Dynamic Range Estimation for Nonlinear Systems[*]

Bin Wu Jianwen Zhu Farid N. Najm

Department of Electrical and Computer Engineering
University of Toronto, Toronto, Ontario, Canada
{bwu, jzhu, najm}@eecg.toronto.edu

ABSTRACT

It has been widely recognized that the dynamic range information of an application can be exploited to reduce the datapath bitwidth of either processors or ASICs, and therefore the overall circuit area, delay and power consumption. While recent advances in analytical dynamic range estimation can deliver results accurate enough to account for both spatial and temporal correlation, the reported methods are only valid for linear systems. In this paper, we use a powerful mathematical tool, called polynomial chaos, which enables not only the orthogonal decomposition of random processes, but also the propagation of random processes through nonlinear systems with difficult constructs such as multiplications, divisions and conditionals. We show that when applied to interesting nonlinear applications such as adaptive filters, polynomial filters and rational filters, this method can produce complete, accurate statistics of each internal variable, thereby allowing the synthesis of bitwidth with the desired tradeoff between circuit performance and signal-to-noise ratio.

1. INTRODUCTION

Today's ASIC designers start with a design specification handed off by system designers. Often in the form of C code, the algorithm-level design specification needs to be converted into register transfer level (RTL) design, typically in the form of hardware description languages. A crucial decision to be made during this process is the datapath bitwidth, including the bitwidths of different registers and functional units. An aggressively designed datapath often replaces floating-point arithmetic contained in the design specification by their fixed-point counterparts. In addition, the redundant bits that do not contribute much to the accuracy of the application are often eliminated. Such datapaths with minimal bitwidth always translate to superior *circuit performance* in terms of area, speed and power consumption. To make this possible, the dynamic range information of the application (*i.e.*, the range of values taken by its variables), and in the case of C code, the dynamic ranges of all declared variables and intermediate expressions (all referred to as signals or variables in the following text), have to be obtained.

Unfortunately, the best practice today for dynamic range estimation is still profiling (also referred to as simulation), which works by instrumenting the original application with code that can trace the value ranges at runtime. While this method can be made very accurate, the accuracy is achieved only by extremely long simulation, and worst of all, no confidence on the accuracy can be obtained. In contrast, analytical methods can avoid long simulation by analyzing the application at compile time. While many advances have been made on this front, the proposed methods have not been able to provide dynamic range information as *accurate* and as *complete* as profiling. The accuracy problem can usually be attributed to either no treatment of signal correlation, as in the cases of bitwidth [1, 2] or moment propagation method [3], or inadequate treatment of signal correlation, as in the case of affine arithmetic method [4]. The completeness problem relates to the fact that these methods typically produce only value or error bounds instead of signal distribution, thereby limiting the scope of their application.

A new method has recently been proposed where full statistics of all signals can be produced analytically and accurately [5]. The key idea of this method is to decompose the system input, modeled as an arbitrary random process, into K number of deterministic signals, each multiplied by a random variable. One can then obtain the system response of a random input by combining system responses of K deterministic inputs, each multiplied by the corresponding random variable. While this method can capture spatial and temporal correlation effectively, it relies on the application of superposition, a property enjoyed only by linear systems.

By definition, a system $y = f(x)$ is *nonlinear* if it does not satisfy the superposition property: $f(a_1x_1 + a_2x_2) = a_1f(x_1) + a_2f(x_2)$. In practice, whenever operations such as multiplication, division, and conditionals are used in the corresponding C behavioral description, a system is likely to be nonlinear. Therefore, the majority of C applications are nonlinear. Even in the domain of digital signal processing, where linear filters are widely used, nonlinear systems comprise a large, important category of applications. For example, in image processing systems, it is typical to have applications that use logarithmic nonlinearity to model the human eye. Similarly, *neural networks* apply nonlinear operations to linear combinations of inputs to mimic nervous system. Other common examples include *median filters*, where the output is the median of the input values in the corresponding neighborhood, *polynomial filters*, where the output is a polynomial function of its inputs, *rational filters*, where the output is a ratio of two input polynomials, and *adaptive filters*, where the output is a linear combination of inputs except that the coefficient is time-variant. The readers are referred to books such as [6] for a detailed, comprehensive treatment.

In this paper, we solve the difficult problem of dynamic range estimation for nonlinear systems. The key idea of our solution is the use of a powerful statistical construct, called *polynomial chaos*, which is a family of polynomials constructed from a set of random variables. With a rigorous, well-defined procedure detailed in Section 4.2, these polynomials are constructed with the powerful properties such that when an input signal is decomposed into a combination of polynomial chaos, called *polynomial chaos expansion*, it is easily propagated through a nonlinear system, and the full statistics of all signals can be derived accordingly. Intu-

[*] This research was supported by Micronet, with funding from ATI Technologies Inc. and from Altera Corp.

itively, the accuracy of this method can be attributed to the fact that the random variable set can effectively capture the correlation between signals, and the use of polynomials can effectively preserve the correlation even after nonlinear operations.

We demonstrate the following advantages over previous approaches: First, it is extremely *fast* compared to profiling, the only method with comparable accuracy. While profiling has to simulate the system for thousands or even millions of random input realizations, our method only needs a single run of a program transformed from the original system specification. Second, our method is *constructive* in the sense that the distribution of all signals can be obtained. As a result, an optimal bitwidth can be synthesized by aggressively cutting the tails of the distribution under the given signal-to-noise ratio constraint. In contrast, other recent analytical approaches, such as [4], can offer only a verification of error bound of a given bitwidth selection, while shedding no insight on the proper choice of bitwidth. Third, our method is mathematically *rigorous*. In fact, we introduce only one engineering assumption, which is later justified experimentally. Fourth, our method is highly *accurate*. Our method can capture not only the spatial correlation, but also the temporal correlation in the system input, which is not possible in the previous methods, but as shown in [5], contributes significantly to the accuracy of the result. And finally, it is the first time that the *complete* result on nonlinear systems are given. In contrast, while the propagation of different signal representations through nonlinear operations are discussed in [3, 4], no results on a complete nonlinear system has been demonstrated.

A word about notation would be useful. We will use bold font, such as $\mathbf{x}$, to denote a random variable (RV). Deterministic (i.e., non-random) variables will be denoted with regular italic font, such as x. In addition, some Greek symbols, such as ξ_i, μ_i and Ψ_i will be used to denote RVs, but sometimes Greek symbols like λ_i will denote deterministic variables. It will be clear from the context which variables are random and which are not. The notation $E[\mathbf{x}]$ will denote the *mean* or *expected value* of the RV $\mathbf{x}$. Two RVs $\mathbf{x}$ and $\mathbf{y}$ are said to be *orthogonal* if $E[\mathbf{xy}] = 0$. Two zero-mean RVs $\mathbf{x}$ and $\mathbf{y}$ are said to be *orthonormal* if they are orthogonal and each has a variance of 1.

The remainder of this paper is organized as follows: Section 2 gives a brief account of the related literature. Section 3 gives the theoretical background of polynomial chaos. Section 4 describes our proposed method in detail. We give experimental result in Section 5 before drawing conclusion in Section 6.

2. RELATED WORK

Profiling or simulation-based approaches [7, 8, 9, 10] have been used extensively to estimate the dynamic range of variables in a program. As mentioned earlier, this method is computationally expensive, since huge amount of sample data need to be simulated.

The L_p norm method, based on the use of the transfer function, is proposed in [11, 12]. This method requires explicit knowledge of the system transfer function, which may not always be available, and which may be difficult to extract from C code. In addition, it propagates only the maximum value of the data and is limited to linear systems.

A moment-based method is presented in [3]. This method models the input as a random variable and propagates its moments through the system. The probability density function (pdf) of all variables can be constructed from the propagated moments. However, this method assumes that the input data is temporally uncorrelated, and that variables internal to the system (*e.g.* at the input to an arithmetic operator) are spatially uncorrelated. Both these assumptions are not true in practice, and can have significant impact on accuracy of the results, as we will demonstrate in Section 5.4.

A bitwidth/interval propagation method is described in [1, 2]. This method propagates the input bitwidth or interval through the whole system to obtain the dynamic range for intermediate variables. The estimated result from this method is overly pessimistic, since it always considers the worst case.

Affine arithmetic, an improvement on the interval propagation technique, is given in [4], where the spatial correlation between internal variables induced by the reconvergent fanout structure in the system is taken into account. However, this method does not capture the inherent temporal correlation of the input signals. For many applications, spatial correlation between variables is a direct result of the temporal correlation of the input signals. Therefore, ignoring the temporal correlation of the input automatically causes loss of true correlation information internally. In addition, because the information propagated is limited to affine, or linear form, accuracy has to be compromised when nonlinear terms are generated after nonlinear operations. As detailed later in Section 5.4, the loss of accuracy can be significant.

Thus, all existing methods have certain shortcomings, especially in the way that correlation and nonlinearity are dealt with. In our work, we take care of both temporal input correlation and spatial internal correlation even for nonlinear systems, and we provide, not only ranges of the data, but its statistical distribution as well.

3. POLYNOMIAL CHAOS

This section provides a brief review of the theory of polynomial chaos, and describes the polynomial chaos expansion of a random variable (RV), based mainly on [13].

Let $\xi_1, \xi_2, \ldots, \xi_n$ be a sequence of zero-mean *orthonormal* Gaussian random variables (RVs). In other words, each ξ_i is Gaussian with mean 0 and variance 1 (i.e., each has the standard normal distribution), and they are *orthogonal*, so that:

$$E[\xi_i \xi_j] = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where $E[\cdot]$ is the expected value (mean) operator [14]. It can be shown that the ξ_i are also *independent*. Let $\Gamma_p(\xi_1, \xi_2, \ldots, \xi_n)$ be a polynomial whose degree is p, defined on the orthonormal Gaussian RVs $\xi_1, \xi_2, \ldots, \xi_n$. If we consider different permutations and replacements of the arguments of $\Gamma_p(\cdot)$, which has at most p distinct arguments, we represent the resulting *set of polynomials* by:

$$\{\Gamma_p(\xi_{i_1}, \xi_{i_2}, \ldots, \xi_{i_p})\} \tag{2}$$

where $i_k \in \{1, 2, \ldots, n\}$ for all $k \in \{1, 2, \ldots, p\}$. This set of polynomials is referred to as *polynomial chaos* of order p. To be exact, polynomial chaos is not just any set of polynomials; rather, it is a set with an important orthogonality property, defined by means of the following construction.

The zeroth order polynomial chaos (PC), $\Gamma_0(\cdot)$ is a constant, and is chosen to be 1 by construction:

$$\Gamma_0() = 1 \tag{3}$$

The first order PC is chosen so that every polynomial in $\Gamma_1(\xi_i)$ is orthogonal to every polynomial in $\Gamma_0(\cdot)$, where two distinct polynomials $\Gamma_q(\xi_{i_1}, \ldots, \xi_{i_q})$ and $\Gamma_r(\xi_{j_1}, \ldots, \xi_{j_r})$ are said to be orthogonal if:

$$E\left[\Gamma_q(\xi_{i_1}, \ldots, \xi_{i_q}) \Gamma_r(\xi_{j_1}, \ldots, \xi_{j_r})\right] = 0 \tag{4}$$

Applying this orthogonality requirement leads to the following choice for the first order PC:

$$\Gamma_1(\xi_i) = \xi_i \tag{5}$$

The second order PC is chosen so that it is orthogonal to both Γ_0 and Γ_1, leading to:

$$\Gamma_2(\xi_{i_1}, \xi_{i_2}) = \xi_{i_1}\xi_{i_2} - \delta_{i_1 i_2} \tag{6}$$

where δ_{ij} is the Kronecker delta:

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Similar procedures give all higher order PCs, so that each is orthogonal to all lower order PCs. What is not obvious from the brief and informal presentation above is that even within a single $\{\Gamma_p\}$, it can be shown that the polynomials are all orthogonal. The concept of polynomial chaos is useful in that it provides a basis for decomposition of a general random variable, called a *polynomial chaos expansion* (PCE), as follows.

If $\mathbf{x}$ is a square integrable RV, that is, $E\left[|\mathbf{x}|^2\right]$ is finite, then it can be shown that one can express it as a decomposition or expansion in terms of an underlying basis of polynomial chaos. The number n of underlying RVs $\xi_1, \ldots, \xi_n$ is referred to as the *dimension* of the expansion, and, as above, the highest degree in the polynomials, p, is called the *order*. The exact PCE generally uses infinite dimension (n) and order (p). However, in practice, one can truncate the infinite series expansion based on a finite n and p, by neglecting higher order terms. Thus the two dimensional ($n = 2$) PCE with order two ($p = 2$) is:

$$\begin{aligned}\mathbf{x} = a_0\Gamma_0 + a_1\Gamma_1(\xi_1) + a_2\Gamma_1(\xi_2) + a_{11}\Gamma_2(\xi_1, \xi_1)\\ +a_{12}\Gamma_2(\xi_1, \xi_2) + a_{22}\Gamma_2(\xi_2, \xi_2) \qquad (8)\end{aligned}$$

where the a_{ij} are constant coefficients. $\Gamma_2(\xi_2, \xi_1)$ is dropped here since it is identical to $\Gamma_2(\xi_1, \xi_2)$. In order to simplify the notation, the polynomials are sorted in a specific way and they are denoted $\Psi_0, \Psi_1, \Psi_2, \ldots, \Psi_m$; this is nothing more than a different numbering scheme. Given an order (p) and a dimension (n), one can quickly determine [13] exactly which polynomial each Ψ_i corresponds to, under this numbering scheme. For example, table 1 shows an example of the numbered polynomials for the 2-dimensional case. With this simplified notation, each RV can be expressed as the following expansion:

$$\mathbf{x} = \sum_{i=0}^{m} x_i\Psi_i \qquad (9)$$

In general, m grows very quickly for larger n and p, but a PCE is typically useful even for small order and dimension, so that m remains tractable in practice. It can be proved that the number of n dimension PCs up to pth order, $m + 1$, is

$$m + 1 = \sum_{i=0}^{p} \frac{(n + i - 1)!}{i!(n - 1)!} = \frac{(n + p)!}{n!p!} \qquad (10)$$

The key property of these polynomials that turns out to be very useful in practice is the following orthogonality property:

$$E\left[\Psi_i\Psi_j\right] = \begin{cases} E\left[\Psi_i^2\right], & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases} \qquad (11)$$

The values of $E[\Psi_i^2]$ can be computed easily based on the fact, easily proven, that if ξ is a zero-mean unity-variance Gaussian, then for any integer $k \geq 0$:

$$E\left[\xi^{2k+1}\right] = 0 \quad \text{and} \quad E\left[\xi^{2k}\right] = \frac{(2k)!}{2^k k!} \qquad (12)$$

With the above, it becomes possible to apply PCE to complex nonlinear differential systems of equations, such as in computational fluid dynamics [15], where random parameter perturbations have typically been expressed using PCE. As a result, an analysis of a system under random stimulus is replaced by repeated deterministic analyses of the system, from which coefficients of the PCE are solved for separately. Once the PCE coefficients are known, the distribution, the moments, or any statistics of the system response can be solved for.

4. PROPOSED SOLUTION

The behavior of digital signal processing (DSP), as well as the behavior of general digital applications, can be captured at a high level of abstraction with a data-flow graph (DFG). Each node of the DFG represents a primitive operation, typically a multiplication or addition of real numbers, or a decision operation. Note that with a decision operation, the if-then-else branch that is typically captured in a control-flow graph (CFG) is implicitly captured in the DFG. The construction procedure for such a DFG is standard and has been reported in both the compiler [16] and CAD [17] community.

With the primary inputs of the DFG viewed as random, we propose to apply a PCE-based analysis to each node of the DFG in order to derive statistics of the various system signals and outputs under random inputs. The "source of randomness" in the system primary input is either a genuine uncertainty about what inputs to expect, or a representation of a large population of possible inputs by means of their aggregate statistics. In any case, we will show that the PCE formulation allows us to extend the analysis of such systems for dynamic range estimation beyond what was recently done for linear systems in [5], so as to cover systems that allow non-linear operations (multiplication, division) and if-then-else branches. The process will involve propagating the PCE coefficients through the DFG, from the system primary input.

Table 1: Polynomial chaos polynomials $\Psi_0, \ldots, \Psi_{14}$ for $n = 2$ and $p = 0, 1, 2, 3, 4$, with $E[\Psi_i^2]$ in each case.

Index	Order	Polynomial	$E[\Psi_i^2]$
0	0	1	1
1	1	ξ_1	1
2		ξ_2	1
3	2	$\xi_1^2 - 1$	2
4		$\xi_1\xi_2$	1
5		$\xi_2^2 - 1$	2
6	3	$\xi_1^3 - 3\xi_1$	6
7		$\xi_1^2\xi_2 - \xi_2$	2
8		$\xi_1\xi_2^2 - \xi_1$	2
9		$\xi_2^3 - 3\xi_2$	6
10	4	$\xi_1^4 - 6\xi_1^2 + 3$	24
11		$\xi_1^3\xi_2 - 3\xi_1\xi_2$	6
12		$\xi_1^2\xi_2^2 + \xi_1^2 - \xi_2^2 + 1$	4
13		$\xi_1\xi_2^3 - 3\xi_1\xi_2$	6
14		$\xi_2^4 - 6\xi_2^2 + 3$	24

4.1 PCE Propagation

In the following, we will show how the output PCE can be derived from the input PCE, for various types of operations that one typically encounters in DFGs.

4.1.1 Scaling

This case is trivial. If $\mathbf{y} = a\mathbf{x}$ where a is a constant, then if a PCE is available for $\mathbf{x}$, so that $\mathbf{x} = \sum_{i=0}^{m} x_i\Psi_i$, then the PCE for $\mathbf{y}$ is $\mathbf{y} = \sum_{i=0}^{m} y_i\Psi_i$, where:

$$y_i = ax_i \qquad (13)$$

4.1.2 Summation

Here too, the linearity of the summation operation makes this case very easy. If $\mathbf{z} = \mathbf{x} + \mathbf{y}$, and if a PCE expansion is considered for all three variables:

$$\mathbf{x} = \sum_{i=0}^{m} x_i\Psi_i, \quad \mathbf{y} = \sum_{i=0}^{m} y_i\Psi_i, \quad \mathbf{z} = \sum_{i=0}^{m} z_i\Psi_i \qquad (14)$$

then it is clear that:

$$z_i = x_i + y_i \qquad (15)$$

4.1.3 Multiplication

The multiplication case is non-trivial, and it brings out the power of the PCE expansion. Suppose $\mathbf{z} = \mathbf{xy}$ and that a PCE expansion is considered for all three variables, so that:

$$\begin{aligned}\sum_i z_i\Psi_i &= \left(\sum_i x_i\Psi_i\right)\left(\sum_j y_j\Psi_j\right)\\ &= \sum_{i,j} x_i y_j \Psi_i\Psi_j \qquad (16)\end{aligned}$$

In order to solve for z_k, we multiply both sides of the above by Ψ_k and take the expected value of both sides. Based on the orthogonality property (11), this leads to:

$$z_k = \frac{1}{E[\Psi_k^2]} \sum_{i,j} x_i y_j E[\Psi_i\Psi_j\Psi_k] \qquad (17)$$

The 3-way expectations on the right-hand-side are actually trivial to compute for a given dimension and order of PCE, because of independence of the ξ_i variables and due to (12). As an example, in the case when the dimension is one ($n = 1$) and the order is two ($p = 2$), in which case the expansion consists of Ψ_0, Ψ_1, and Ψ_2 only, this type of analysis leads to:

$$\begin{aligned} z_0 &= x_0y_0 + x_1y_1 + 2x_2y_2 \\ z_1 &= x_0y_1 + x_1y_0 + 2x_1y_2 + 2x_2y_1 \qquad (18) \\ z_2 &= x_0y_2 + x_2y_0 + x_1y_1 + 4x_2y_2 \end{aligned}$$

Notice that there is no assumption of independence between $\mathbf{x}$ and $\mathbf{y}$. Indeed, the PC expansion offers a way by which correlations among different RVs can be captured via the coefficients of the various polynomials, so that correlation is taken into account quite naturally. For larger values of n and p, the expressions for z_i become more involved but they remain of this nature, meaning they consist of cross-products of the x_i and y_j terms. For example, in the case when $n = 4$ and $p = 3$, these expressions can grow to include about 35 terms.

4.1.4 Division

Division being the inverse of multiplication, it is easy to see that equations such as (18) can be used in the reverse direction, to give values of x_i, given values of z_i and y_i for example. In the reverse direction, these equations constitute a linear system of simultaneous equations $Ax = b$ whose solution techniques are standard. For example, in the case $n = 1$, $p = 2$, shown above, the resulting matrix equation is:

$$\begin{bmatrix} y_0 & y_1 & 2y_2 \\ y_1 & y_0 + 2y_2 & 2y_1 \\ y_2 & y_1 & y_0 + 4y_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} \tag{19}$$

For higher dimension and order, the equations remain linear, but the size of the matrix and the values of the matrix entries would change.

4.1.5 Multiplexing

We consider the case where the value of a variable depends on a "switch" or multiplexing decision based on another variable. Specifically, if $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{x}$ are RVs, and if (c_0, c_1) is a partition of the domain of $\mathbf{c}$, then we consider the following operation:

$$\mathbf{x} = \begin{cases} \mathbf{a}, & \text{if } \mathbf{c} \in c_0; \\ \mathbf{b}, & \text{if } \mathbf{c} \in c_1. \end{cases} \tag{20}$$

As before, we consider that each variable has a polynomial chaos expansion, so that:

$$\sum_i x_i \Psi_i = \begin{cases} \sum_i a_i \Psi_i, & \text{if } \mathbf{c} \in c_0; \\ \sum_i b_i \Psi_i, & \text{if } \mathbf{c} \in c_1. \end{cases} \tag{21}$$

If we define a new set of RVs $\mathbf{x_i}$ by:

$$\mathbf{x_i} = \begin{cases} a_i, & \text{if } \mathbf{c} \in c_0; \\ b_i, & \text{if } \mathbf{c} \in c_1. \end{cases} \tag{22}$$

then we can write:

$$\sum_i x_i \Psi_i = \sum_i \mathbf{x_i} \Psi_i \tag{23}$$

If we multiply both sides by Ψ_k and take the expected value of both sides, and applying the orthogonality property, leads to:

$$x_k = \frac{1}{E[\Psi_k^2]} \sum_i E\left[\mathbf{x_i} \Psi_i \Psi_k\right] \tag{24}$$

We can further simplify this result by using conditional expectations:

$$\begin{aligned} E\left[\mathbf{x_i} \Psi_i \Psi_k\right] &= a_i E\left[\Psi_i \Psi_k | c \in c_0\right] \mathcal{P}\{c \in c_0\} \\ &+ b_i E\left[\Psi_i \Psi_k | c \in c_1\right] \mathcal{P}\{c \in c_1\} \end{aligned} \tag{25}$$

The conditional expectations can be easily and quickly estimated by Monte Carlo techniques. Random samples of $\xi_1, \ldots, \xi_n$ are generated and classified as to whether they produce a $\mathbf{c}$ in c_0 or c_1. The probabilities of the two outcomes are thus computed, and within each classification, the mean value of $\Psi_i \Psi_k$ is computed in usual Monte Carlo fashion. Since the underlying variables are Gaussians, convergence is easily achieved.

4.1.6 Time shift

The preceding operations were operations on RVs, without regard to the time dimension. In general, a behavioral description consists of a sequencing of the operations in time, so that the RVs considered are really signal values at specific time points on a discrete time scale: $\mathbf{x}[1], \mathbf{x}[2], \ldots$. The preceding analysis for scaling, summation, multiplication, division, and multiplexing were all relevant to specific time point. In addition, it is trivial to handle a time shift operation: if

$$\mathbf{y}[j] = \mathbf{x}[j-k] \tag{26}$$

then the resulting PCE coefficients are simply time-shifted themselves:

$$y_i[j] = x_i[j-k] \tag{27}$$

4.2 Generating the input PCE

In order to propagate the PCE through a DFG, based on the above operations, one needs a PCE expression for the system input to begin with. We assume that the system has a single input data channel, represented with a stochastic process $\mathbf{p}[k], k = 0, 1, \ldots, N$. The extension to multiple inputs is not hard to do, but we will focus on the single input case for clarity. At each time index k, the value $\mathbf{p}[k]$ is an RV. We will show how a Karhunen-Loéve expansion [14] (KLE, or KL expansion) for the stochastic process $\mathbf{p}[k]$ can be used to generate an input PCE expansion.

4.2.1 Karhunen-Loéve expansion (KLE)

A Karhunen-Loéve expansion is a way to decompose a stochastic process in terms of a number of underlying orthogonal random variables, in a way that compactly captures the time domain correlation in the process. Traditionally, it is defined for continuous time processes [14], but can be applied to discrete time processes as well, as was done in [18], whereby a stochastic process $\mathbf{p}[k], k = 0, 1, \ldots, N$, is expressed as:

$$\mathbf{p}[k] = \sum_{i=0}^{N} \sqrt{\lambda_i} f_i[k] \mu_i \quad k = 0, 1, \cdots, N \tag{28}$$

where the μ_i are RVs and where λ_i and $f_i[k]$ are the eigenvalues and eigenfunctions of the autocorrelation matrix of the discrete-time random process $\mathbf{p}[k]$, i.e., they are solutions of:

$$\begin{bmatrix} R(0,0) & R(0,1) & \cdots & R(0,N) \\ R(1,0) & R(1,1) & \cdots & R(1,N) \\ . & . & & . \\ . & . & \cdots & . \\ . & . & & . \\ R(N,0) & R(N,1) & \cdots & R(N,N) \end{bmatrix} \begin{bmatrix} f_i[0] \\ f_i[1] \\ . \\ . \\ . \\ f_i[N] \end{bmatrix} = \lambda_i \begin{bmatrix} f_i[0] \\ f_i[1] \\ . \\ . \\ . \\ f_i[N] \end{bmatrix} \tag{29}$$

where $R(k_1, k_2) = E\left[\mathbf{p}[k_1]\mathbf{p}[k_2]\right]$ is the autocorrelation function. The μ_i are a set of zero-mean orthonormal RVs, so that:

$$E[\mu_i \mu_j] = \delta_{ij} \tag{30}$$

Likewise, the $f_i[k]$ are orthonormal, in the following sense:

$$\sum_{k=0}^{N} f_i[k] f_j[k] = \delta_{ij} \tag{31}$$

The KL expansion (28) can be truncated, yielding a least-squares-optimal expansion on fewer variables $n < N$:

$$\mathbf{p}[k] \approx \sum_{i=0}^{n} \sqrt{\lambda_i} f_i[k] \mu_i \quad k = 0, 1, \cdots, N \tag{32}$$

It can be shown, based on [19], that the relative mean square error resulting from the truncation is given by:

$$e = 1 - \frac{\sum_{i=0}^{n} \lambda_i}{\sum_{i=0}^{N} \lambda_i} \tag{33}$$

where $\lambda_0, \ldots, \lambda_n$ are the eigenvalues that are *kept* in the truncated KL expansion. We will refer to this error term as the *truncation error* of the KL expansion. It can be used as the basis for deciding where and how much to truncate the KLE expansion. The KLE was used in [5] to compute dynamic range for *linear* systems. In this paper, we extend the analysis to the general case of *nonlinear* systems, and this turns out to be possible using PCE. We will now see how a PCE expansion for the input process can be derived from its KLE.

4.2.2 Generating the PCE

From basic probability theory, it is known that if $\mathbf{x}$ is any RV, and if $F(\cdot)$ is its cumulative distribution function (cdf), then $\mathbf{y} = F(\mathbf{x})$ is another RV whose cdf can be shown to be the *uniform distribution* on $[0, 1]$. As a corollary, if $\mathbf{u}$ is an RV which is uniform on $[0, 1]$, and if $F(\cdot)$ is a valid cdf, then $\mathbf{x} = F^{-1}(\mathbf{u})$ is an RV with the cdf $F(\cdot)$. Let $\Phi(\cdot)$ be the cdf of the standard normal (zero mean, unity variance, Gaussian) distribution, and let $F_i(\cdot)$ be the cdf of μ_i (resulting from KLE). Therefore,

$$\xi_i = \Phi^{-1}(F_i(\mu_i)) \tag{34}$$

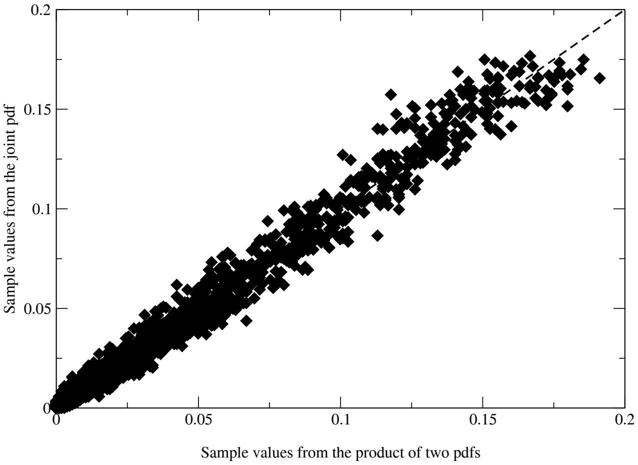

Figure 1: Checking the independence assumption.

is a standard normal RV (Gaussian with mean 0 and variance 1). We propose to use the resulting $\xi_1, \ldots, \xi_n$ as the basis for a polynomial chaos expansion. The advantage, in doing this, would be that we would have an easy way to compute the required PCE coefficients from the KLE for each $\mathbf{p}[k]$. Before we can do that, however, we must first establish that the ξ_i are orthonormal. Since they are standard normal RVs, orthonormality would be established if we could prove that they are also orthogonal, *i.e.*, that $E[\xi_i\xi_j] = 0$. This orthogonality, in turn, would be achieved if it were possible to prove that they are uncorrelated, *i.e.*, that $E[\xi_i\xi_j] = E[\xi_i]E[\xi_j]$. Since they have zero means, this uncorrelatedness would lead to orthogonality, and they would become orthonormal. However, uncorrelatedness is not guaranteed by the above construction (34) of the ξ_i's.

One easy way to achieve uncorrelatedness is if it were the case that the μ_i's are independent. That would lead to the conclusion that the ξ_i's are independent, due to the construction (34), and therefore uncorrelated. However, KLE guarantees that the μ_i's are independent only in the case when the process $\mathbf{p}[k]$ is itself Gaussian. Otherwise, KLE only guarantees that the μ_i's are uncorrelated, which is not enough to establish that the ξ_i's are uncorrelated. To the extent that most data that one expects to see in practice is not too far from having a bell-shaped (Gaussian) distribution, we will *assume* that the μ_i can be *approximated* as being independent in the general case.

We have checked the error of this approximation in the following way. We randomly generated 80,000 sample traces of $\mathbf{p}[k]$ over 10 time points, using a typical Auto-Regression Moving Average (ARMA) model of time series [20], which is extensively used in engineering. We then built the KLE (28) for $\mathbf{p}[k]$, so that we end up with 80,000 random samples of the vector $[\mu_1\ \mu_2\ \cdots\ \mu_n]$. This allows us to check whether the joint distribution of μ_i and μ_j can be approximated as the product of their individual (marginal) distributions. If yes, then μ_i and μ_j may be assumed independent. The distributions were built from the data by simply approximating the pdf with a histogram (80,000 is a large enough data set that this is very well justified). We show the results of this comparison in Fig. 1, where, for every observed (μ_i, μ_j) pair of values, the vertical axis shows the joint pdf of μ_i and μ_j evaluated at that point, $f_{\mu_i,\mu_j}(\mu_i, \mu_j)$, and the horizontal axis shows the product $f_{\mu_i}(\mu_i)f_{\mu_j}(\mu_j)$. Ideally, the data should be right on the diagonal line, but it is not of course. However, it is close enough to the diagonal that we consider that this justifies the independence assumption.

With the ξ_i available as a basis, we can now proceed to obtain the PCE for $\mathbf{p}[k]$ as follows. For each μ_i, consider a one-dimensional PCE expansion, based on ξ_i, so that:

$$\mu_i = \sum_j c_{ij}\Psi_j \tag{35}$$

One can easily compute the values of the coefficients c_{ij}, as follows. To compute c_{ik}, multiply both sides of (35) by Ψ_k and take the expectation on both sides, leading to:

$$c_{ik} = \frac{E[\mu_i\Psi_k]}{E[\Psi_k^2]} \tag{36}$$

The right hand side is a function of the statistics of μ_i and the ξ_i derived from it (recall that, in this one-dimensional PCE, Ψ_k is a function of only ξ_i). The denominator, for one thing, is very easy, and is available from the properties of the polynomial chaos polynomials as we saw earlier in connection with (12). The expectation in the numerator can be computed by a simple Monte Carlo procedure, as follows. Given a large number of user-supplied samples of the input process $\mathbf{p}[k]$ (or, given some probabilistic model of the input process, such an ARMA model, from which one can generate such samples), we construct a KLE (28) and use it to generate samples of the cdf of the μ_i's based on samples of the process $\mathbf{p}[k]$, from which the cdf of each μ_i can be approximated by its empirical cdf. The empirical cdf of an RV $\mathbf{x}$, evaluated at some value x_0, is given by the fraction of observed values of $\mathbf{x}$ that are less than or equal to x_0. Once these cdfs $F_i(\cdot)$ are available, we can use the construction (34) to get the corresponding sample values of the ξ_i's, from which Ψ_k can be computed, and the mean $E[\mu_i\Psi_k]$ computed by Monte Carlo.

4.3 Obtaining variable statistics from PCE

Once the PC expansion for an RV is available, one can use that PCE to compute statistics of various kinds for that variable. Moments of arbitrary order (mean, variance, etc.) can be easily generated [13]. Analytical approximations to the full distribution of that RV can then be obtained from the moments, using techniques such as the Edgeworth expansion [21] and others.

For determining dynamic range and choosing the right bitwidth, it is important to have the cdf, at least the tail of the cdf, in order to estimate the error incurred if bitwidth is reduced. Given that we have in-hand a PCE for an RV, we have found it is very easy and practical to estimate the cdf by Monte Carlo techniques. Given a PCE expansion for an RV, we generate samples of the orthonormal basis $\xi_1, \ldots, \xi_n$ from which we obtain samples of the value of the RV itself, which we use to build an empirical approximation to the cdf $F(x)$, as was described above in connection with (36). It typically takes less than a second of CPU time to compute the cdf in this way.

5. RESULTS

We construct a set of experiments to verify and demonstrate the effectiveness of the proposed methodology. All our experiments are conducted on a Sun workstation (Ultra 80, Model 4450).

Seven benchmarks are selected from practical applications in the DSP domain. Benchmarks 1–3 are polynomial filters. Among them, *volterra* is a third order cubic volterra filter, which derives its name from the volterra expansion of general nonlinear functions [6]; *teager* implements a one-dimensional Teager's algorithm [6] and is commonly used for contrast enhancing in image and speech recognition; *bilinear* is a nonlinear filter whose output is linear with respect to every single system variable. Both benchmarks 4 and 5 are adaptive filters, where *adaptive1* is a regular LMS adaptive filter which adjusts its parameters in proportion to the computed error signal [6], and *adaptive2* adjusts its parameter only according to the sign of the error signal. It is interesting to note that the latter includes conditionals. Benchmarks 6 and 7 are rational filters, which use both multiplication and division.

Sample sets of input random processes with different characteristics were generated to conduct the experiments. These were generated according to the Auto-Regression Moving Average (ARMA) model of time series [20], which is extensively used in engineering. In our experiments, every sample set consisted of 10,000 traces of the process. Such a large number of traces is necessary in order to avoid artifacts caused by finite sample size. This is especially important if we want to capture the tails of the distribution.

Table 2 shows the PCE extraction time from the given data set with different dimension n and order p. Normally the time it takes is on the order of seconds. The time shown in Table 2

Table 2: PCE extraction time.

Dimension	Order	Time (s)
4	4	0.82
6	2	1.09
12	5	2.36
20	5	3.53
30	5	5.53

Table 3: Analysis of speed of PCE v.s. simulation.

#	Benchmark	PCE (s)	Simulation (s)	Speedup
1	volterra	0.10	14.56	145.6
2	teager	0.07	7.18	102.6
3	bilinear	0.05	19.01	380.2
4	adaptive1	0.09	21.64	240.0
5	adaptive2	0.11	20.15	181.0
6	rational1	0.13	13.76	110.1
7	rational2	0.15	16.25	108.3

is reported for 10,000 sample traces of the input random process with length of up to 30 time points.

5.1 Analysis of speed and accuracy

In this section, we demonstrate both the analysis speed and accuracy we can achieve using the proposed method, compared to the traditional profiling-based method. We ran our PCE technique based on $n = 4$ and $p = 3$. In order to compare to simulation or profiling, the question comes up as to how many samples are needed in simulation. In our experience, in order to faithfully reproduce the tail of the distribution in simulation, which is important for bitwidth determination, about 10,000 samples are required. Based on this, the comparison in run-time between PCE and profiling is shown in Table 3, where time is in seconds. Just like all analytical methods, our tool runs order-of-magnitude faster than profiling. For an accuracy comparison, Table 4 shows a comparison of the variance (of the system output) obtained from PCE versus simulation. It is clear that PCE can faithfully re-produce the statistics of the system outputs. Further accuracy comparisons will be given below, where a comparison of dynamic range for the same SNR will be given in section 5.2 and a comparison of the actual cdf curves will be shown in section 5.3.

5.2 Bitwidth and SNR tradeoff

As an application of our work, the distribution and statistics obtained from the PCE-based method can be used to make tradeoff between bitwidth and the signal-to-noise ratio (SNR) or, to phrase it differently, one can synthesize bitwidth under an SNR constraint. The method was detailed in [5]. In this work, we use the simpler metric of range probability in place of SNR. Intuitively, given a signal distribution in pdf form and a quantization scheme, a bitwidth selection corresponds to a dynamic range selection where both tails of the the distribution curve are cut. The area inside the selected dynamic range is called the range probability, and the areas outside the selected dynamic range represent the overflow probability. Thus, under the same SNR constraint, a signal with a sharper distribution can afford to use a narrower dynamic range than those with flatter distributions. In addition, a signal that can afford larger error can use a smaller dynamic range with smaller range probability. In both cases, the bitwidth can be reduced.

In Table 5, we show the different dynamic ranges reported by the PCE method under different range probabilities specified by the user for the output of the *volterra* benchmark. The table also shows the dynamic range from simulation, for the same SNR; the good match between PCE and simulation again emphasizes the accuracy of this technique. Furthermore, because of the fact that PCE can reproduce the entire signal distribution, a spectrum of bitwidth/SNR tradeoff can be made depending on the need of the application. This is in contrast to other analytical methods [1, 2] where only a single dynamic range can be reported.

Table 4: Variance from PCE v.s. simulation.

#	Benchmark	PCE	Simulation	Difference
1	volterra	559.71	549.25	1.9%
2	teager	0.7002	0.7068	0.94%
3	bilinear	3.5195	3.512	0.21%
4	adaptive1	0.0088	0.0087	1.15%
5	adaptive2	0.013	0.0132	1.50%
6	rational1	4.01×10^{-4}	4.03×10^{-4}	0.38%
7	rational2	1.0009	0.991	1.10%

Table 5: Dynamic range and signal-to-noise ratio.

Dynamic Range		Range	SNR
PCE	Simulation	Probability	(dB)
$[-100.05, 99.95]$	$[-102.29, 101.71]$	99.51%	10.49
$[-180.15, 179.85]$	$[-178.29, 178.71]$	99.80%	16.81
$[-269.83, 270.17]$	$[-270.29, 269.71]$	99.97%	26.94
$[-337.83, 338.17]$	$[-340.29, 339.71]$	99.98%	37.99
$[-368.33, 368.67]$	$[-372.29, 371.71]$	99.98%	83.06

5.3 Impact of PCE dimension and order

In general, larger dimension and order imply better accuracy and a more expensive analysis. The exact choice if n and p depends on the input characteristics. To demonstrate the impact of dimension and order on the accuracy of our analysis, we apply two different types of ARMA processes to the *volterra* filter for different dimension/order combinations.

The first process is highly temporally correlated and, in this case, the *output* cdfs under different dimensions and orders are shown in Fig. 2. It can be seen that a larger dimension does not improve the accuracy by much, in this case. This can be explained by the fact that for a highly correlated process, the energy of the KLE expansion is concentrated only on a few terms, therefore a small dimension suffices and this can be determined during the input KLE expansion. On the other hand, reducing the order does degrade the results. Notice that the cdf from simulation is included in the "bundle" of plots corresponding to $p = 3$, and shows that PCE faithfully reproduces the distribution curve. The tail distributions diverge from simulation cdf significantly when the order is reduced to 2 and 1. Other associated statistics, such as means and variances, indicate the same trend.

The output cdfs under different dimensions and orders for the second type of process, which is much less correlated, are shown in Fig. 3. It can be seen that reducing the dimension does affect

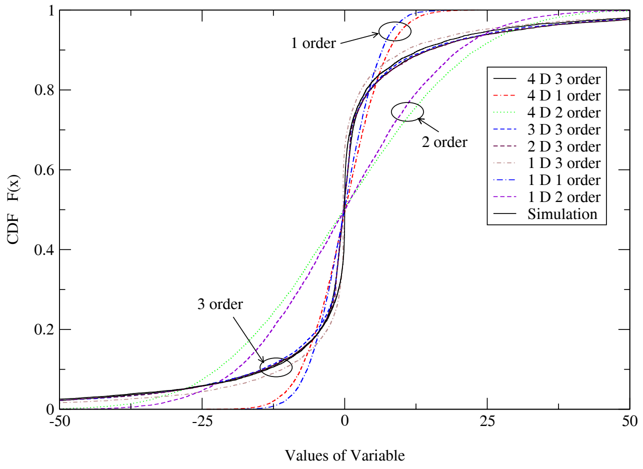

Figure 2: Dimension/order impact for highly correlated input.

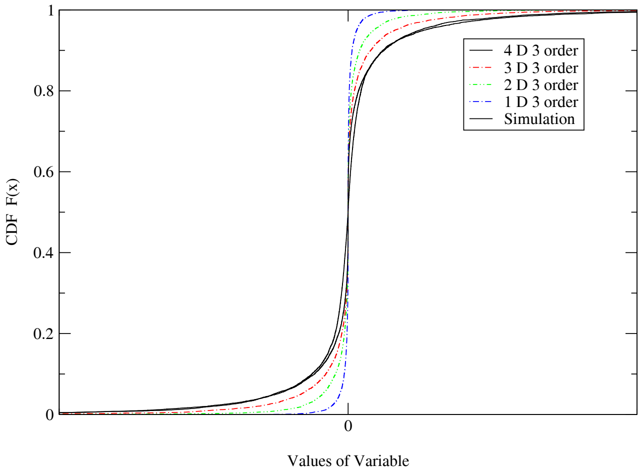

Figure 3: Dimension/order impact for less correlated input.

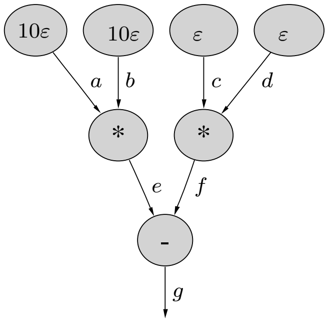

Figure 4: Dataflow graph example.

the accuracy in this case, since signal energy is distributed more evenly. Here too, it is clear that there is a very good match between the two cdfs, from simulation and from PCE, for $n = 4$, $p = 3$.

As a practical matter, in order to determine an appropriate dimension and order, one can consider various alternatives. One can certainly use the KLE expansion (and the eigenvalues resulting from it) to determine what an appropriate dimension n may be, based perhaps on a chosen truncation error, as discussed in connection with (33). Alternatively, since PCE propagation is cheap, one may consider increasing the order and dimension and re-doing the PCE propagation, until minimal change is observed in the results.

5.4 Impact of correlation and nonlinearity

In this section, we will show that the assumptions of correlation and treatments of nonlinearity by previous approaches may incur large errors. To quantify the advantages of the PCE method and offer insight as to why it can perform better, we apply different analysis methods on a common example, shown in Fig. 4, which was extracted from the *teager* benchmark, and compare the statistics obtained against the exact solution. The dataflow graph in the figure computes the equation $g = e - f = ab - cd$. Given an input random process x, we apply $x(n)$, its value at time point n to a and b, $x(n-1)$ to c and $x(n+1)$ to d.

To show the impact of correlation and simplify the demonstration, we assume that the input process is highly correlated so that $x(n), x(n-1), x(n+1)$ are given by 10ε, ε, and ε respectively, where ε is a *uniformly distributed* RV over $[-1, 1]$. It is easy to see that the temporal correlation of the input is translated into the spatial correlation on the dataflow graph, as a, b, c and d can all be characterized by the same RV ε. For this simple example, we can solve for the *exact* statistics at the output, as follows. Since $g = e - f = ab - cd = (10\varepsilon)^2 - \varepsilon^2$, we have $e = 99\varepsilon^2$. Therefore, the exact statistics are:

$$\begin{aligned} E[g] &= E[99\varepsilon^2] = 33 \\ E[g^2] &= E[(99\varepsilon^2)^2] = 99^2/5 = 1960.2 \end{aligned}$$

We now consider the moment propagation approach proposed in [13]. Since it always assumes that the inputs of an operator are independent, then:

$$\begin{aligned} E[g] &= E[a]E[b] - E[c]E[d] \\ &= E[10\varepsilon]E[10\varepsilon] - E[\varepsilon]E[\varepsilon] = 0 \\ E[g^2] &= E[(e-f)^2] = E[e^2 + f^2 - 2ef] \\ &= E[e^2] + E[f^2] - 2E[e]E[f] \\ &= E[10\varepsilon]^2 + E[\varepsilon]^2 = 3333.6667 \end{aligned}$$

It is clear that both the mean and variance predicted by the moment propagation approach deviate significantly from the exact result.

We next consider the affine arithmetic approach proposed in [4], which is able to capture spatial correlation by representing each signal as a linear combination of uniformly distributed RVs. In the case of our example, a, b, c, and d are already in affine form. This representation handles linear operations such as additions precisely, however, the captured correlation may be lost after the nonlinear operations. For example, when multiplications are applied, products of RVs may result, and to keep the result in affine form, a constant bound for each input has to be found, and an extra RV has to be introduced. For example, consider $e = ab$, according to [4], we have:

$$e = 10\varepsilon \times 10\varepsilon = B(10\varepsilon)B(10\varepsilon)\varepsilon_1 = 100\varepsilon_1,$$

where ε_1 is a new RV independent of ε. Following the same procedure, we can derive $g = 100\varepsilon_1 - \varepsilon_2$, where ε_1 and ε_2 are independent RVs. It is evident that the original correlations are lost after the nonlinear operations. As a result, we obtain the statistics as follows:

$$\begin{aligned} E[g] &= E[100\varepsilon_1 - \varepsilon_2] = 0 \\ E[g^2] &= 100^2 E[\varepsilon_1^2] + E[\varepsilon_2^2] \\ &= 3333.6667 \end{aligned}$$

The power of the PCE method lies in the fact that signals are represented as combinations of polynomials, therefore correlations can naturally survive nonlinearity. For the given example, we generate sample data for the uniform RV ε, from which a 1-dimensional 5-order PCE is extracted as follows:

$$\begin{aligned} \epsilon &= 0.002757\Psi_0 + 0.565888\Psi_1 - 0.001018\Psi_2 - \\ &\quad 0.048217\Psi_3 + 0.000139\Psi_4 + 0.003308\Psi_5 \end{aligned}$$

The PCEs for a, b, c, and d can be obtained accordingly. Using the propagation method described in Section 4.1, we can finally obtain:

$$\begin{aligned} g &= 33.2147\Psi_0 + 0.1182\Psi_1 + 18.3929\Psi_2 - \\ &\quad 0.0590\Psi_3 - 2.7226\Psi_4 + .0154\Psi_5 \\ E[g] &= 33.2147 \\ E[g^2] &= 1957.8. \end{aligned}$$

Compared with the exact result, the predicted statistics only have 0.65% and 0.12% error respectively. Fig. 5 shows that the complete distribution obtained by the PCE method matches closely the cdf obtained from simulation.

6. CONCLUSION

In conclusion, we have argued that previous analytical methods for dynamic range estimation are inadequate for handling correlated signals and nonlinear systems. A new method based on a powerful mathematical tool, called polynomial chaos expansion, is proposed. This new method can fully capture the temporal/spatial correlation at the input and consider these correlations during the propagation of range information through nonlinear systems. In addition, our new method automatically extracts the input random process models from real world sample data and thus is much more flexible and accurate than previous analytical

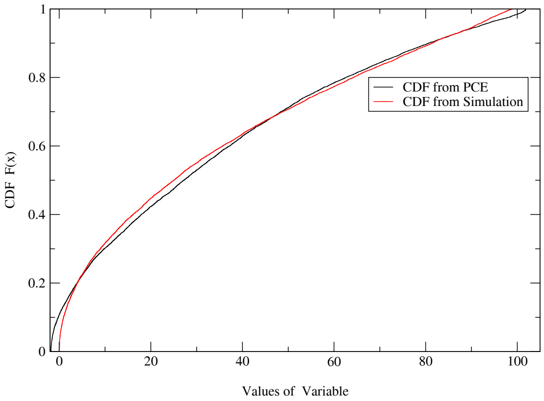

Figure 5: cdf obtained by PCE v.s. simulation for the example in Fig. 4.

methods, which only construct simple input models (such as uniform distribution) from unjustified assumptions. Based on our study, we conclude that the proposed method is as fast as any other analytical method, is accurate enough to handle both temporal and spatial correlations, and gives a complete solution for nonlinear systems.

7. REFERENCES

[1] S. Mahlke, R. Ravindran, M. Schlansker, R. Schreiber, and T. Sherwood. Bitwidth cognizant architecture synthesis of custom hardware accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(11):1355–1371, November 2001.

[2] M. Willems, V. Bursgens, T. Grotker, and H. Meyr. FRIDGE: an interactive code generation environment for HW/SW codesign. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 287–290, Munich, Germany, April 21-24 1997.

[3] A. Garcia-Ortiz, L. Kabulepa, T. Murgan, and M. Glesner. Moment-based power estimation in very deep submicron technologies. In *International Conference on Computer Aided Design*, San Jose, CA, November 9-12 2003.

[4] C. F. Fang, , R. A. Rutenbar, M. Puschel, and T. Chen. Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling. In *Design Automation Conference*, pages 656–661, Anaheim, CA, June 2-6 2003.

[5] B. Wu, J. Zhu, and F. N. Najm. An analytical approach for dynamic range estimation. In *ACM/IEEE 41st Design Automation Conference (DAC-04)*, San Diego, CA, June 7-11 2004.

[6] V. J. Mathews and G. L. Sicuranza. *Polynomial Signal Processing*. John Wiley & Sons, New York, 2000.

[7] Y. Cao and H. Yasuura. A system-level energy minimization approach using datapath width optimization. In *International Symposium on Low Power Electronics and Design*, pages 895–903, Huntington Beach, CA, August 2001.

[8] K. Kum and W. Sung. Combined word-length optimization and high-level synthesis of digital signal processing systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(8):921–930, August 2001.

[9] K. Kum, J. Kang, and W. Sung. A floating-point to integer C converter with shift reduction for fixed-point digital signal processors. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 2163–2166, Phoenix, AZ, March 15-19 1999.

[10] T. Aamodt and P. Chow. Embedded ISA support for enhanced floating-point to fixed-point ANSI C compilation. In *International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pages 128–137, San Jose, CA, November 17-19 2000.

[11] L. B. Jackson. On the interaction of roundoff noise and dynamic range in digital filters. *Bell System Technical Journal*, 49:159–184, February 1970.

[12] J. Carletta, R. Veillette, F. Krach, and Z. Fang. Determining appropriate precisions for signals in fixed-point IIR filters. In *Design Automation Conference*, pages 656–661, Anaheim, CA, June 2-6 2003.

[13] R. G. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Dover Publications, Inc., Mineola, NY, revised edition, 2003.

[14] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, NY, 2nd edition, 1984.

[15] O. P. Le Maitre, O. M. Knio, H. N. Najm, and R. G. Ghanem. A stochastic projection method for fluid flow. *Journal of Computational Physics*, 173:481–511, 2001.

[16] J. C. Park and M. Schlansker. On predicated execution. *Tech. Report. HPL-91-58, HP Laboratories*, May 1991.

[17] V. Chaiyakul, D. Gajski, and L. Ramachandran. High-level transformations for minimizing syntactic variances. In *ACM/IEEE 30th Design Automation Conference (DAC-93)*, Dallas, TX, June 1993.

[18] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 2002.

[19] A. Dur. On the optimality of the discrete karhunen-loeve expansion. *SIAM Journal on Control and Optimization*, 36(6):1937–1939, November 1998.

[20] H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications*. Springer-Verlag, New York, 2000.

[21] P. Hall. *The Bootstrap and Edgeworth Expansion*. Springer-Verlag, Berlin/New York, NY, 1992.