms (that are non-negative and graphically indistinguishable with the original probability density functions), the results from asymptotic probability extraction have larger errors. As suggested by [29], [30], we increase the order of moment matching to 15, hoping to produce non-negative results. Unfortunately, Fig. 11 (d) and (e) show that negative probability density function values still appear, although the accuracy is improved around the peaks. Further increasing the order to 17, we observe that some positive poles are generated by asymptotic waveform evaluation [41]. Such positive poles make the computed probability density functions unbounded and far from the original ones, as demonstrated by Fig. 11 (g) & (h). For the low-noise amplifier, the approximated probability density function curve also becomes unbounded once we increase the order of moment matching to 20, which is not shown in the plot.

These undesirable phenomenon have been explained in Section II-A [c.f. Items 2) and 3)]. Although it is possible to compute the statistical moments in some other ways (e.g., using maximum likelihood [53] or point estimation method [31]), the shortcomings of asymptotic waveform evaluation (i.e., numerical instability and causing negative impulse response for a linear system) cannot be overcome. Because the density functions from asymptotic probability extraction may be physically inconsistent, they cannot be reused in a stochastic simulator (otherwise non-physical results may be obtained). Since the obtained probability density function is not guaranteed non-negative, the computed κ_i in the three-term relation (3) may become negative, whereas (4) implies that κ_i should always be non-negative.

VII. CONCLUSIONS

Motivated by hierarchical uncertainty quantification, this paper has proposed a framework to determine generalized polynomial-chaos basis functions and Gauss quadrature rules from surrogate models. Starting from a general surrogate model, closed-form density functions have been constructed by two monotone interpolation techniques. It has been shown that the obtained density functions are physically consistent: the cumulative density function is monotone and bounded by 0 and 1; the probability density function is guaranteed non-negative. Such properties are not guaranteed by existing moment-matching density estimators. By exploiting the special forms of our obtained probability density functions, generalized polynomial-chaos basis functions and Gauss quadrature rules have been easily determined, which can be used for higher-level stochastic simulation. The effectiveness of our proposed algorithms has been verified by several synthetic and practical circuit examples, showing excellent efficiency (at the cost of milliseconds) and accuracy (with errors around 10^{-14}). The obtained generalized polynomial-chaos basis functions

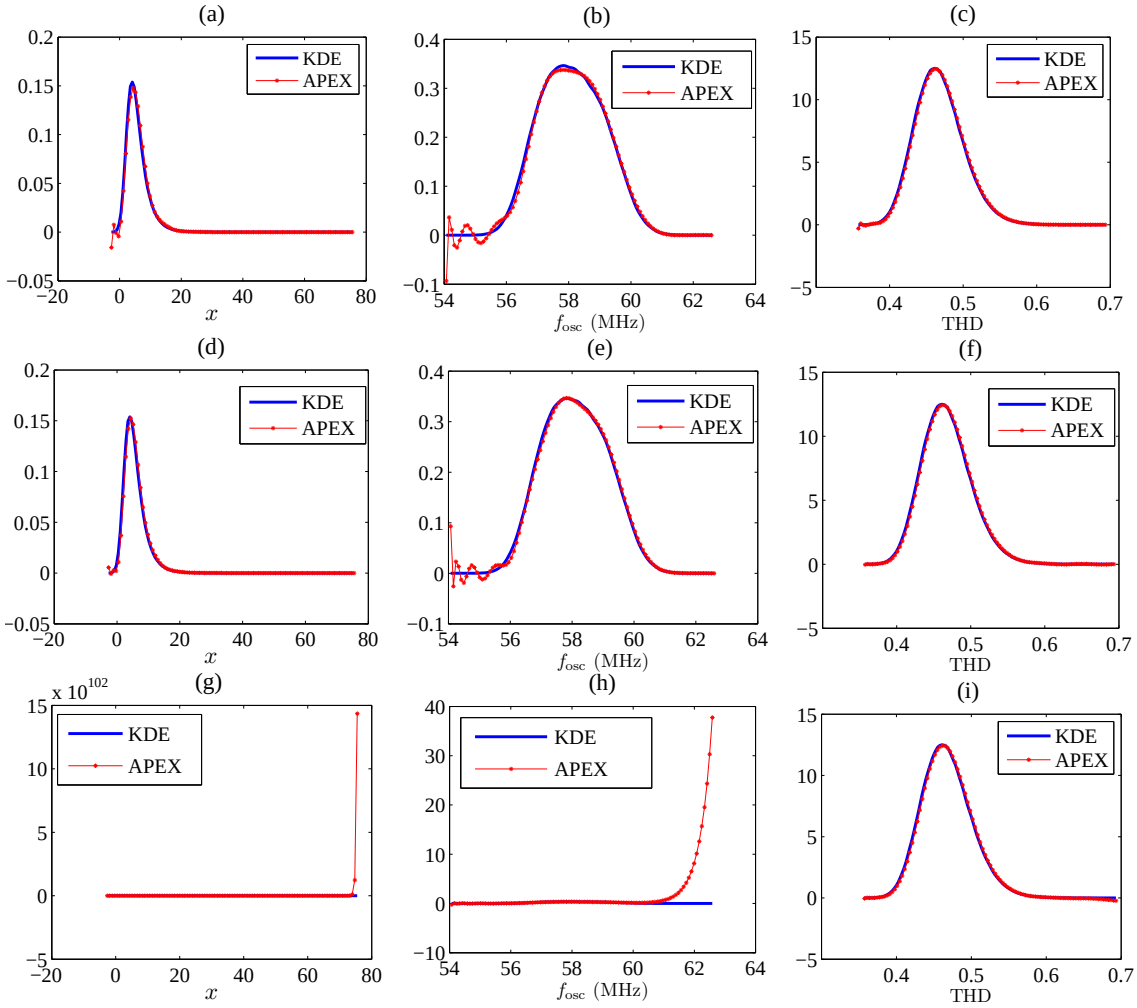


Fig. 11. Probability density functions extracted by asymptotic probability extraction (APEX) [29], [30], compared with the results from kernel density estimation (KDE). Left column: the synthetic example. Central column: frequency of the Colpitts oscillator. Right column: total harmonic distortion (THD) of the low-noise amplifier. (a)-(c): with 10 moments; (d)-(f): with 15 moments; (g)-(i): with 17 moments.

and Gauss quadrature points/weights allow standard stochastic spectral methods to efficiently handle surrogate models in a hierarchical simulator.

REFERENCES

[1] Z. Zhang, T. A. El-Moselhy, I. M. Elfadel, and L. Daniel, "Stochastic testing method for transistor-level uncertainty quantification based on generalized polynomial chaos," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 10, pp. 1533–1545, Oct 2013.

[2] Z. Zhang, T. A. El-Moselhy, P. Maffezzoni, I. M. Elfadel, and L. Daniel, "Efficient uncertainty quantification for the periodic steady state of forced and autonomous circuits," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 60, no. 10, Oct 2013.

[3] Z. Zhang, I. M. Elfadel, and L. Daniel, "Uncertainty quantification for integrated circuits: Stochastic spectral methods," in *Proc. Intl. Conf. Computer-Aided Design*. San Jose, CA, Nov 2013, pp. 803–810.

[4] R. Pulch, "Polynomial chaos for linear differential algebraic equations with random parameters," *Int. J. Uncertainty Quantification*, vol. 1, no. 3, pp. 223–240, 2011.

[5] ——, "Modelling and simulation of autonomous oscillators with random parameters," *Mathematics and Computers in Simulation*, vol. 81, no. 6, pp. 1128–1143, Feb 2011.

[6] A. Singhee and R. A. Rutenbar, "Statistical blockade: Very fast statistical simulation and modeling of rare circuit events and its application to memory design," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 28, no. 8, pp. 1176–1189, Aug. 2009.

[7] ——, "Why Quasi-Monte Carlo is better than Monte Carlo or latin hypercube sampling for statistical circuit analysis," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 29, no. 11, pp. 1763–1776, Nov. 2010.

[8] E. Felt, S. Zanella, C. Guardiani, and A. Sangiovanni-Vincentelli, "Hierarchical statistical characterization of mixed-signal circuits using behavioral modeling," in *Proc. Intl. Conf. Computer-Aided Design*. Washington, DC, Nov 1996, pp. 374–380.

[9] J. Wang, P. Ghanta, and S. Vrudhula, "Stochastic analysis of interconnect performance in the presence of process variations," in *Proc. Design Auto Conf.*, 2004, pp. 880–886.

[10] S. Vrudhula, J. M. Wang, and P. Ghanta, "Hermite polynomial based interconnect analysis in the presence of process variations," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2001–2011, Oct. 2006.

[11] P. Sumant, H. Wu, A. Cangellaris, and N. R. Aluru, "Reduced-order models of finite element approximations of electromagnetic devices exhibiting statistical variability," *IEEE Trans. Antennas Propag.*, vol. 60, no. 1, pp. 301–309, Jan. 2012.

[12] T. Moselhy and L. Daniel, "Variation-aware stochastic extraction with large parameter dimensionality: Review and comparison of state of the art intrusive and non-intrusive techniques," in *Proc. Intl. Symp. Quality Electronic Design*, Mar. 2011, pp. 1–16.

[13] ——, "Stochastic integral equation solver for efficient variation aware interconnect extraction," in *Proc. Design Auto. Conf.*, Jun. 2008, pp. 415–420.

[14] T. El-Moselhy and L. Daniel, "Variation-aware interconnect extraction using statistical moment preserving model order reduction," in *Proc.*

Design Autom. Test in Europe. Dresden, Germany, March 2010, pp. 453–458.

[15] T. A. El-Moselhy, I. M. Elfadel, and D. Widiger, "Efficient algorithm for the computation of on-chip capacitance sensitivities with respect to a large set of parameters," in *Proc. Design Automation Conf.* Anahei, CA, Jun. 2008, pp. 906–911.

[16] H. Chang and S. S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 9, pp. 1467–1482, Sept 2005.

[17] ——, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *Proc. Intl. Conf. Computer-Aided Design*. San Jose, CA, Nov 2003, pp. 621–625.

[18] J. Singh and S. Sapatnekar, "Statistical timing analysis with correlated non-gaussian parameters using independent component analysis," in *Proc. Design Autom. Conf.* San Francisco, CA, Jun 2006, pp. 155–160.

[19] O. Le Maitre and O. Knio, *Spectral methods for uncertainty quantification: with application to computational fluid dynamics*. Springer, 2010.

[20] D. Xiu and J. S. Hesthaven, "High-order collocation methods for differential equations with random inputs," *SIAM J. Sci. Comp.*, vol. 27, no. 3, pp. 1118–1139, Mar 2005.

[21] I. Babuška, F. Nobile, and R. Tempone, "A stochastic collocation method for elliptic partial differential equations with random input data," *SIAM J. Numer. Anal.*, vol. 45, no. 3, pp. 1005–1034, Mar 2007.

[22] F. Nobile, R. Tempone, and C. G. Webster, "A sparse grid stochastic collocation method for partial differential equations with random input data," *SIAM J. Numer. Anal.*, vol. 46, no. 5, pp. 2309–2345, May 2008.

[23] ——, "An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data," *SIAM J. Numer. Anal.*, vol. 46, no. 5, pp. 2411–2442, May 2008.

[24] R. Ghanem and P. Spanos, *Stochastic finite elements: a spectral approach*. Springer-Verlag, 1991.

[25] D. Xiu and G. E. Karniadakis, "The Wiener-Askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, Feb 2002.

[26] ——, "Modeling uncertainty in flow simulations via generalized polynomial chaos," *Journal of Computational Physics*, vol. 187, no. 1, pp. 137–167, May 2003.

[27] D. Xiu, "Fast numerical methods for stochastic computations: A review," *Communications in Computational Physics*, vol. 5, no. 2-4, pp. 242–272, Feb. 2009.

[28] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, D. K. Beece, J. Piaget, N. Venkateswaran, and J. G. Hemmett, "First-order incremental block-based statistical timing analysis," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2170 – 2180, Oct 2006,.

[29] X. Li, J. Le, P. Gopalakrishnan, and L. T. Pileggi, "Asymptotic probability extraction for non-normal distributions of circuit performance," in *Proc. Int. Conf. Computer-Aided Design*, Nov 2004, pp. 2–9.

[30] ——, "Asymptotic probability extraction for nonnormal performance distributions," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 1, pp. 16–37, Jan. 2007.

[31] F. Gong, H. Yu, and L. He, "Stochastic analog circuit behavior modeling by point estimation method," in *Proc. Intl. Symp. Physical Design*. Santa Barbara, CA, March 2011, pp. 175–182.

[32] C. Michael and M. Ismail, "Statistical modeling of device mismatch for analog MOS integrated circuits," *IEEE Journal Solid-State Circuits*, vol. 27, no. 2, pp. 154–166, Feb. 1992.

[33] M. Frangos, Y. Marzouk, K. Willcox, and B. van Bloemen Waanders, "Surrogate and reduced-order modeling: A comparison of approaches for large-scale statistical inverse problems," *Large-Scale Inverse Problems and Quantification of Uncertainty*, pp. 123–149, 2010.

[34] F. N. Fritsch and R. E. Carlson, "Monotone piecewise cubic interpolation," *SIAM J. Numerical Analysis*, vol. 17, no. 2, pp. 238 –246, April 1980.

[35] J. M. Hyman, "Accurate monotonicity preserving cubic interpolation," *SIAM J. Sci. Stat. Comput.*, vol. 4, no. 4, pp. 645 –654, Dec. 1983.

[36] R. Delbourgo and J. A. Gregory, "Shape preserving piecewise rational interpolation," *SIAM J. Sci. Stat. Comput.*, vol. 6, no. 4, pp. 967 –976, Oct. 1985.

[37] J. A. Gregory and R. Delbourgo, "Piecewise rational quadratic interpolation to monotonic data," *IMA J. Numer. Anal.*, vol. 2, no. 2, pp. 123 – 130, 1982.

[38] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, Sept 1956.

[39] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, Sept 1962.

[40] J.-N. Hwang, S.-R. Lay, and A. Lippman, "Nonparametric multivariate density estimation: a comparative study," *IEEE Trans. Signal Processing*, vol. 42, no. 10, pp. 2795–2810, Oct 1994.

[41] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 4, pp. 352 –366, April 1990.

[42] R. Krishnan, W. Wu, F. Gong, and L. He, "Stochastic behavioral modeling of analog/mixed-signal circuits by maximizing entropy," in *Proc. Intl. Symp. Qual. Electr. Design*. Santa Clara, CA, Mar 2013, pp. 572–579.

[43] G. H. Golub and J. H. Welsch, "Calculation of gauss quadrature rules," *Math. Comp.*, vol. 23, pp. 221–230, 1969.

[44] W. Gautschi, "On generating orthogonal polynomials," *SIAM J. Sci. Stat. Comput.*, vol. 3, no. 3, pp. 289–317, Sept. 1982.

[45] X. Li, "Finding deterministic solution from underdetermined equation: large-scale performance modeling of analog/RF circuits," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 29, no. 11, pp. 1661–1668, Nov 2011.

[46] X. Yang and G. E. Karniadakis, "Reweighted l_1 minimization method for sothcastic elliptic differential equations," *J. Comp. Phys.*, vol. 248, no. 1, pp. 87–108, Sept. 2013.

[47] L. Yan, L. Guo, and D. Xiu, "Stochastic collocation algorithms using l_1 minimization," *Int. J. Uncert. Quant.*, vol. 2, no. 3, pp. 279–293, Nov. 2012.

[48] I. H. Slaon and H. Wozniakowski, "When are quasi-Monte Carlo algorithms efficient for high-dimensional integrals?" *Journal of Complexity*, vol. 14, no. 1, pp. 1–33, Mar 1998.

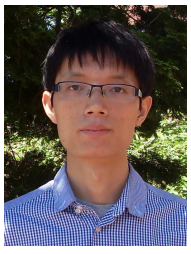
[49] C. Soize and R. Ghanem, "Physical systems with random uncertainties: Chaos representations with arbitrary probability measure," *SIAM Journal on Scientific Computing*, vol. 26, no. 2, pp. 395–410, Feb 2004.

[50] Z. Zhang, I. M. Elfadel, and L. Daniel, "Hierachical uncertainty quantification: Practical algorithms and numerical validation," in preparation.

[51] D. L. Donoho, "Compressed sensing," *IEEE Trans. Informa. Theory*, vol. 52, no. 4, pp. 578 –594, April 2006.

[52] J. C. Wheeler, "Modified moments and gaussian quadrature," *Rochy Muntain J. Math.*, vol. 4, no. 2, pp. 287–296, Spring 1974.

[53] C. Gu, E. Chiprout, and X. Li, "Efficient moment estimation with extremely small sample size via bayesian inference for analog/mixed-signal validation," in *Proc. Design Autom. Conf.* Austin, TX, Jun 2013, pp. 1–7.



Zheng Zhang (S'09) received his B.Eng. degree from Huazhong University of Science and Technology, China, in 2008, and his M.Phil. degree from the University of Hong Kong, Hong Kong, in 2010. He is a Ph.D student in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology (MIT), Cambridge, MA. His research interests include numerical methods for uncertainty quantification, computer-aided design (CAD) of integrated circuits and microelectromechanical systems (MEMS), and model order reduction.

In 2009, Mr. Zhang was a visiting scholar with the University of California, San Diego (UCSD), La Jolla, CA. In 2011, he collaborated with Coventor Inc., working on CAD tools for MEMS design. In the summer of 2013, he was a visiting scholar at the Applied Math Division of Brown University, Providence, RI. He was recipient of the Li Ka Shing Prize (university best M.Phil/Ph.D thesis award) from the University of Hong Kong, in 2011, and the Mathworks Fellowship from MIT, in 2010.

Tarek El-Moselhy received the B.Sc. degree in electrical engineering in 2000 and a diploma in mathematics in 2002, then the M.Sc. degree in mathematical engineering, in 2005, all from Cairo University, Cairo, Egypt. He received the Ph.D. degree in electrical engineering from Massachusetts Institute of Technology, Cambridge, in 2010.

From 2010 to 2013, he was a postdoctoral associate in the Department of Aeronautics and Astronautics at Massachusetts Institute of Technology (MIT). Currently he is a Quantitative Analyst at The D. E. Shaw Group, New York, NY. His research interests include fast algorithms for deterministic and stochastic electromagnetic simulations, stochastic algorithms for uncertainty quantification in high dimensional systems, and stochastic inverse problems with emphasis on Bayesian inference.

Dr. El-Moselhy received the Jin Au Kong Award for Outstanding PhD Thesis in Electrical Engineering from MIT in 2011, and the IBM Ph.D Fellowship in 2008.

Ibrahim (Abe) M. Elfadel (SM'-02) received his Ph.D. from Massachusetts Institute of Technology (MIT) in 1993 and is currently Professor and Head of the Center for Microsystems at the Masdar Institute of Science and Technology, Abu Dhabi, UAE.

Dr. Elfadel is the Director of the TwinLab/Abu Dhabi Center for 3D IC Design, a joint R & D program with the Technical University of Dresden, Germany. He is also the co-director of the ATIC-SRC Center of Excellence for Energy-Efficient Electronic Systems (ACE4S). Dr. Elfadel has 15 years of industrial experience with IBM in the research, development and deployment of advanced computer-aided design (CAD) tools and methodologies for deep-submicron, high-performance digital designs. His group's research is concerned with various aspects of energy-efficient digital system design and includes CAD for variation-aware, low-power nano-electronics, power and thermal management of multicore processors, embedded DSP for mmWave wireless systems, modeling and simulation of micro power sources, and 3D integration for energy-efficient VLSI design.

Dr. Elfadel is the recipient of six Invention Achievement Awards, an Outstanding Technical Achievement Award and a Research Division Award, all from IBM, for his contributions in the area of VLSI CAD. He is currently serving as an Associate Editor for the IEEE Transactions on Computer-Aided Design for Integrated Circuits and Systems and the IEEE Transactions on Very-Large-Scale Integration.

Luca Daniel (S'98-M'03) received the Laurea degree (*summa cum laude*) in electronic engineering from the Universita di Padova, Italy, in 1996, and the Ph.D. degree in electrical engineering from the University of California, Berkeley, in 2003.

He is an Associate Professor in the Electrical Engineering and Computer Science Department of the Massachusetts Institute of Technology (MIT), Cambridge. His research interests include accelerated integral equation solvers and parameterized stable compact dynamical modeling of linear and nonlinear dynamical systems with applications in mixed-signal/RF/mm-wave circuits, power electronics, MEMs, and the human cardiovascular system.

Dr. Daniel received the 1999 IEEE TRANSACTIONS ON POWER ELECTRONICS best paper award, the 2003 ACM Outstanding Ph.D. Dissertation Award in Electronic Design Automation, five best paper awards in international conferences, the 2009 IBM Corporation Faculty Award, and 2010 Early Career Award from the IEEE Council on Electronic Design Automation.