Circuit Uncertainty Quantification in Time Domain using Sensitivity Integrated Stochastic Collocation Method

Yanhao Gong

Master of Science

Department of Electrical and Computer Engineering

McGill University

Montreal,Quebec

August 19, 2020

A thesis submitted to McGill University in partial fullfillment of requirements of the degree of Master of Science

©Yanhao Gong, 2020

DEDICATION

To my parents and myself, thank you!

ACKNOWLEDGEMENTS

My heartfelt appreciation goes to Prof. Roni Khazaka, whose insight, guidance and suggestions were of inestimable value for my journey through Master of Engineering. I am also indebted to my colleague, as well as a good friend of mine, Karan Sidhu, whose assistance and discussion made an enormous contribution to my work. I would also like to express my gratitude to my family in China for their moral support and warm encouragement, especially at this particular time.

ABSTRACT

During the process of designing and fabricating electronic products, uncertainty quantification is a significant component of numerical simulation that assesses the designed devices in the early design phase. As the size of integrated circuits shrinking continuously, the performance of existing simulation tools that designers using to analyze uncertainties needs to be more effectively guaranteed. In the semiconductor industry, designers customarily rely on the solvers that traditionally based on the Monte Carlo algorithm. Despite of its ease of implementation, the Monte Carlo approach is often time-consuming because of a vast number of simulations required. Indeed, the community has proposed many other stochastic solvers involving Polynomial Chaos theory that becomes excellent alternatives to Monte Carlo, such as the Stochastic Collocation method and Stochastic Galerkin method. However, as the growth of the circuit complexity, the CPU cost of these stochastic methods hits the bottleneck spontaneously. As a result, we further extend the Stochastic Collocation method by integrating sensitivity analysis in the time domain to accelerate the process of solving for polynomial chaos coefficients in this thesis. This would directly allow the deterministic system constructed by this newly proposed method to have a smaller dimension compared to Stochastic Collocation Method, i.e. use fewer sampling points, and in turn, minimize the time consumption. Moreover, this whole methodology will be evaluated at both linear and nonlinear transmission line circuits, and compared to Monte Carlo and Stochastic Collocation method in terms of accuracy and CPU cost, respectively.

ABRÉGÉ

Durant le processus de la conception et de la fabrication des produits électroniques, uncertainty quantification est une composante importante de la simulation numérique qui évalue les appareils conçus dans la phase précoce de la conception. Étant donné que la taille des circuits intégrés réduit continuellement, la performance des outils de simulation que les concepteurs utilisent pour analyser les incertitudes doit être plus efficace. Dans les industries de semi-conducteurs, les concepteurs utilisent habituellement les solvers basés sur l'algorithme de Monte Carlo. Malgré sa facilité de mise en œuvre, l'approche Monte Carlo prend souvent beaucoup plus de temps, car un grand nombre de simulations est requis. En effet, la communauté a proposé de nombreux autres stochastic solvers impliquant la théorie du Chaos Polynomial tel que la méthode de Stochastic Collocation et la méthode de Stochastic Galerkin qui sont d'excellentes alternatives. Toutefois, comme les circuits deviennent de plus en plus complexes, le coût CPU de ces méthodes est frappé par un goulot d'étranglement. Pour cette raison, nous approfondissons la méthode Stochastic Collocation en intégrant l'analyse de sensibilité dans le domaine temporel pour accélérer le processus de résolution pour les coefficients de chaos polynomial dans cette thèse. Cela permettra au système construit par cette nouvelle méthode proposée d'avoir une dimension plus petite que la méthode de Stochastic Collocation. Par exemple, cette méthode utilise moins de points d'échantillonnage et, par conséquent, le temps requis est minimisé. De plus, cette méthodologie sera évaluée pour son circuit de

transmission linéaire et non linéaire et elle sera aussi comparée à la méthode de Monte Carlo et de Stochastic Collocation en termes de précision et de coût CPU.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CHAPTER 1
Introduction

1.1 Background and Motivation

In a world full of technology, various devices play a significant role in almost every corner of our daily lives. In order to match the rapid growth for demand and productivity, the key to those devices, electronic systems are designed to be denser and operate under higher frequency while sustaining the exactitude of chip performance or attaining even higher. Nevertheless, manufacturing technology has encountered a bottleneck that the physical limit is looming. Hence, actively seeking for a cure is imminent. As things stand, enhancing the accuracy and diminishing the computational cost in the design process, specifically in circuit simulation, becomes an inevitable and hotspot topic in the semiconductor industry. Nowadays, companies and researchers devote themselves to promoting multiple new methods, which include the Stochastic Galerkin method(SGM) [10], Stochastic Collocation(SCM) method [5], other than Monte Carlo Algorithm(MCM) [2] for system-level circuit simulation tools. Nevertheless, the trade-off between those methods is either feasibility of different designs or the difficulty of implementation. The existing dilemma leads to the necessity of choosing suitable methods for analyzing them during the simulation process.

As the early stage of designing core systems, numerical simulation of the entire integrated circuit is commonly employed by engineers before complex and expensive

manufacturing processes [13]. Simulating the designed circuits or systems is to construct a deterministic model with well-defined physical and geometrical parameters except for MCM, which is a sampling-based method. However, the biggest challenge in simulating designed circuits is to precisely predict how the process variability [7] [17] or parametric uncertainties [23] affect the overall performance of circuits. Essentially, before diving deep into scrutinizing those impacts, we need to quantify uncertainties accurately, where methods mentioned previously, e.g. SGM and SCM, are introduced.

For decades, the semiconductor industry uses the MCM to simulate the statistical distribution of circuit performance. However, the computational cost, especially for large and complex circuits or systems, is not acceptable due to the slow convergence of Monte Carlo itself. Moreover, since MCM is a sampling-based method, the optimization of itself is laborious. Therefore, alternative methods with Generalized Polynomial Chaos(gPC) [21] involved have been proposed. The gPC framework is capable of expanding the circuit response either in the frequency or time domain as a series of orthogonal polynomials [9], where polynomials types are various so that the statistical properties of different types of circuits can be estimated. In this thesis, Hermite Polynomials [18] will be adopted to model the stochastic process with Gaussian random variables. Typically, the mainstream of stochastic analysis methods can be classified into two categories, which are intrusive and non-intrusive [20], depending on whether circuit or system equations are modified or not. Specifically, the non-intrusive solver is a sampling-based model that is easier to be implemented, which has no modification on existing deterministic solver but needs a considerable

amount of simulation times. In contrast, the intrusive solver will build a new deterministic model that is orthogonal to each selected polynomial basis and only needs to be solved once for the stochastic solution.

SCM [5] is one of the non-intrusive solvers which solves the circuit equation in a decoupled manner. This method hugely reduces the computational cost compared to MCM since Polynomial Chaos are used to determine the evolution of uncertainty in circuits while MCM is a sampling-based method. However, with the growth of the complexity of systems and circuits, expediting the design process is always in demand. This thesis offers an approach called Sensitivity Integrated Stochastic Collocation method(SSCM) to optimize the SCM further to lower the computational time in terms of the number of sample points required while retaining the same accuracy. In other words, we need fewer sampling points compared to the SCM criterion [21] because sensitivity will be integrated into our method to substitute certain number of the polynomial chaos expansion sequences.

1.2 Thesis Contributions & Organization

The main contribution of this thesis to the literature mainly embodies at providing a stochastic method with Polynomial Chaos involved called 'Sensitivity Integrated Stochastic Collocation method'(SSCM). As a result of a derivative based method in the frequency domain is proposed in [15], we approach to the time domain. This SSCM analyze the parameters' uncertainty of a circuit that solving in the time domain by creating a modified deterministic matrix system that assisted with

sensitivity to obtain the coefficients of expansion series with less computing effort compared to SCM.

The rest of part of the thesis is organized as follows. Chapter 2 will explore and review the mathematical background and some important concepts of circuit simulation and stochastic analysis, including the preliminaries of both SCM and SSCM in detail, as well as the truncation scheme. An elaborate introduction to numerical methods of sensitivity analysis in the time domain are presented in Chapter 3. Chapter 4 inherits the knowledge from the previous chapters and illustrates how SSCM is implemented in the time domain and its variation on solutions to linear and non-linear circuits. Chapter 5 reports the testing verification for both linear and non-linear transmission line circuits together with accuracy and time consumption comparison between MCM, SCM and SSCM. Ultimately, Chapter 6 summaries the work accomplished in this thesis, as well as outlooks for future research directions.

CHAPTER 2
Literature Review

This chapter starts with introducing the basics of circuit simulation in the time domain, followed by a brief review of several existing essential concepts and methods utilized for our SSCM. Mainly, section 2.2 gives a brief introduction to stochastic analysis. Section 2.4 explores the definition of Generalized Polynomial Chaos and its concrete mathematical variations on univariate and multivariate cases with Hermite Polynomial set in section 2.5. Section 2.6 presents two distinct ways of truncation applied to Polynomials Chaos Expansion. Moreover, several existing stochastic solvers are introduced in section 2.7.

2.1 Transient Circuit Simulation in Time Domain

This thesis mainly proposes a new method to study how the transient circuit response affected by the parameters' uncertainty in the time domain [1]. To obtain the transient response of a circuit in the time domain, we need to solve the circuit equation represented in Modified Nodal Analysis(MNA) form. Equation 2.1 demonstrates the general form of a circuit equation in the time domain, where $\boldsymbol{G}$ and $\boldsymbol{C}$ are two matrices containing all memory-less and memory elements, respectively. $\vec{\boldsymbol{x}}(\boldsymbol{t})$ denotes the nodal voltages and branch current. $\boldsymbol{f}(\vec{\boldsymbol{x}}(\boldsymbol{t}))$ is a vector that stores the information of non-linear elements such as diodes. Besides, the independent input source is stamped in $\boldsymbol{b}(\boldsymbol{t})$ as a vector form as well. The following two sections 2.1.1

and 2.1.2 minutely introduce the solutions to both linear and nonlinear circuit equations in the time domain

$$\boldsymbol{G\vec{x}(t) + C\frac{d\vec{x}(t)}{dt} + f(\vec{x}(t)) = b(t)} \tag{2.1}$$

2.1.1 Transient Solution to Linear Circuits

To simulate the circuit containing only linear elements or, in other words, solve the linear MNA equation in the time domain, we consider using the Backward Euler method because 2.1 is basically an ordinary differential equation, where $\boldsymbol{f(\vec{x}(t))}$ is all zero. With the definition of the Backward Euler method in 2.2 and substituting it into 2.1, the system equation becomes the form represented in 2.3, where $\boldsymbol{h = t_{n+1} - t_n}$ is an adaptive time step size.

$$\boldsymbol{\frac{dx(t)}{dt} = \frac{x(t_{n+1}) - x(t_n)}{h}} \tag{2.2}$$

$$\boldsymbol{G\vec{x}(t_{n+1}) + \frac{C}{h}(\vec{x}(t_{n+1}) - \vec{x}(t_n)) = b(t_{n+1})} \tag{2.3}$$

For convenience, let $\boldsymbol{\vec{x}_{n+1} = \vec{x}(t_{n+1})}$, $\boldsymbol{\vec{x}_n = \vec{x}(t_n)}$ and $\boldsymbol{b(t_{n+1}) = b_{n+1}}$ used in the following equations. We can directly apply LU decomposition to the matrix within the parentheses of left side in 2.5, which is the organized form of 2.3.

$$\boldsymbol{G + \frac{C}{h} = LU} \tag{2.4}$$

Hereafter, with an initial guess to $\boldsymbol{\vec{x}_0}$, normally a all zero vector is selected, and all other known components of this equation, $\boldsymbol{\vec{x}_{n+1}}$ at each time point can be solved

iteratively, as 2.6 demonstrates.

$$(\boldsymbol{G} + \frac{\boldsymbol{C}}{\boldsymbol{h}})\vec{\boldsymbol{x}}_{n+1} = \boldsymbol{b}_{n+1} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\vec{\boldsymbol{x}}_n \tag{2.5}$$

$$\begin{aligned}
\vec{\boldsymbol{x}}_1 &= \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}(\boldsymbol{b}_{n+1} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\vec{\boldsymbol{x}}_0) \\
\vec{\boldsymbol{x}}_2 &= \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}(\boldsymbol{b}_{n+1} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\vec{\boldsymbol{x}}_1) \\
&\vdots \\
\vec{\boldsymbol{x}}_{n+1} &= \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}(\boldsymbol{b}_{n+1} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\vec{\boldsymbol{x}}_n)
\end{aligned}$$

2.1.2 Transient Solution to Nonlinear Circuits

The approach to solve nonlinear circuit equation is sightly different from linear case with the presence of $\boldsymbol{f}(\vec{\boldsymbol{x}})$. To begin with, we need to apply the Newton's Iteration method to 2.6. Specifically, let $\boldsymbol{R}(\vec{\boldsymbol{x}})$ denotes the circuit equation 2.7. Then the approximation to the roots can be represented by 2.8, where $\boldsymbol{R'}(\vec{\boldsymbol{x}})$ has a form of $\boldsymbol{G} + \frac{\boldsymbol{C}}{\boldsymbol{h}} + \boldsymbol{J}(\vec{\boldsymbol{x}})$, in which $\boldsymbol{J}(\vec{\boldsymbol{x}})$ is the Jacobian matrix of nonlinear vector $\boldsymbol{f}(\vec{\boldsymbol{x}})$. Finally, similar to the linear case, we compute the change from $\vec{\boldsymbol{x}}_n$ to $\vec{\boldsymbol{x}}_{n+1}$ at each time point until convergence with an initial guess to $\vec{\boldsymbol{x}}_0$. Generally, the breaking condition of iterative loop for computing $\boldsymbol{\Delta}\vec{\boldsymbol{x}}$ is set to its normalization less than the machine error.

$$(\boldsymbol{G} + \frac{\boldsymbol{C}}{\boldsymbol{h}})\vec{\boldsymbol{x}}_{n+1} + \boldsymbol{f}(\vec{\boldsymbol{x}}_{n+1}) = \boldsymbol{b}_{n+1} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\vec{\boldsymbol{x}}_n \tag{2.6}$$

$$\boldsymbol{R}(\vec{\boldsymbol{x}}) = (\boldsymbol{G} + \frac{\boldsymbol{C}}{\boldsymbol{h}})\vec{\boldsymbol{x}}_{n+1} + \boldsymbol{f}(\vec{\boldsymbol{x}}_{n+1}) - (\boldsymbol{b}_{n+1} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\vec{\boldsymbol{x}}_n) \tag{2.7}$$

$$\vec{x}_{n+1} - \vec{x}_n = \Delta x = -\frac{R(\vec{x})}{R'(\vec{x})} \tag{2.8}$$

2.2 Stochastic Analysis

Stochastic analysis is a commonly used tool in modelling complex engineering problems involving uncertainties. Specifically, in the electronic design automation area, stochastic analysis is an indispensable step that been done in the circuit simulation process to help designers to foresee the possible errors that may occur during the manufacturing process and therefore adjust accordingly. As such, manufacturing vendors are able to ensure the quality of products and avoid the expensive cost for re-fabrication. This particular step is also known as variability analysis [14]. As illustrated in Figure 2–1, the system block is affected by the random variables, and the output of this deterministic system is randomized, consequently concerning different parameters.

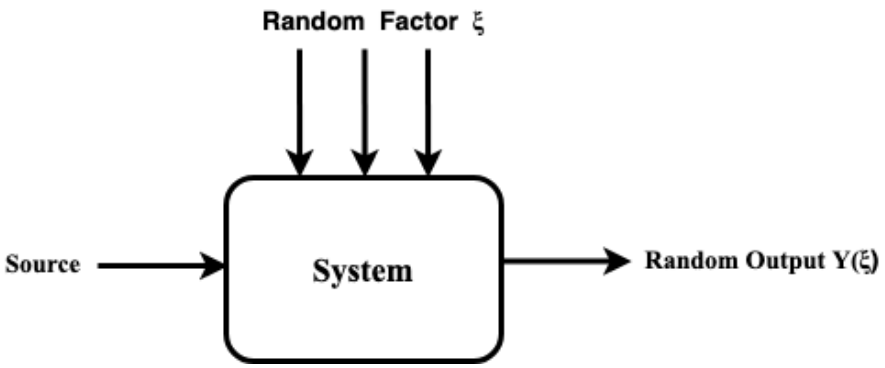

Figure 2–1: Stochastic Process

In terms of variability analysis in circuit simulation, let $\vec{\boldsymbol{\xi}} = [\vec{\boldsymbol{\xi^1}}; \vec{\boldsymbol{\xi^2}}; \cdots \vec{\boldsymbol{\xi^m}}]$ denotes the random variables set, the resulted MNA equation is presented in 2.9, in which $\boldsymbol{G}(\vec{\boldsymbol{\xi^i}}) = \boldsymbol{G_\mu} + \vec{\boldsymbol{\xi^i}}\boldsymbol{G_\sigma}$. Particularly, $\boldsymbol{G_\mu}$ and $\boldsymbol{G_\sigma}$ are mean value and standard deviation of corresponding memory-less element, and same pattern applies to $\boldsymbol{C}$. $\boldsymbol{f}(\vec{\boldsymbol{x}}, \boldsymbol{\xi^i})$ has a form of 2.10, e.g. a diode

$$\boldsymbol{G}(\vec{\boldsymbol{\xi}})\vec{\boldsymbol{x}} + \boldsymbol{C}(\vec{\boldsymbol{\xi}})\dot{\vec{\boldsymbol{x}}} + \boldsymbol{f}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{\xi}}) = \boldsymbol{b} \tag{2.9}$$

$$\boldsymbol{f}(\vec{\boldsymbol{x}}, \boldsymbol{\xi^i}) = \begin{bmatrix} (\boldsymbol{I_s})_\mu \boldsymbol{e}^{\frac{x}{V_T}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} + \boldsymbol{\xi^i} \begin{bmatrix} (\boldsymbol{I_s})_\sigma \boldsymbol{e}^{\frac{x}{V_T}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{2.10}$$

Once the circuit equations are built up, we make use of different methods of interest to solve 2.9 for statistical information.

2.3 Monte Carlo Method

There are existing distinct methods that can be used for stochastic analysis. Among all of them, the Monte Carlo method is the most common and straightforward approach to the variability analysis due to its ease of implementation. However, it is also time-consuming since it relies on repeatedly simulating circuits with different sets of random parameters assigned to circuit elements. The values of random parameters are generated by a specific probability density function. In this thesis, we rely on the built-in function in MATLAB to generate the random points with

Gaussian distribution. The main steps of performing MCM are summarized as the following:

I Determine the domain of random parameters and their distribution.

II Generate sets of random parameters based on the domain and its distribution, the resulted samples can be a multi-dimensional vector if multi-variate case is considered.

III Repeatedly solve the circuit with each sample point generated at previous step fitted into circuit equation, e.g. 2.9

IV Acquire the results of circuit simulation and analyze for statistical information.

Generally, we utilize the mean and variance value of MC simulations to estimate the statistical information, i.e. random parameters. Assuming a MC simulation process has $\boldsymbol{N}$ repeated simulations with a discrete set of random parameters obtained by a random generator and its corresponding output set $\boldsymbol{\{y_i\}}$, the mean value $\boldsymbol{\mu}$ can be estimated as the expected value of $\boldsymbol{y}$ approximated by the population mean value $\tilde{\boldsymbol{\mu}}$:

$$\boldsymbol{\mu = E\{y\} \approx \tilde{\mu} = \frac{1}{N}\sum_{i=1}^{N} y_i} \tag{2.11}$$

This equation is used as a good estimator since it is unbiased, meaning that 2.11 establishes the equality between the expected value and the true value of the quantity it estimates.

Similarly, let $\tilde{\boldsymbol{\sigma}}$ denotes the standard deviation of a MC simulation, the estimator of the variance of $\boldsymbol{y}$ is calculated as follow:

$$\boldsymbol{Var(y) \approx \tilde{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \tilde{\mu})^2} \tag{2.12}$$

Notably, $\tilde{\boldsymbol{\sigma}}^{\mathbf{2}}$ is also a random variable since $\boldsymbol{y_i}$ is randomly generated, and $\tilde{\boldsymbol{\mu}}$ has different values with different MC simulations on the same system equations. Thus, the expected value of unbiased variance $\boldsymbol{\sigma}^{\mathbf{2}}$ can be represented as:

$$\boldsymbol{E\{\sigma^2\} = E}\left[\frac{\boldsymbol{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}\left(\boldsymbol{y_i} - \frac{\boldsymbol{1}}{\boldsymbol{N}}\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}}\boldsymbol{y_j}\right)^{\mathbf{2}}\right] = \frac{\boldsymbol{N-1}}{\boldsymbol{N}}\tilde{\boldsymbol{\sigma}}^{\mathbf{2}} \tag{2.13}$$

and therefore the unbiased variance estimator:

$$\boldsymbol{\sigma^2} = \frac{\boldsymbol{1}}{\boldsymbol{N-1}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{y_i} - \boldsymbol{\mu})^{\mathbf{2}} \tag{2.14}$$

2.4 Generalized Polynomial Chaos Framework

To overcome the limitation of MCM, the proposed method is one of the polynomial chaos approaches. Circuit responses can be modelled during the stochastic process by a truncated series of orthogonal polynomials with certain distributed random variables, so-called Polynomial Chaos Expansion. Generalized Polynomial Chaos Expansion is extended from PC by Dr.Xiu [21] to various continuous and discrete distributions using orthogonal polynomials from Askey-scheme to solve for both Gaussian and non-Gaussian problems. The selection of different polynomials is based on the probability distribution of random variables. The widespread distribution and corresponding orthogonal polynomials pairs are summarized in Table 2–1.

As mentioned previously, the random variables used in this thesis are Gaussian distributed. Therefore, Hermite polynomials Expansion is chosen to model the

Table 2–1: Popular distribution and orthogonal polynomials pairs of generalized polynomial chaos

Orthogonal Polynomials	Distribution of Random Variables
Legendre polynomials	Uniform Distribution
Hermite polynomials	Gaussian Distribution
Jacobi polynomials	Beta Distribution
Laguerre polynomials	Gamma Distribution

stochastic process correspondingly and used to explore other concepts in the following several sections. The mathematical approximation of gPC is demonstrated in Equation 2.15,

$$\vec{\boldsymbol{x}}(\vec{\boldsymbol{\xi}}) = \sum_{\vec{\boldsymbol{\alpha}}} \boldsymbol{A}_{\vec{\boldsymbol{\alpha}}} \boldsymbol{H}_{\vec{\boldsymbol{\alpha}}}(\vec{\boldsymbol{\xi}}) \tag{2.15}$$

where $\vec{\boldsymbol{x}}(\boldsymbol{\xi})$ is a vector of circuit responses that affected by random variables, $\boldsymbol{H}_{\vec{\boldsymbol{\alpha}}}(\vec{\boldsymbol{\xi}})$ is the Hermite polynomial basis in this case, and $\boldsymbol{A}$ is the corresponding coefficients or weights. $\vec{\boldsymbol{\alpha}}$ is a set of selected index vectors, which will be further discussed in later sections. The basis functions $\boldsymbol{H}$ are chosen in a special way in order to meet the orthonormal condition, i.e. In a stochastic space $\boldsymbol{S}$ with a probability density function $\boldsymbol{\rho}(\vec{\boldsymbol{\xi}})$, we have

$$\langle \boldsymbol{H}_{\vec{\boldsymbol{\alpha}}}(\vec{\boldsymbol{\xi}}), \boldsymbol{H}_{\vec{\boldsymbol{\beta}}}(\vec{\boldsymbol{\xi}}) \rangle_{\boldsymbol{S},\boldsymbol{\rho}(\vec{\boldsymbol{\xi}})} = \boldsymbol{\delta}_{\vec{\boldsymbol{\alpha}},\vec{\boldsymbol{\beta}}} \tag{2.16}$$

where $\boldsymbol{\delta}_{\vec{\boldsymbol{\alpha}},\vec{\boldsymbol{\beta}}}$ is a Delta function. The computation of Hermite Polynomial basis functions is described in the following sections.

2.5 Hermite Polynomials Expansion

Hermite Polynomials Expansion [18] has two different standardized types, which are ‘probabilists’’ and ‘physicists’’. Considering a random variable as $\boldsymbol{\xi}$, the polynomials function $\boldsymbol{H_n(\xi)}$ of each type is defined as follow.

- Probabilists’ Polynomials

$$\boldsymbol{H_n(\xi) = (-1)^n e^{\frac{\xi^2}{2}} \frac{d^n}{d\xi^n} e^{-\frac{\xi^2}{2}}} \tag{2.17}$$

- Physicists’ Polynomials

$$\boldsymbol{H_n(\xi) = (-1)^n e^{\xi^2} \frac{d^n}{d\xi^n} e^{-\xi^2}} \tag{2.18}$$

In fact, each of these two is a re-scaling of the other with respect to different weight functions. However, *probabilists’* polynomial is used by this thesis because the weight function of itself is the probability density function for Gaussian distribution with the expected value and standard deviation of 0 and 1 respectively, which precisely fits the requirement of SSCM.

2.5.1 Derivation to Univariate Case of Hermite Polynomial

Hermite polynomials sequence satisfies three-term recurrence relation [4] with initial condition $\boldsymbol{H_0 = 1}$ and $\boldsymbol{H_1 = \xi}$. See Equation 2.19.

$$\boldsymbol{H_{n+1}(\xi) = \xi H_n(\xi) - H_n^{'}(\xi)} \tag{2.19}$$

Where $\boldsymbol{n > 1}$. As so, each term of the polynomials can be derived, and the first six terms are shown in Table 2–2. As an orthogonal polynomial [9], the orthogonality of

Hermite polynomials can be easily verified by Equation 2.20.

$$\int_{-\infty}^{\infty} \boldsymbol{H_m(\xi)H_n(\xi)W(\xi)d\xi} = \int_{-\infty}^{\infty} \boldsymbol{H_m(\xi)H_n(\xi)e^{\frac{-\xi^2}{2}}d\xi} = \boldsymbol{\sqrt{2\pi}m!\delta_{mn}} \tag{2.20}$$

Where $\boldsymbol{\delta_{mn}}$ is the *Kronecker delta* function.

Based on Equation 2.15, assuming a stochastic process with single Gaussian random variable $\boldsymbol{\xi}$, output as $\boldsymbol{x}$ and every single coefficient as $\boldsymbol{a_n}$, the Hermite polynomials expansion of this system is defined as:

$$\boldsymbol{x(\xi) = a_0H_0(\xi) + a_1H_1(\xi) + a_1H_1(\xi) + \cdots = \sum_{n=0}^{\infty} a_nH_n(\xi)} \tag{2.21}$$

Table 2–2: First 6 terms of Hermite Polynomials

Order	Hermite polynomials
0	$\boldsymbol{1}$
1	$\boldsymbol{\xi}$
2	$\boldsymbol{\xi^2 - 1}$
3	$\boldsymbol{\xi^3 - 3\xi}$
4	$\boldsymbol{\xi^4 - 6\xi^2 + 3}$
5	$\boldsymbol{\xi^5 - 10\xi^3 + 15\xi}$

2.5.2 Derivation to Multivariate Case of Hermite Polynomial

The multivariate case can be generalized based on the univariate case. Considering a relatively more complex circuit with uncertainties from multiple elements, each of the random variable set $\boldsymbol{\vec{\xi} = \{\vec{\xi^1}; \vec{\xi^2}; \cdots ; \vec{\xi^m}\}}$ of a size $\boldsymbol{m}$ can be constructed if its members are mutually independent. The expansion under this circumstance is:

$$\boldsymbol{\vec{x}(\vec{\xi}) = \sum_{\vec{\alpha}} \vec{A}_{\vec{\alpha}}\phi_{\vec{\alpha}}(\vec{\xi})} \tag{2.22}$$

Where $\boldsymbol{\phi_{\vec{\alpha}}}$ is the multi-variable polynomial chaos defined as:

$$\phi_{\vec{\alpha}}(\vec{\xi}) = \prod_{i}^{m} \boldsymbol{H}_{\vec{\alpha}_i}(\vec{\xi^i}) \tag{2.23}$$

Furthermore, $\vec{\boldsymbol{\alpha}}$ is the index vector, in which every single entry, $\vec{\boldsymbol{\alpha}}_{\boldsymbol{i}}$, points to each univariate polynomial of the different degrees in the direction $\boldsymbol{\xi^i}$ with a one-to-one relation, up until the highest degree $\boldsymbol{p}$.

The multivariate Hermite polynomials Expansion also satisfies the orthogonal property with respect to weighting function 2.25.

$$\langle \phi_{\vec{\alpha}}(\vec{\xi}), \phi_{\vec{\beta}}(\vec{\xi}) \rangle = \int_{\Omega} \phi_{\vec{\alpha}}(\vec{\xi}) \phi_{\vec{\beta}}(\vec{\xi}) \boldsymbol{W}(\vec{\xi}) d\vec{\xi} = \delta_{\vec{\alpha},\vec{\beta}} \tag{2.24}$$

$$\boldsymbol{W}(\vec{\xi}) = \sqrt{\boldsymbol{2\pi^m}}^{\boldsymbol{-1}} e^{\frac{-\|\xi\|^2}{2}} \tag{2.25}$$

2.6 Truncation Scheme of Polynomial Sequence Expansion

As seen in Equation 2.21, the number of basis functions of Hermite Polynomials Chaos Expansion can be extended to infinity, especially for multivariate cases due to its multi-indices. It is meaningless to use as more as basis functions to expand the random parameters since a finite number of functions can produce a highly accurate result. As such, the concept of truncation [12] is introduced here to cut down the number of functions in the expansion sequence. For example, given an order threshold $\boldsymbol{p}$, the Hermite Polynomials will only generate $\boldsymbol{p+1}$ terms since all terms with an order higher than $\boldsymbol{p}$ is truncated.

As for the multi-indices situation, considering the degree selection set $\boldsymbol{P}$ and an order $\boldsymbol{p}$, the most two conventional approaches [23] to truncation are:

- *Total Degree Method*: The degree of multivariate term $\boldsymbol{\phi_\alpha}$ in expansion that satisfy the condition in Equation 2.26 will be included in $\boldsymbol{P}$.

$$\sum_{i=1}^{m} \boldsymbol{\alpha_i} \leq \boldsymbol{p} \tag{2.26}$$

  As a result, the number of basis functions $\boldsymbol{N}$ can be calculated by Equation 2.27.

$$\boldsymbol{N} = \frac{(\boldsymbol{m}+\boldsymbol{p})!}{\boldsymbol{m}!\boldsymbol{p}!} \tag{2.27}$$

- *Tensor Order Method*: Term $\boldsymbol{\phi_\alpha}$ is included in set $\boldsymbol{P}$ only if its multi-indices satisfy Equation 2.28.

$$\boldsymbol{0} \leq \boldsymbol{\alpha_i} \leq \boldsymbol{p} \tag{2.28}$$

  The total number of basis function $\boldsymbol{N}$ is derived from Equation 2.29.

$$\boldsymbol{N} = \prod_{i=1}^{t} (\boldsymbol{p_i}+\boldsymbol{1})^{\boldsymbol{m}} \tag{2.29}$$

Table 2–3 shows an example of using total degree method with $\boldsymbol{m} = \boldsymbol{3}$ and $\boldsymbol{p} = \boldsymbol{3}$.

Table 2–3: Truncation Example of using Total Degree method with $\boldsymbol{m = 3}$ and $\boldsymbol{p = 3}$

Total Order	Multi-indices	Single Index of $\boldsymbol{\alpha}$
0	(0 0 0)	1
1	(1 0 0)	2
1	(0 1 0)	3
1	(0 0 1)	4
2	(2 0 0)	5
2	(0 2 0)	6
2	(0 0 2)	7
2	(1 1 0)	8
2	(1 0 1)	9
2	(0 1 1)	10
3	(3 0 0)	11
3	(0 3 0)	12
3	(0 0 3)	13
3	(0 1 2)	14
3	(0 2 1)	15
⋮	⋮	⋮

2.7 Existing Stochastic Solver Exploration

This section provides a brief exploration into several popular stochastic solvers involving polynomial chaos theory. As we mentioned before, there are two main categories of the techniques that used to analyze circuit variability, which are intrusive and non-intrusive. Monte Carlo is one of the non-intrusive family since it uses a fixed number of samples to directly solve the computational model 2.9 for statistical information.

2.7.1 Stochastic Collocation Method

Stochastic Collocation method(SCM)[5] is the best-known non-intrusive stochastic method, and our proposed method is improved based on SCM. The general idea of SCM is to approximate the nodal voltage or branch current $\vec{\boldsymbol{x}}$ with a deterministic system, which essentially is a linear combination of polynomial chaos basis functions shown in 2.30. Assuming there are $\boldsymbol{M}$ deterministic equations in total and each basis function after applying truncation scheme has $\boldsymbol{K}$ terms, then the linear combination $\boldsymbol{H} \in \mathbb{R}^{\boldsymbol{M}\times\boldsymbol{K}}$ with each row been filled with $\boldsymbol{\phi}(\vec{\boldsymbol{\xi}^{\boldsymbol{i}}})$, the polynomial chaos coefficients $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{K}}$, and the circuit response $\boldsymbol{V} \in \mathbb{R}^{\boldsymbol{M}}$ with $\boldsymbol{V}(\boldsymbol{k}) = \boldsymbol{V}(\vec{\boldsymbol{\xi}^{\boldsymbol{i}}})$. Notably, the $\vec{\boldsymbol{\xi}}$ here denotes the random variable set or the collocation points chosen from random space by a quadrature rule, e.g. Gaussian Quadrature Rule.

$$\boldsymbol{HA} = \boldsymbol{V} \tag{2.30}$$

Similar to Monte Carlo, the circuit response is obtained by solving equation 2.9 with $\vec{\boldsymbol{\xi}}$ fitted in. However, SCM requires fewer sample points than MCM, which makes it

much faster in terms of convergence speed.

2.7.2 Stochastic Galerkin Method

Stochastic Galerkin method(SGM) is an intrusive spectral method that involving gPC. Similar to SCM, it starts by generating random parameters with an appropriate probability density function, and then construct Polynomial chaos basis function to model the circuit response $\vec{\boldsymbol{x}}$. The hinge that makes SGM an intrusive method is that it applies the Galerkin projection procedure [20] to aggregate a set of coupled deterministic equations instead of using stochastic equations. Statistical information can be obtained from the solution to the constructed system solved by an applicable numerical method.

Whereas, the efficiency of SGM degrades expeditiously as the growth of number of random parameters and the complexity of circuits since all deterministic equations are solved in a coupled manner.

2.7.3 Stochastic Testing Method

Stochastic Testing method [23] is a generalized polynomial chaos-based intrusive method, which varies from the interpolation-based stochastic collocation. The main difference from SCM is that Stochastic Testing method directly solves a large-scaled deterministic system for polynomial chaos coefficients without decoupling techniques applied to the system itself. Instead, it computes the Jacobian matrix of difference equation in 2.31 that can be decoupled during iterations. Moreover, this method uses a point selection technique to reduce the number of quadrature points required.

There are totally $\boldsymbol{K}$ samples selected from a set of $(\boldsymbol{p}+\mathbf{1})^{\boldsymbol{m}}$ points generated by a Gaussian-quadrature tensor product rule, where $\boldsymbol{K} \ll (\boldsymbol{p}+\mathbf{1})^{\boldsymbol{m}}$ and $\boldsymbol{p}$ is the highest total order of polynomial chaps basis functions. This particular algorithm also inspires our proposed SSCM explained in the later chapter.

$$\boldsymbol{T}(\vec{\boldsymbol{x}}(\vec{\boldsymbol{\xi}})) = \tilde{\boldsymbol{G}}\vec{\boldsymbol{x}}(\vec{\boldsymbol{\xi}}) + \tilde{\boldsymbol{C}}\dot{\vec{\boldsymbol{x}}}(\vec{\boldsymbol{\xi}}) + \tilde{\boldsymbol{F}}(\vec{\boldsymbol{x}}(\vec{\boldsymbol{\xi}}), \vec{\boldsymbol{\xi}}) - \tilde{\boldsymbol{B}} = \mathbf{0} \tag{2.31}$$

This method begins with approximating the exact solution $\vec{\boldsymbol{x}}$ in MNA equation 2.9 by truncated generalized polynomial chaos expansion $\vec{\boldsymbol{x}}(\boldsymbol{\xi})$ with selected quadrature points fitted in, such as Hermite polynomials in 2.22. This operation results in the equation in 2.31, where

$$\tilde{\boldsymbol{G}} = \begin{bmatrix} \boldsymbol{G}(\vec{\boldsymbol{\xi^1}}) & & & \\ & \boldsymbol{G}(\vec{\boldsymbol{\xi^2}}) & & \\ & & \ddots & \\ & & & \boldsymbol{G}(\vec{\boldsymbol{\xi^K}}) \end{bmatrix}, \tilde{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{C}(\vec{\boldsymbol{\xi^1}}) & & & \\ & \boldsymbol{C}(\vec{\boldsymbol{\xi^2}}) & & \\ & & \ddots & \\ & & & \boldsymbol{C}(\vec{\boldsymbol{\xi^K}}) \end{bmatrix} \tag{2.32}$$

$$\tilde{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{b} \\ \vdots \\ \boldsymbol{b} \end{bmatrix}, \tilde{\boldsymbol{F}} = \begin{bmatrix} \boldsymbol{f}(\vec{\boldsymbol{\xi^1}}, \vec{\boldsymbol{x}}(\boldsymbol{\xi})(\vec{\boldsymbol{\xi^1}})) \\ \boldsymbol{f}(\vec{\boldsymbol{\xi^2}}, \vec{\boldsymbol{x}}(\boldsymbol{\xi})(\vec{\boldsymbol{\xi^2}})) \\ \vdots \\ \boldsymbol{f}(\vec{\boldsymbol{\xi^K}}, \vec{\boldsymbol{x}}(\boldsymbol{\xi})(\vec{\boldsymbol{\xi^K}})) \end{bmatrix} \tag{2.33}$$

Let $\tilde{\boldsymbol{\lambda}}$ denotes the polynomial chaos coefficients, then 2.31 leads to the equation 2.34 with Backward Euler Integration as an example.

$$\boldsymbol{T}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}}) = \tilde{\boldsymbol{G}}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}}) + \frac{\mathbf{1}}{\boldsymbol{h}}(\tilde{\boldsymbol{C}}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}}) - \tilde{\boldsymbol{C}}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k-1}}, \vec{\boldsymbol{\xi}})) + \tilde{\boldsymbol{F}}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}}) - \tilde{\boldsymbol{B}} = \mathbf{0} \tag{2.34}$$

As a result, 2.34 can be solved directly with Newton Iteration as coupled structure instead of block by block, i.e.

$$\boldsymbol{J}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}})\Delta\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}} = -\boldsymbol{T}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}}) \tag{2.35}$$

where $\boldsymbol{J}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}})$ is the Jacobian matrix of $\boldsymbol{T}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}})$. Generally, we start with giving an initial guess to $\tilde{\boldsymbol{\lambda}}_{\mathbf{0}}$ and computed all coefficients until convergence. However, the author of [23] proposed a specialized way by using a Vandermonde-like matrix $\boldsymbol{V} \in \mathbb{R}^{\boldsymbol{k}\times\boldsymbol{k}}$ with $\boldsymbol{V}(\boldsymbol{i}, \boldsymbol{k}) = \boldsymbol{\phi}_{\boldsymbol{k}}(\vec{\boldsymbol{\xi}^{\boldsymbol{i}}})$ that only dependent on the sampling points and polynomial basis functions to solve the system in a decoupled manner. The resulted time complexity of this algorithm has a linear scale. As so, the Jacobian matrix can be transformed into the following form

$$\boldsymbol{J}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}}) = \tilde{\boldsymbol{J}}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}})(\boldsymbol{V} \otimes \boldsymbol{I}_{\boldsymbol{n}}) \tag{2.36}$$

where $\boldsymbol{I}_{\boldsymbol{n}}$ is an identity matrix of size $\boldsymbol{n} \times \boldsymbol{n}$ and operator $\otimes$ denotes the Kronecker product. Afterwards, we first solve

$$\tilde{\boldsymbol{J}}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}})\Delta\boldsymbol{\Lambda} = -\boldsymbol{T}(\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}}, \vec{\boldsymbol{\xi}})$$

iteratively and calculate the result by

$$\Delta\tilde{\boldsymbol{\lambda}}_{\boldsymbol{k}} = (\boldsymbol{V}^{-\mathbf{1}} \otimes \boldsymbol{I}_{\boldsymbol{n}})\Delta\boldsymbol{\Lambda}$$

simultaneously. Moreover, the computation of $\boldsymbol{V}^{-\mathbf{1}}$ only requires once and can be reused for all time points.

This process hugely increases the efficiency of computing polynomial chaos coefficients because after multiplying the Kronecker product part to the Jacobian matrix, it turns into a spares matrix with a block-diagonal structure:

$$\tilde{\boldsymbol{J}}(\tilde{\boldsymbol{\lambda}}_k^i, \vec{\boldsymbol{\xi}}) = \begin{bmatrix} \boldsymbol{J}(\tilde{\boldsymbol{\lambda}}_k^i, \vec{\boldsymbol{\xi}^1}) & & \\ & \ddots & \\ & & \boldsymbol{J}(\tilde{\boldsymbol{\lambda}}_k^i, \vec{\boldsymbol{\xi}^K}) \end{bmatrix} \tag{2.37}$$

CHAPTER 3
Sensitivity Analysis

Before we run into the implementation of SSCM, we need to introduce the sensitivity analysis [6], which is the fundamental part that makes our method unconventional and optimized. This chapter emphasizes on demonstrating approaches to calculating sensitivity in the time domain with the presence of the deterministic system.

3.1 Sensitivity Definition

Sensitivity analysis is the way to model the outcome of a system concerning the effects of different sources of uncertainties in inputs. Notably, in the circuit simulation area, the sensitivity is a significant metric to model the change of the response of a circuit along with the variety of different electronic elements. In other words, the sensitivity of a circuit is defined as the ratio of the change in node voltage or branch current to the change in different electronic elements such as resistors, capacitors and inductors. Mathematically, e.g., consider the circuit in Figure 3–1, the sensitivity of voltage at node 2 with respect to the change of $\boldsymbol{C_1}$ is derived as follows. Assuming $\boldsymbol{C_1}$ changes by $\boldsymbol{\Delta C_1}$ and $\boldsymbol{V_2}$ accordingly changes by $\boldsymbol{\Delta V_2}$, the sensitivity is defined as:

$$\frac{\boldsymbol{\Delta V_2}}{\boldsymbol{\Delta C_1}} \rightarrow \frac{\boldsymbol{\partial V_2}}{\boldsymbol{\partial C_1}} \tag{3.1}$$

as $\boldsymbol{\Delta C_1}$ approaches to 0.

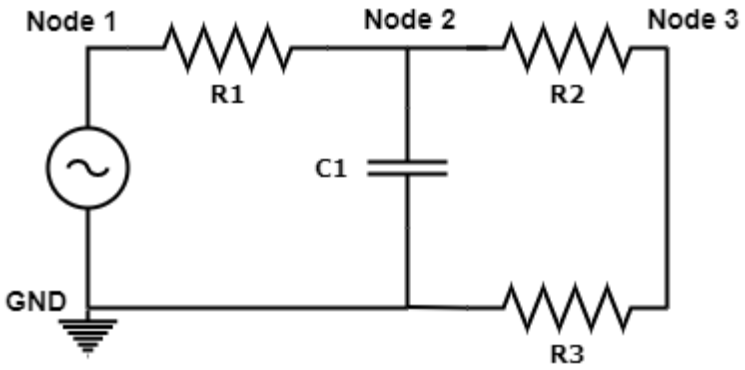

Figure 3–1: Example Circuit for introducing sensitivity

The definition we seen above is called absolute sensitivity. Considering the circuit equation 2.1 in MNA form, it can be derived by:

$$\boldsymbol{D_{C_1}^{x(t)} = \frac{\partial x(t)}{\partial C_1}} \tag{3.2}$$

with $\boldsymbol{x(t)}$ represents the circuit response at the output node. However, the disadvantage of this definition is evident because it is not scaled free, which means comparison for various elements concurrently is difficult. Therefore, we introduce another way of finding sensitivity called relative or normalized sensitivity.

Relative Sensitivity:

$$\boldsymbol{S_{C_1}^{x(t)} = \frac{\partial \ln x(t)}{\partial \ln C_1}} \tag{3.3}$$

It can be represented in term of absolute sensitivity:

$$\boldsymbol{S_{C_1}^{x(t)} = \frac{\partial \ln x(t)}{\partial \ln C_1} = \frac{C_1}{x(t)} \frac{\partial x(t)}{\partial C_1} = \frac{C_1}{x(t)} D_{C_1}^{x(t)}} \tag{3.4}$$

Generally, relative sensitivity will be chosen over absolute because it is related to a unit percentage change in parameters' value, and typically small values of electronic elements like capacitor and inductor used in circuits. Meanwhile, it is also dimensionless and we can readily compare different elements at the same time.

3.2 Three Approaches to Deriving Sensitivity in the Time Domain

There are three approaches to calculate sensitivity. For later demonstration, let us use the example circuit in Figure 3–1 and pick node 2 for analysis. To begin with, considering the system equation 2.1 and denoting

$$\boldsymbol{S} = \frac{\boldsymbol{\partial x_2}}{\boldsymbol{\partial \xi}}$$

as sensitivity output.

3.2.1 Perturbation Approach

Perturbation is a brute force way to derive sensitivity but also has the highest computational cost among all three methods. Firstly, we need to approximate the exact solution $\vec{\boldsymbol{x}}(\boldsymbol{t})$ in 2.1 at one time point, e.g. for circuit in Figure 3–1, we have

$$\vec{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \end{bmatrix} \tag{3.5}$$

Then we add numerical variation into 2.1 based on random variable set $\vec{\boldsymbol{\xi}}$, i.e. 2.9, and solve the new MNA equation again for $\vec{\boldsymbol{x}}(\vec{\boldsymbol{\xi}})$. Upon having both $\vec{\boldsymbol{x}}$ and $\vec{\boldsymbol{x}}(\vec{\boldsymbol{\xi}})$,

the corresponding sensitivity at node 2 with respect to $\boldsymbol{C_1}$ can be calculated by

$$\boldsymbol{S} = \frac{\boldsymbol{x_2}(\vec{\boldsymbol{\xi}}) - \boldsymbol{x_2}}{\boldsymbol{\xi}_{C_1}} \tag{3.6}$$

Distinctly, this method requires solving system equations two times and computes the sensitivity one by one at each node for each element. This process is very time-consuming, and it scales up promptly with the growth of circuit complexity.

3.2.2 Differentiation Approach

Differentiation [3] is an improved method compared to perturbation in terms of efficiency and time complexity. Specifically, the sensitivity of all nodes with respect to one element only requires to simulate system once. Different from perturbation, we approximate $\vec{\boldsymbol{x}}(\vec{\boldsymbol{\xi}})$ with a random variable set $\vec{\boldsymbol{\xi}} = [\boldsymbol{\xi_1}, \boldsymbol{\xi_2}, \cdots, \boldsymbol{\xi}_i]$ fitted into MNA equation 2.9 as the first step. Meanwhile, we compute the derivative form of 2.9. In this case, the derivative form we have is

$$\frac{\boldsymbol{\partial}}{\boldsymbol{\partial \xi}_i}\left[\boldsymbol{G}(\vec{\boldsymbol{\xi}})\vec{\boldsymbol{x}}\right] + \frac{\boldsymbol{\partial}}{\boldsymbol{\partial \xi}_i}\left[\boldsymbol{C}(\vec{\boldsymbol{\xi}})\dot{\vec{\boldsymbol{x}}}\right] + \frac{\boldsymbol{\partial}}{\boldsymbol{\partial \xi}_i}\left[\boldsymbol{f}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{\xi}})\right] = \boldsymbol{0} \tag{3.7}$$

Then the sensitivity at all nodes with respect to one random parameter is computed by evaluating $\frac{\boldsymbol{\partial}\vec{\boldsymbol{x}}}{\boldsymbol{\partial \xi}_i}$. We repeat this process for each $\boldsymbol{\xi}_i$ in the $\vec{\boldsymbol{\xi}}$ set to compute all the sensitivities at one time point.

Overall, even though we need to solve the system equation twice, one for original form and the other for derivative form, the LU decomposition applied to solve for $\vec{\boldsymbol{x}}$ can be inherited during the second computation for sensitivity, which hugely

decreases the CPU cost.

3.2.3 Adjoint Approach

Adjoint Approach [22] aims to create the adjoint network and derives the sensitivity of one output with respect to all random parameters at a time. This method offers advantages on computational speed, as well as accuracy. However, this method is not used in this thesis due to the unsolved difficulties in computing sensitivities in the time domain, especially for nonlinear circuits. Details can be found in [19]. The following part briefly introduces the adjoint network in the frequency domain as a reference.

Considering a circuit equation in the frequency domain as

$$\boldsymbol{GX} + \boldsymbol{sCX} = \boldsymbol{B} \Rightarrow \boldsymbol{AX} = \boldsymbol{B} \tag{3.8}$$

we start by approximating the circuit response $\boldsymbol{X}$ with the presence of random variable $\vec{\boldsymbol{\xi}}$. The difference is we denotes $\boldsymbol{d}$ as an index vector, such that the entry of which has a value of 1 indicating the corresponding output node to be analyzed, e.g.

$$\boldsymbol{V_2} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{V_1} \\ \boldsymbol{V_2} \\ \boldsymbol{V_3} \end{bmatrix} = \boldsymbol{d}^t \boldsymbol{X} \tag{3.9}$$

In this case, the adjoint network is defined as 3.10. The process to construct 3.10 is shown below.

$$\boldsymbol{A}^t \boldsymbol{X}^a = -\boldsymbol{d} \tag{3.10}$$

- Step one: Based on the definition of sensitivity incorporated with index vector $\boldsymbol{d}$, we have

$$\frac{\partial \boldsymbol{V_2}}{\partial \boldsymbol{\xi}} = \boldsymbol{d}^t \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}} \tag{3.11}$$

and by substituting $\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}}$ with equation 3.12,

$$\boldsymbol{A}\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}} = -\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{\xi}}\boldsymbol{X} \tag{3.12}$$

3.11 can be written as

$$\frac{\partial \boldsymbol{V_2}}{\partial \boldsymbol{\xi}} = -\boldsymbol{d}^t \boldsymbol{A}^{-1}\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{\xi}}\boldsymbol{X} \tag{3.13}$$

- Step two: Given the following notation, we can readily observe the adjoint network shown in 3.10.

$$(\boldsymbol{X}^a)^t = -\boldsymbol{d}^t \boldsymbol{A}^{-1} \Rightarrow (\boldsymbol{X}^a)^t \boldsymbol{A} = -\boldsymbol{d}^t \tag{3.14}$$

In summary, we first solve the MNA equation 3.8 for $\boldsymbol{X}$ and then $\boldsymbol{X}^a$ from adjoint network 3.10. The resulted sensitivity is computed by 3.15.

$$\frac{\partial \boldsymbol{V_2}}{\partial \boldsymbol{\xi}} = (\boldsymbol{X}^a)^t \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{\xi}} \tag{3.15}$$

3.2.4 Comparison between Three Approaches

In this thesis, the proposed method SSCM uses a differentiation approach to calculating sensitivity due to its significant advantage on the computational cost compared to the perturbation method because the obtained upper and lower matrices by LU decomposition in circuit simulation step can be reused when solving for

sensitivity. In other words, we need less computational efforts to obtain sensitivity with the differentiation approach. Details will be covered in Chapter 4. As for perturbation method, it requires extra Lu decomposition and forward or backward substitution to find the sensitivity, leading to large CPU cost. Moreover, it is hard to determine the value for randomly parameters $\boldsymbol{\xi}$ since electronic elements generally have a quite small value which indeed differs from each other. Any minute error will lead to enormous variation in resulted sensitivity.

CHAPTER 4
Sensitivity Integrated Stochastic Collocation Solver

This chapter presents the implementation of the proposed sensitivity integrated stochastic collocation method(SSCM), for the variability analysis of circuits in the time domain. This method extends the original SCM with sensitivity to construct a different deterministic system for computing polynomial chaos coefficients. Precisely, because of integrating the sensitivity to the new system, our method is capable of using only a small portion of the generated quadrature points while SCM requires all of them. Thus, the numerical process can be hugely accelerated.

4.1 Starting Point of Sensitivity Integrated Stochastic Collocation Method

Our proposed method computes the polynomial chaos coefficients by a new deterministic system. Due to the similarities between SCM and SSCM in terms of constructing the deterministic system, we start with a demonstration with system equation 2.30 used in SCM and extending to our method. The key to the construction of the deterministic system is generating the sampling points with the truncation scheme applied to. The following section elaborates on the approach to randomly generating truncated sampling points of interest. It should be noted that the following method is outlined on the basis of multi-variate cases.

4.1.1 Point Space Selection

First of all, we need to select the testing sample for our method as input. This thesis benefits from the point selection method from Dr. Zhang's paper [23]. The method has two main steps. Firstly, assume $\boldsymbol{n = p + 1}$ while $\boldsymbol{p}$ is the highest total order of basis function $\boldsymbol{\phi(\vec{\xi})}$, and then apply Gaussian Quadrature rule [11] to generate quadrature points for each random variable $\boldsymbol{\xi_i}$, together with their Gaussian weights. As a result, we have $\boldsymbol{(p+1)^m}$ sampling points as a candidate pool assuming

$$\vec{\boldsymbol{\xi}} = [\vec{\boldsymbol{\xi^1}}, \vec{\boldsymbol{\xi^2}}, \cdots, \vec{\boldsymbol{\xi^m}}]$$

. Secondly, we apply filtration to those points to choose only $\boldsymbol{K}$ points from them. To select $\boldsymbol{K}$ points from the pool, Dr.Zhang first sorts all the weights in descending order and picks the point having the largest weight as the first sample point. Afterwards, a vector space with $\boldsymbol{i}$ sample points selected is constructed as:

$$\boldsymbol{V_{i-1}} = \begin{bmatrix} \boldsymbol{\phi(\vec{\xi^1})} \\ \boldsymbol{\phi(\vec{\xi^2})} \\ \vdots \\ \boldsymbol{\phi(\vec{\xi^i})} \end{bmatrix} \tag{4.1}$$

The rest $\boldsymbol{(K - i)}$ points are selected based on the scheme that only if it has large component orthogonal to this vector until a maximum $\boldsymbol{K}$ points are achieved with a threshold scalar $\boldsymbol{\beta}$. However, an alternative solution to this method is proposed in [23] as generating and saving the candidate points becomes more and more expensive with the increment of parameters' dimension $\boldsymbol{m}$. Instead, the proposed new solution only generate weight and corresponding index as a reference and then create

the sample points without storing it at the first place. Only points that meet the orthogonal condition will be stored as our final output. This approach significantly improves efficiency and avoids the waste in space. The difference in our thesis is that we define $\boldsymbol{K}$ based on the total degree method mentioned in section 2.6, which leads to $\boldsymbol{K} = \frac{(p+m)!}{p!m!}$ by the end.

4.1.2 System Construction

In order to formulate both sides of equation 2.30, first, we simulate circuit equation 2.9 with $\boldsymbol{K}$ selected sample points for $\vec{\boldsymbol{x}}$ from the previous section. This gives us

$$\vec{\boldsymbol{x}} = \boldsymbol{V} = \begin{bmatrix} \boldsymbol{V_1^o} \\ \boldsymbol{V_2^o} \\ \vdots \\ \boldsymbol{V_m^o} \end{bmatrix} \tag{4.2}$$

where $\boldsymbol{o}$ denotes the output node if the voltage is analyzed. Afterwards, we need to set up the matrix $\boldsymbol{H}$ that containing the polynomial chaos basis function. To start with, let us consider the example in Table 2–2 and represents the first five terms of Hermite polynomial basis functions as a product of two matrices in 4.3.

$$\begin{bmatrix} \boldsymbol{H_0} \\ \boldsymbol{H_1} \\ \boldsymbol{H_2} \\ \boldsymbol{H_3} \\ \boldsymbol{H_4} \end{bmatrix} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{-1} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{-3} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{3} & \boldsymbol{0} & \boldsymbol{-6} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi^0} \\ \boldsymbol{\xi^1} \\ \boldsymbol{\xi^2} \\ \boldsymbol{\xi^3} \\ \boldsymbol{\xi^4} \end{bmatrix} \tag{4.3}$$

In the second place, we accumulatively multiply the Hermite basis functions based on the multi-variate case in 2.23 and apply the truncation scheme in section 2.6 to which at the same time. As a result, we have

$$\begin{bmatrix} \phi_0(\vec{\xi^i}) \\ \phi_1(\vec{\xi^i}) \\ \vdots \\ \phi_{m-1}(\vec{\xi^i}) \end{bmatrix} = \begin{bmatrix} H_0 H_0 H_0 \\ H_0 H_0 H_1 \\ \vdots \\ H_a H_b H_c \end{bmatrix} \tag{4.4}$$

with a highest total order p, such that $(a+b+c) \leq p$ and there are three variables in the set $\vec{\xi^i}$. Finally, we perform the same pattern to the rest of $\vec{\xi^i}$ in $\vec{\xi}$ to obtain the polynomial chaos basis function at each sample point and linearly combine all the deterministic equations to form H. A detailed structure of this linear deterministic system is given in 4.5, where $\vec{\xi^m}$ denotes the corresponding random variable along one direction. Besides, k is the number of terms selected after applying the truncation scheme. Upon obtaining the circuit responses V, polynomial chaos coefficients can be computed with either a direct or an iterative solver.

$$\begin{bmatrix} \phi_0(\xi_1^1) & \phi_1(\xi_2^1) & \cdots & \phi_{k-1}(\xi_k^1) \\ \phi_0(\xi_1^2) & \phi_1(\xi_2^2) & \cdots & \phi_{k-1}(\xi_k^2) \\ \phi_0(\xi_1^3) & \phi_1(\xi_2^3) & \cdots & \phi_{k-1}(\xi_k^3) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(\xi_1^m) & \phi_1(\xi_2^m) & \cdots & \phi_{k-1}(\xi_k^m) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_3 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} V_1^o \\ V_2^o \\ V_3^o \\ \vdots \\ V_m^o \end{bmatrix} \tag{4.5}$$

SCM has a considerable advantage in terms of computational cost compared to the MCM since polynomial chaos is used to model stochastic process and thus fewer

sample points are required with the deterministic system. However, as we can see in 4.5, the cost of constructing and solving this system increases expeditiously as the complexity of circuits and number of random variables growing.

4.2 Implementation of Sensitivity Integrated Stochastic Collocation Method

The SSCM proposed by this thesis progressively employs only a small portion of the quadrature points to set up our advanced deterministic system, while SCM requires all of them. Thus, our method provides extra speedup in terms of the solution in the time domain. The key to the method is done by integrating sensitivities analyzed at the output node with respect to each random variable by the differentiation approach in section 3.1. The deterministic system of our proposed SSCM is shown below

$$\bar{\boldsymbol{H}}(\phi_{\boldsymbol{k}}(\vec{\boldsymbol{\xi^m}}))\boldsymbol{A} = \bar{\boldsymbol{V}} \tag{4.6}$$

where $\bar{\boldsymbol{H}} \in \mathbb{R}^{\boldsymbol{m(k+1)\times k}}$ denotes the new deterministic system with $\boldsymbol{k}$ random variables in each set $\vec{\boldsymbol{\xi^i}}$, $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{k}}$ represents the polynomial chaos coefficients, and $\bar{\boldsymbol{V}} \in \mathbb{R}^{\boldsymbol{m(k+1)}}$ stores the circuit responses and sensitivities at desired output node with respect to different parameters.

To construct the deterministic system in 4.6, we start with $\bar{\boldsymbol{V}}$ first as circuit response $\vec{\boldsymbol{x}}$ and corresponding sensitivity at one time point can be computed within one iteration. However, approaches to calculating sensitivities vary in linear and nonlinear circuits, as presented in the following section.

Linear Case : To begin with, we deduce the derivative of system equation 2.3 with respect to different random variables. One aspect to note is the type of element

that containing a random variable leads to inconsistent forms of derivative functions. Specifically, the derivative equations for computing sensitivity differ in $\boldsymbol{G}$ and $\boldsymbol{C}$ matrices as far as MNA representation is concerned.

- Case G: Derivative of memory device

$$\frac{\boldsymbol{\partial G}}{\boldsymbol{\partial \xi_k^m}}\vec{\boldsymbol{x}}_{\boldsymbol{n+1}} + \boldsymbol{G}\frac{\boldsymbol{\partial \vec{x}_{n+1}}}{\boldsymbol{\partial \xi_k^m}} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\left(\frac{\boldsymbol{\partial \vec{x}_{n+1}}}{\boldsymbol{\partial \xi_k^m}} - \frac{\boldsymbol{\partial \vec{x}_n}}{\boldsymbol{\partial \xi_k^m}}\right) = \boldsymbol{0} \tag{4.7}$$

  Since $\boldsymbol{G} = \boldsymbol{G_\mu} + \boldsymbol{\xi_k^m G_\sigma}$, after simple organization, we have

$$\left(\boldsymbol{G} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\right)\frac{\boldsymbol{\partial \vec{x}_{n+1}}}{\boldsymbol{\partial \xi_k^m}} = \frac{\boldsymbol{C}}{\boldsymbol{h}}\frac{\boldsymbol{\partial \vec{x}_n}}{\boldsymbol{\partial \xi_k^m}} - \boldsymbol{G_\sigma}\vec{\boldsymbol{x}}_{\boldsymbol{n+1}} \tag{4.8}$$

- Case C: Derivative of memoryless device

$$\boldsymbol{G}\frac{\boldsymbol{\partial \vec{x}_{n+1}}}{\boldsymbol{\partial \xi_k^m}} + \frac{\boldsymbol{\partial C}}{\boldsymbol{\partial \xi_k^m}}\frac{\boldsymbol{1}}{\boldsymbol{h}}(\vec{\boldsymbol{x}}_{\boldsymbol{n+1}} - \vec{\boldsymbol{x}}_{\boldsymbol{n}}) + \frac{\boldsymbol{C}}{\boldsymbol{h}}\left(\frac{\boldsymbol{\partial \vec{x}_{n+1}}}{\boldsymbol{\partial \xi_k^m}} - \frac{\boldsymbol{\partial \vec{x}_n}}{\boldsymbol{\partial \xi_k^m}}\right) = \boldsymbol{0} \tag{4.9}$$

  Similar to $\boldsymbol{G}$, we rewrite it as

$$\left(\boldsymbol{G} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\right)\frac{\boldsymbol{\partial \vec{x}_{n+1}}}{\boldsymbol{\partial \xi_k^m}} = \frac{\boldsymbol{C}}{\boldsymbol{h}}\frac{\boldsymbol{\partial \vec{x}_n}}{\boldsymbol{\partial \xi_k^m}} - \frac{\boldsymbol{C_\sigma}}{\boldsymbol{h}}(\vec{\boldsymbol{x}}_{\boldsymbol{n+1}} - \vec{\boldsymbol{x}}_{\boldsymbol{n}}) \tag{4.10}$$

Advantageously, $(\boldsymbol{G} + \frac{\boldsymbol{C}}{\boldsymbol{h}})$ has already been decomposed by LU in the previous regression step for calculating circuit response. This means that the upper matrix $\boldsymbol{U}$ and lower matrix $\boldsymbol{L}$ can be directly reused in 4.10, therefore significantly avoids the additional increase in the time complexity of computing sensitivities.

Nonlinear Case : Likewise, the derivative of nonlinear MNA equation 2.6 embedded with Backward Euler method varies on different types of elements. Additionally, its solution depends on whether the nonlinear element is consolidated with a random variable or not. For solely revealing the difference with the presence of

$\boldsymbol{f}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{\xi}})$, we assume $\boldsymbol{G}$ and $\boldsymbol{C}$ are not a function subject to $\vec{\boldsymbol{\xi}}$ here. The derivative form of 2.6 is

$$\boldsymbol{G}\frac{\partial \vec{\boldsymbol{x}}_{n+1}}{\partial \boldsymbol{\xi}_k^m} + \frac{\boldsymbol{C}}{\boldsymbol{h}}\left(\frac{\partial \vec{\boldsymbol{x}}_{n+1}}{\partial \boldsymbol{\xi}_k^m} - \frac{\partial \vec{\boldsymbol{x}}_n}{\partial \boldsymbol{\xi}_k^m}\right) + \frac{\partial \boldsymbol{f}}{\partial \vec{\boldsymbol{x}}_{n+1}}\frac{\partial \vec{\boldsymbol{x}}_{n+1}}{\partial \boldsymbol{\xi}_k^m} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\xi}_k^m} = \boldsymbol{0} \tag{4.11}$$

$$\Downarrow$$

$$\left(\boldsymbol{G} + \frac{\boldsymbol{C}}{\boldsymbol{h}} + \boldsymbol{J}(\vec{\boldsymbol{x}}_{n+1}, \boldsymbol{\xi}_k^m)\right)\frac{\partial \vec{\boldsymbol{x}}_{n+1}}{\partial \boldsymbol{\xi}_k^m} = \frac{\boldsymbol{C}}{\boldsymbol{h}}\frac{\partial \vec{\boldsymbol{x}}_n}{\partial \boldsymbol{\xi}_k^m} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\xi}_k^m} \tag{4.12}$$

where $\boldsymbol{J}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{\xi}})$ is the Jacobian matrix of $\boldsymbol{f}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{\xi}})$.

After concatenating the circuit response at output node and its sensitivities, $\bar{\boldsymbol{V}}$ is formed as:

$$\begin{bmatrix} \boldsymbol{V_1^o} \\ \frac{\partial \boldsymbol{V_1^o}}{\partial \boldsymbol{\xi_1^1}} \\ \frac{\partial \boldsymbol{V_1^o}}{\partial \boldsymbol{\xi_2^1}} \\ \vdots \\ \frac{\partial \boldsymbol{V_1^o}}{\partial \boldsymbol{\xi_k^1}} \\ \boldsymbol{V_2} \\ \frac{\partial \boldsymbol{V_2^o}}{\partial \boldsymbol{\xi_1^2}} \\ \vdots \\ \boldsymbol{V_m^o} \\ \vdots \end{bmatrix} \tag{4.13}$$

The next step is to build the linearly combined matrix $\bar{\boldsymbol{H}}$. Accordingly, we need to integrate the polynomial expansion of resulted sensitivities in $\bar{\boldsymbol{V}}$. Based on 2.15,

we have its variation on derivative form.

$$\frac{\partial \boldsymbol{V}}{\partial \vec{\boldsymbol{\xi}}} = \sum_{\vec{\alpha}} \boldsymbol{A}_{\vec{\alpha}} \frac{\partial \phi_{\vec{\alpha}}(\vec{\boldsymbol{\xi}})}{\partial \vec{\boldsymbol{\xi}}} \tag{4.14}$$

The process of generating the derivative polynomial chaos basis function for modelling sensitivity is the same as what we have done in the previous section. Afterwards, we concatenate the polynomial chaos basis function with their derivative form for approximating sensitivities. Specifically, the functions after a row of Hermite polynomial chaos expansion at one sample point until the next one is the derivative form of current basis functions with respect to each random variable in the set $\vec{\boldsymbol{\xi}^{\boldsymbol{m}}}$. Thus, this leads to $\boldsymbol{m(k+1)}$ linearly combined equations in total. Equation 4.15 illustrates the new system with sensitivity analysis involved.

$$\begin{bmatrix}
\boldsymbol{\phi_0(\xi_1^1)} & \boldsymbol{\phi_1(\xi_2^1)} & \cdots & \boldsymbol{\phi_{k-1}(\xi_k^1)} \\
\frac{\partial \boldsymbol{\phi_0(\xi_1^1)}}{\partial \boldsymbol{\xi_1^1}} & \frac{\partial \boldsymbol{\phi_1(\xi_2^1)}}{\partial \boldsymbol{\xi_1^1}} & \cdots & \frac{\partial \boldsymbol{\phi_{k-1}(\xi_k^1)}}{\partial \boldsymbol{\xi_1^1}} \\
\frac{\partial \boldsymbol{\phi_0(\xi_1^1)}}{\partial \boldsymbol{\xi_2^1}} & \frac{\partial \boldsymbol{\phi_1(\xi_2^1)}}{\partial \boldsymbol{\xi_2^1}} & \cdots & \frac{\partial \boldsymbol{\phi_{k-1}(\xi_k^1)}}{\partial \boldsymbol{\xi_2^1}} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\partial \boldsymbol{\phi_0(\xi_1^1)}}{\partial \boldsymbol{\xi_k^1}} & \frac{\partial \boldsymbol{\phi_1(\xi_2^1)}}{\partial \boldsymbol{\xi_k^1}} & \cdots & \frac{\partial \boldsymbol{\phi_{k-1}(\xi_k^1)}}{\partial \boldsymbol{\xi_k^1}} \\
\boldsymbol{\phi_0(\xi_1^2)} & \boldsymbol{\phi_1(\xi_2^2)} & \cdots & \boldsymbol{\phi_{k-1}(\xi_k^2)} \\
\frac{\partial \boldsymbol{\phi_0(\xi_1^2)}}{\partial \boldsymbol{\xi_1^2}} & \frac{\partial \boldsymbol{\phi_1(\xi_2^2)}}{\partial \boldsymbol{\xi_1^2}} & \cdots & \frac{\partial \boldsymbol{\phi_{k-1}(\xi_k^2)}}{\partial \boldsymbol{\xi_1^2}} \\
\vdots & \vdots & \vdots & \vdots \\
\boldsymbol{\phi_0(\xi_1^m)} & \boldsymbol{\phi_1(\xi_2^m)} & \cdots & \boldsymbol{\phi_{k-1}(\xi_k^m)} \\
\vdots & \vdots & \vdots & \vdots
\end{bmatrix}
\begin{bmatrix} \boldsymbol{a_0} \\ \boldsymbol{a_1} \\ \boldsymbol{a_2} \\ \vdots \\ \boldsymbol{a_k} \end{bmatrix}
=
\begin{bmatrix}
\boldsymbol{V_1} \\
\frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi_1^1}} \\
\frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi_2^1}} \\
\vdots \\
\frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi_k^1}} \\
\boldsymbol{V_2} \\
\frac{\partial \boldsymbol{V_2}}{\partial \boldsymbol{\xi_1^2}} \\
\vdots \\
\boldsymbol{V_m} \\
\vdots
\end{bmatrix} \tag{4.15}$$

Like SCM, the desired polynomial chaos coefficients $\boldsymbol{A}$ can be computed with a direct solver or an iterative solver at each time point.

In summary, the steps to implement SSCM is concluded in Algorithm 1.

Algorithm 1: Stochastic Collocation Method with Sensitivity Analysis

input : List of elements having uncertainties attached; Highest total degree $\boldsymbol{p}$; Output node; Time step $\boldsymbol{h}$ and end point $\boldsymbol{T}$ of transient analysis range

output: Coefficients of Hermite Polynomial Chaos Expansion $\boldsymbol{A}$ at each time point

```
begin
    Choose K points from generated quadrature points pool;
    Construct H matrix;
    for each set of random variables do
        Fit corresponding random variables into MNA equation;
        Initialization of Circuit Simulation;
        while t < T do
            Solving for circuit response at current time point;
            for each element in the element list do
                Compute the derivative form of MNA equation based on the
                 current element type;
                Calculate sensitivities at output node;
            end
            Concatenate circuit response at output node with its sensitivities
             with respect to all random variables at current time point to
             form V̄;
            Compute polynomial chaos coefficients at current time point
             by 4.6;
            Increase t by h for next time point;
        end
    end
end
```

CHAPTER 5
Example Evaluation

This chapter presents several examples to prove the enhancement of the proposed SSCM in terms of time complexity while maintaining the accuracy compared to MCM and SCM. MCM serves as the baseline for validating the accuracy since the time consumption of MCM is prohibitively expensive for transient simulation and thus no need to compare, while SCM is used for comparing time complexity to SSCM. The following examples, both linear and nonlinear circuits, are based on the uni-conductor or multi-conductor transmission line(MTL), which are the fundamental component of high-speed interconnects. In Figure 5–1, an example of a conductor has a length $\boldsymbol{L = 2cm}$ and can be coupled together to form the MTL sharing one reference node. The data of crucial parameters of MTL, Resistance, Inductance, Capacitance and Conductance, denoted as $\boldsymbol{R}$, $\boldsymbol{L}$, $\boldsymbol{C}$ and $\boldsymbol{S}$ respectively, are all defined in terms of per-unit-length(PUL) parameter matrices. Each conductor is constructed with 100 sections of lumped $\boldsymbol{RLCS}$ segmentation.

Notably, in the following examples, uniform time step sizes are used for all three methods as we are seeking for the statistical information of the time domain solution. The time domain response is plotted with 400 time points based on the period of each input source and its time step, which gives a complete period from one rising edge to next with an initial delay interval. Moreover, all three methods have the same 100000 randomized Gaussian distributed testing samples with a mean and standard

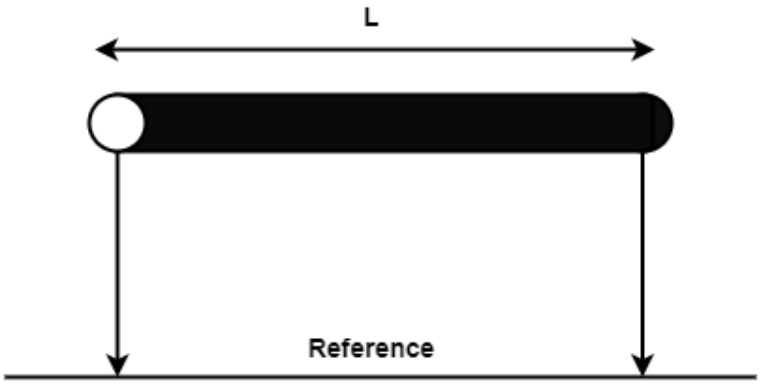

Figure 5–1: Illustration of a conductor with a length of 20cm

deviation of 0 and 1, respectively. MCM uses these testing samples as different sets of random variables $\{\xi\}$ and samples at each of them to obtain statistical information. SCM and SSCM are tested with these 100000 testing samples after obtaining the coefficients.

5.1 Exploration on Linear Circuits

5.1.1 Example 1: 7-Uniconductor Transmission Line Circuit

The linear transmission line circuit in Figure 5–2, also presented in [8], has 7 uni-conductors. The specification of conductors is included in Table 5–1. The signal input in this circuit is an AC pulse wave voltage source with an initial and a peak voltage of 0V and 5V, respectively. Let $\boldsymbol{p}$ denotes the highest total order of each polynomial expansion term and $\boldsymbol{N}$ represents the number of elements that associated with uncertainties, in this particular example, we have $\boldsymbol{p = 3}$ and $\boldsymbol{N = 10}$. As a result, we have in total **286** sets of random variables for SCM based on the total

degree method mentioned in section 2.6 and our SSCM only takes a twenty percent of them, i.e. only **57** sets.

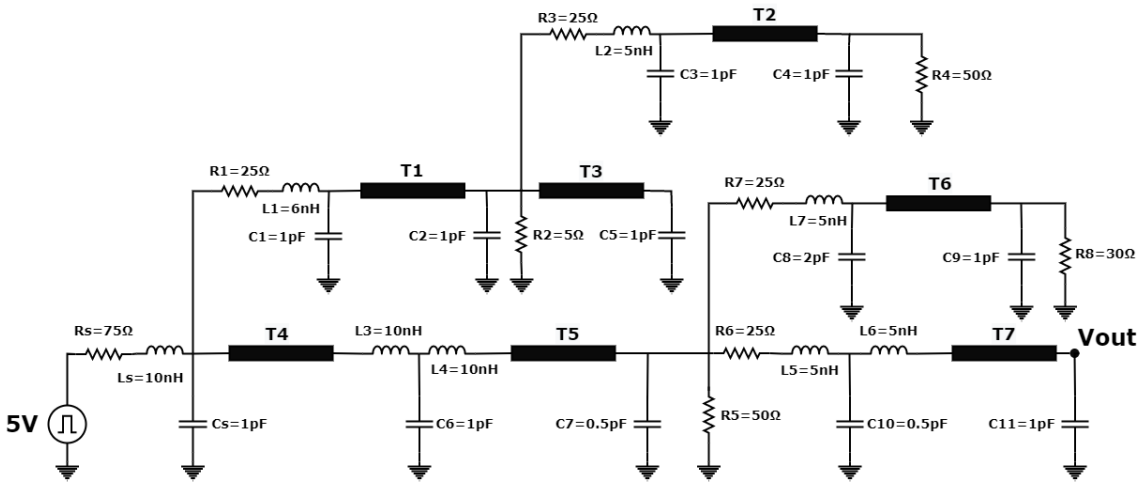

Figure 5–2: Schematic of a seven uni-conductor circuit with a pulse voltage source of 5V

Table 5–1: Specification of the 7 uni-conductor circuit

	Resistance (Ω/m)	Inductance (nH/m)	Capacitance (pF/m)	Conductance $(m\Omega^{-1}/m)$	Length (m)
T1	8	60	120	5	0.03
T2	8	60	100	5	0.03
T2	8	60	100	5	0.03
T2	8	100	100	5	0.05
T2	8	60	120	5	0.03
T2	8	60	100	5	0.04
T2	8	100	150	5	0.02

Before verifying the accuracy of the proposed SSCM, efficiently, we emphasize on comparing the mean, variance and possibility density function between three methods to evaluate the correctness of SSCM. In Figure 5–3, the mean values of these three methods generating by 100000 identical testing samples have nearly the same results. For further comparison, we see in the magnified portion that illustrating the mean value within uniformly distributed time interval from $\mathbf{3.28443}\boldsymbol{ns}$ to $\mathbf{3.28445}\boldsymbol{ns}$, the error of SSCM is less than $\mathbf{2 \times 10^{-5}}$. Similar to mean values, the variance of three methods showing in Figure 5–4 are substantially adjacent to each other, with a divergence less than $\mathbf{2 \times 10^{-8}}$.

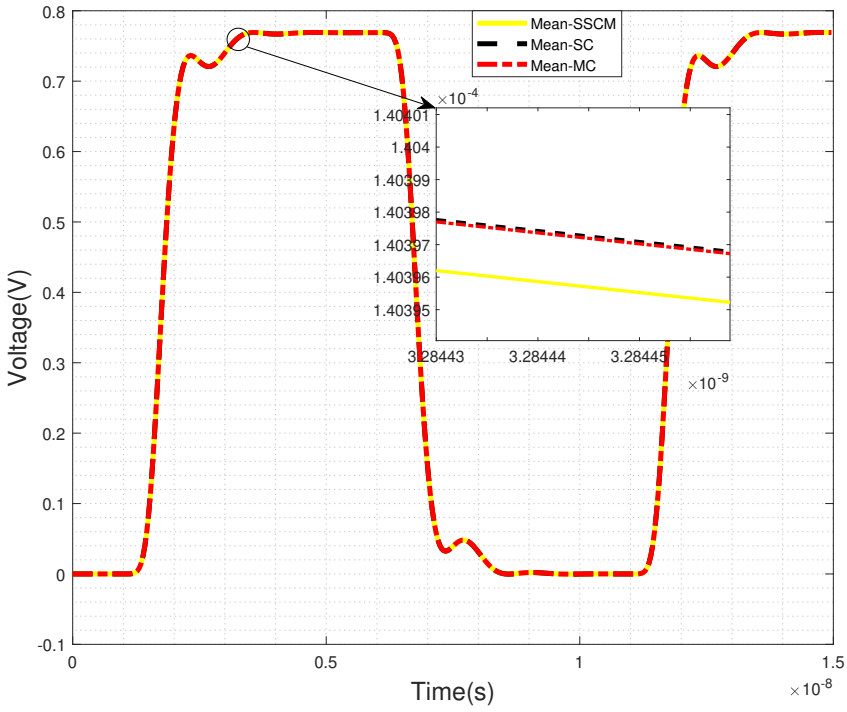

Figure 5–3: Mean value comparison of Example 1 with SSCM, SCM at $\boldsymbol{p} = \mathbf{3}$ and $\boldsymbol{N} = \mathbf{10}$ and MCM

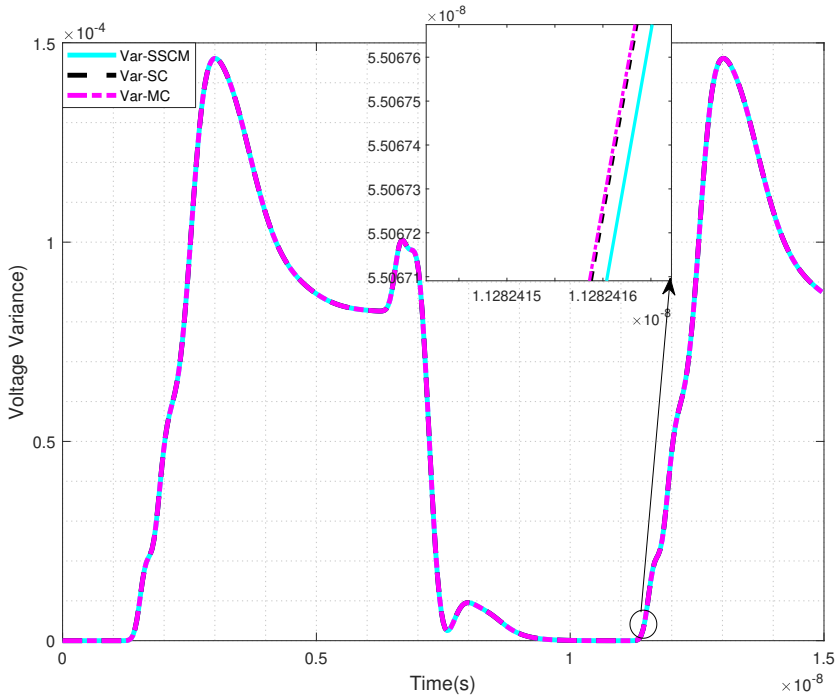

Figure 5–4: Variance comparison of Example 1 with SSCM, SCM at $\boldsymbol{p = 3}$ and $\boldsymbol{N = 10}$ and MCM

Figure 5–5 shows the probability distribution computed at two different time points. Again, the results of MCM and SCM are generated as the baseline. As seen, SSCM has the same observation as the baseline methods, whether at the center or side areas.

In order to further statistically quantify the variance trend, we plot the 3-sigma limits, known as 68–95–99.7 rule in statistics area [16], of the output of SSCM in Figure 5–6. What is observed is that the circuit response all lies within the three standard deviation of the mean, which gives a solid guarantee of the accuracy of our proposed method.

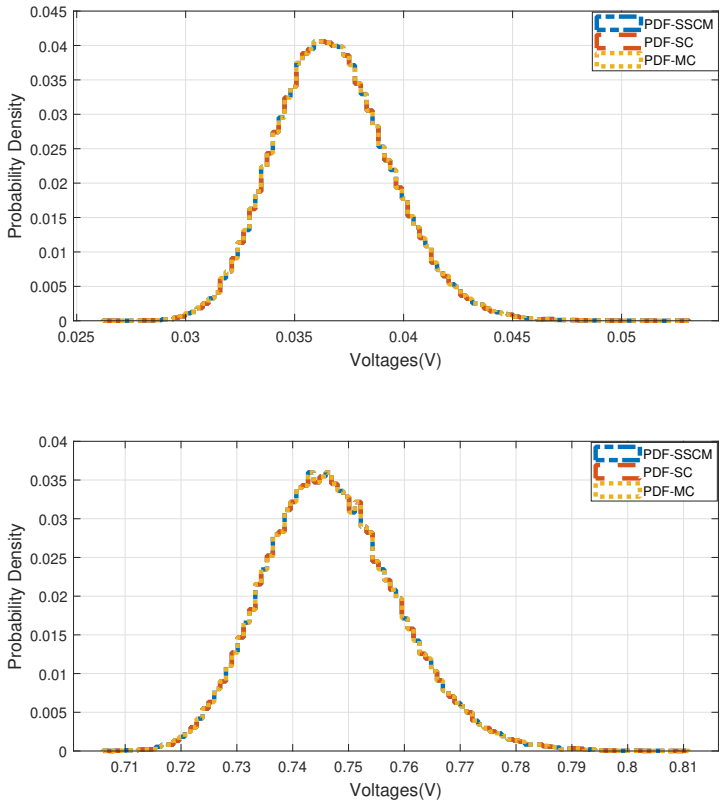

Figure 5–5: Probability density function comparison of Example 1 at time $\mathbf{7.5}\boldsymbol{ns}$ and $\mathbf{13.125}\boldsymbol{ns}$ with SSCM, SCM at $\boldsymbol{p} = \mathbf{3}$ and $\boldsymbol{N} = \mathbf{10}$ and MCM

Apropos of the time consumption, to clarify, all three methods, MCM, SCM and SSCM, are implemented to fit the requirements of the Parallel Computing Tool in MATLAB with numbers of workers of 4 in order to save the time budget for testing since a large number of simulations required in time domain solution, especially for MCM. Moreover, all simulations are performed with a workstation with a processor of 4GHz and 16GB RAM. The CPU time is calculated based on a scheme that: starting from the beginning of the algorithm until the polynomial chaos expansion

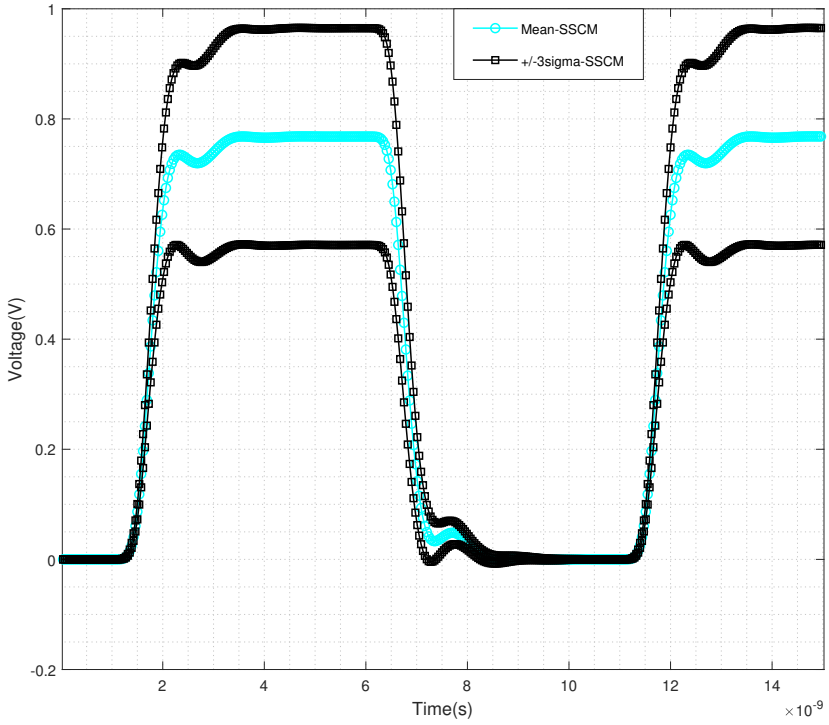

Figure 5–6: 3-Sigma limits of Example 1 with SSCM at $\boldsymbol{p = 3}$ and $\boldsymbol{N = 10}$

coefficients are obtained. With ten random variables in each set of $\{\boldsymbol{\xi}\}$, SCM essentially requires 10 mins and 28 secs to finish running while our SSCM only takes 2 min and 8 seconds and maintains the same accuracy simultaneously. Thus, SSCM has almost 5 times the advantage in terms of time complexity compared to SCM because it is capable of using fewer points but meanwhile ensures the same accuracy.

5.1.2 Example 2: One 8 Coupled MTL Circuit

This example considers the variability analysis with proposed SSCM of the circuit shown in 5–7, in which a MTL with 8 conductors coupled is presented. The

whole circuit is excited by 4 pulse wave voltage sources with a peak of 5V and a period of 10ns. In like manner, we choose the highest total order $\boldsymbol{p}$ of **3** and randomly picks **12** elements associated with uncertainties. This leads to **455** sets of random variables for SCM and **91** for SSCM competitively.

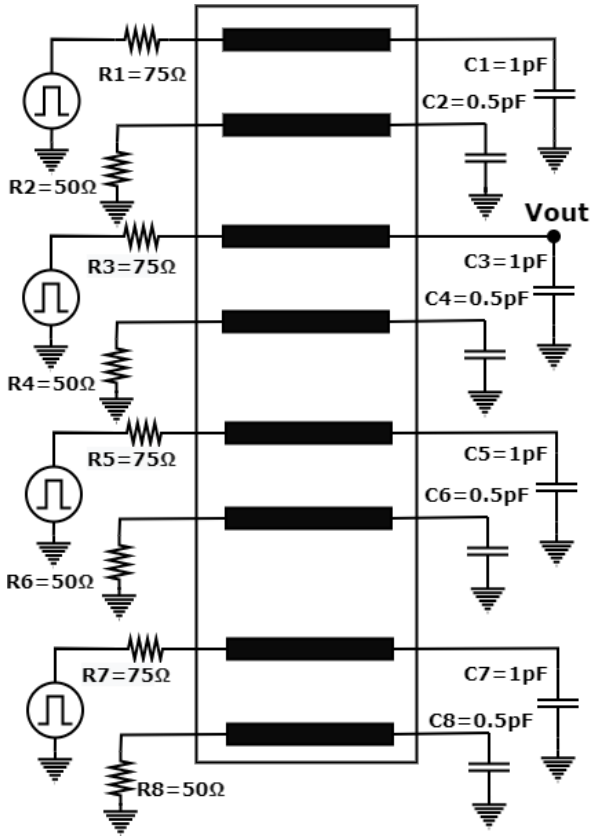

Figure 5–7: Schematic of an eight-coupled MTL circuit with 4 pulse voltage sources of 5V

Like the previous example, we start by comparing the mean, variance and possibility density function of all three methods. Figure 5–8 shows the comparison between mean values of MCM, SCM and SSCM. Evidently, in the magnified plot, MCM and SCM have almost the same results, and the simulated error of SSCM to

MCM at time $\mathbf{10.4063}\boldsymbol{ns}$ is less than $\mathbf{1 \times 10^{-5}}$. The variance comparison is illustrated in Figure 5–9. As expected, all three methods have the same trend variation, and the difference between SSCM and the other two are within $\mathbf{5 \times 10^{-7}}$, which can be neglected.

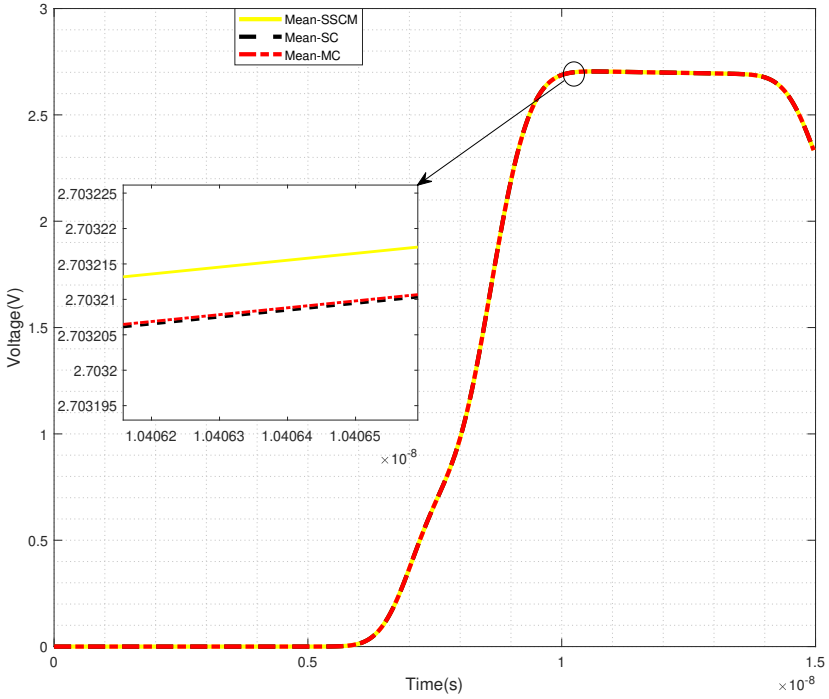

Figure 5–8: Mean value comparison of Example 2 with SSCM, SCM at $\boldsymbol{p} = \mathbf{3}$ and $\boldsymbol{N} = \mathbf{12}$ and MCM

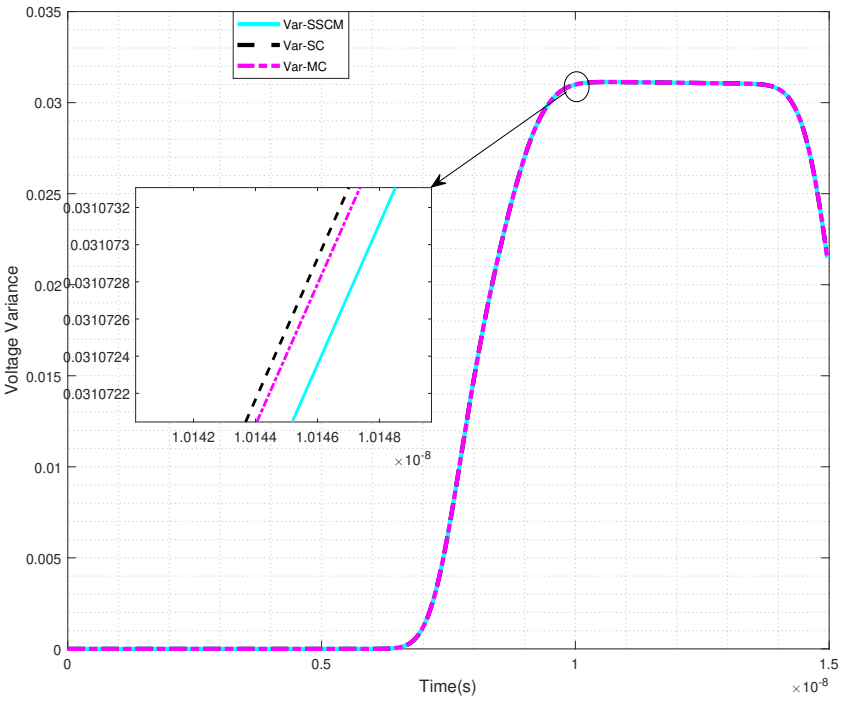

Figure 5–9: Variance comparison of Example 2 with SSCM, SCM at $\boldsymbol{p} = \mathbf{3}$ and $\boldsymbol{N} = \mathbf{12}$ and MCM

Figure 5–10 illustrates the probability density function at two specific time points as observed. At the time $\mathbf{7.5}\boldsymbol{ns}$ and $\mathbf{13.125}\boldsymbol{ns}$, the values of our SSCM lies in the same higher or lower probabilities regions as MCM and SCM. Correspondingly, a plot of 3 sigma limits is evaluated for further verification. Figure 5–11 shows the 3-sigma band of this example.

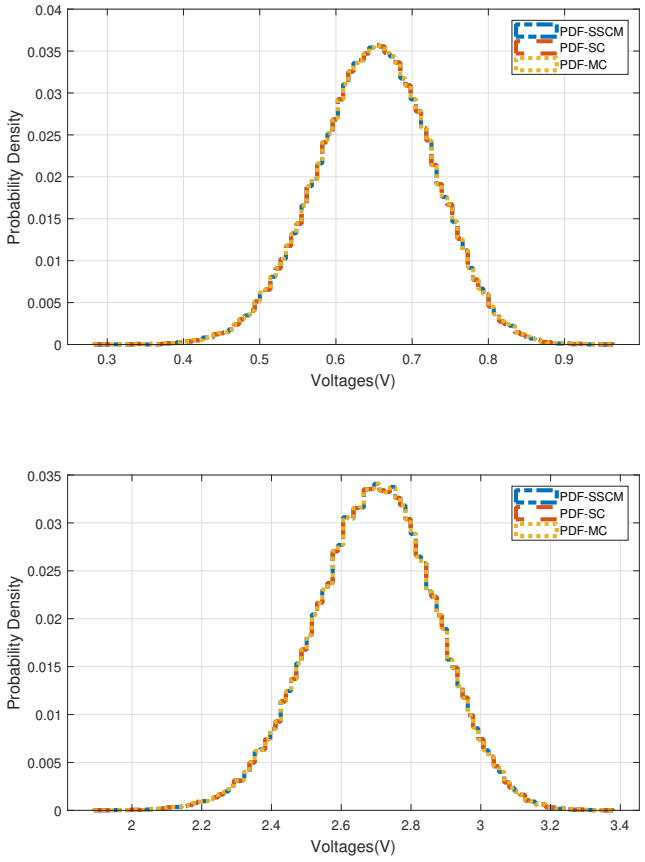

Figure 5–10: Probability density function comparison of Example 2 at time $\mathbf{7.5}\boldsymbol{ns}$ and $\mathbf{13.125}\boldsymbol{ns}$ with SSCM, SCM at $\boldsymbol{p} = \mathbf{3}$ and $\boldsymbol{N} = \mathbf{12}$ and MCM

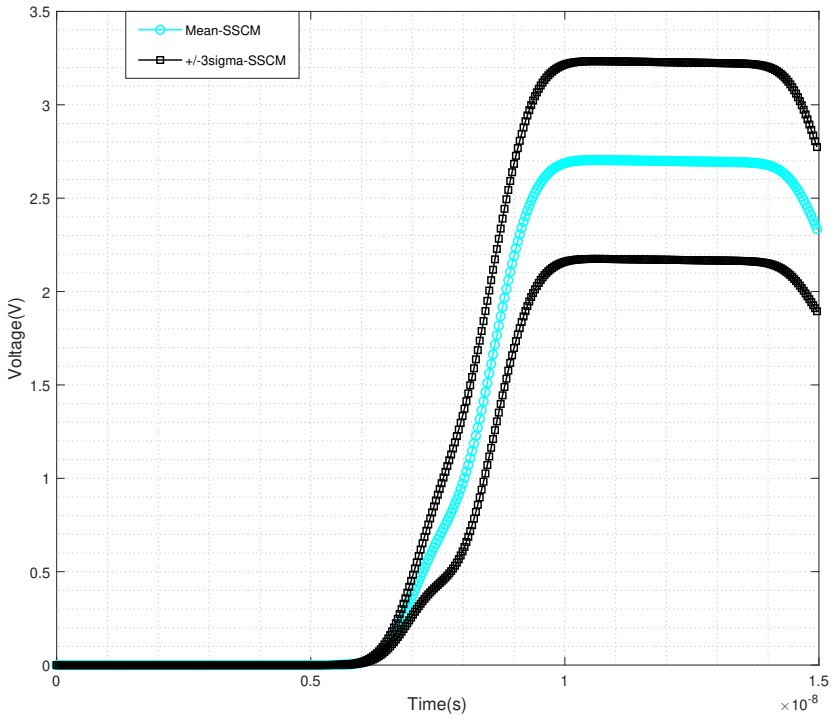

Figure 5–11: 3-Sigma limits of Example 2 with SSCM at $\boldsymbol{p = 3}$ and $\boldsymbol{N = 12}$

As with the first linear example, the circuit is evaluated with 100000 test samples but requires a longer time to converge than that taken for the previous one since we increase the number of random variables and the complexity of the circuit itself. SCM takes 52 mins and 15 secs to obtain the voltage outputs at 400 fixed time points. Whereas, the time for SSCM to acquire the results with the same accuracy is 12 mins and 20 secs. In other words, SSCM is 4.3 times faster than SCM in terms of this case.

5.2 Exploration on Nonlinear Circuits

5.2.1 Example 3: Two 2 Coupled MTL Circuit with diode

The nonlinear circuit in Figure 5–12 mainly consists of two 2-coupled multi-conductor transmission line, and the left one is attached with two diodes as the nonlinear part. Analogously, we use a 5V pulse wave voltage source to excite the circuit and evaluate the response at the output node. Following the same pattern, the number of collocation points used by SCM is **165** based on the number of random variables $\boldsymbol{N}$ of **8** and highest total order $\boldsymbol{p}$ of 3, while SSCM only takes **33** sets of points, which is one-fifth of that.

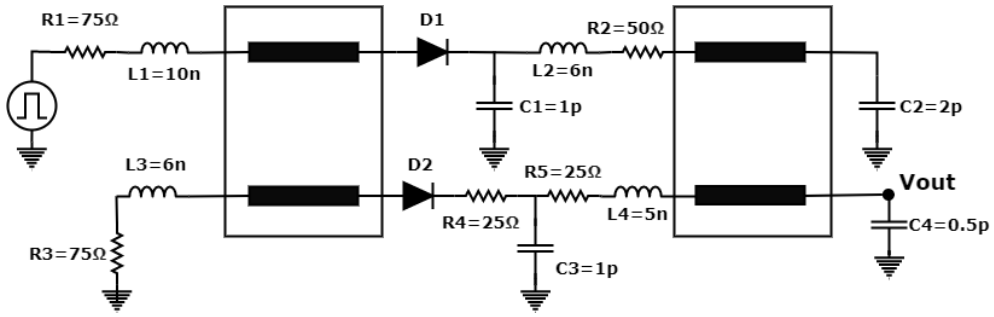

Figure 5–12: Schematic of a two-coupled MTL circuit with a pulse voltage source of 5V

To begin with, we compare the mean, variance and possibility density function of the three methods as the previous examples. The mean value comparison in Figure 5–13 shows that SSCM has nearly the same trend variation as the other two under the equivalent numerical level. Additionally, in the magnified plot, the error between SSCM and MCM has a value of less than $\mathbf{2 \times 10^{-6}}$ at a randomly chosen time point. Similarly, the difference of variance between SSCM and MCM is within acceptable limits, as illustrated in Figure 5–14.

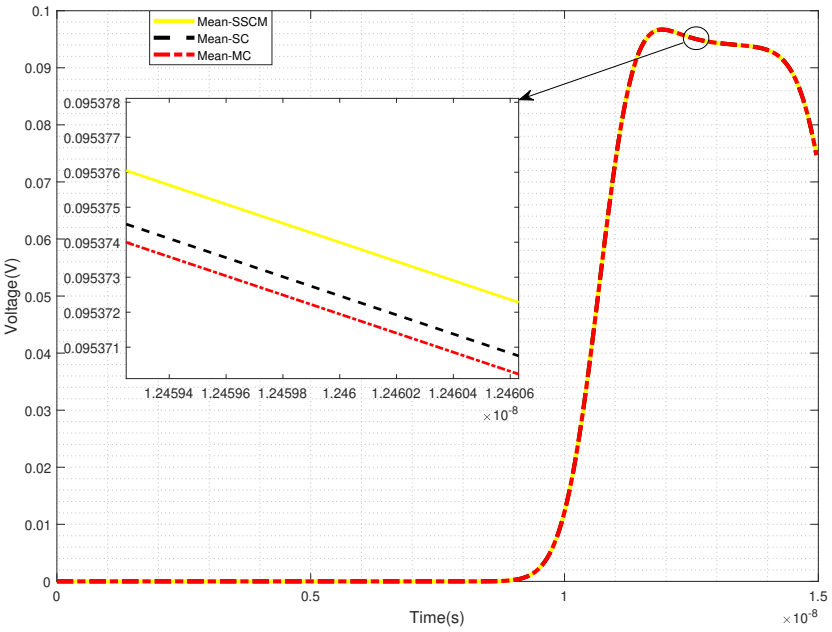

Figure 5–13: Mean value comparison of Example 3 with SSCM, SCM at $\boldsymbol{p} = \mathbf{3}$ and $\boldsymbol{N} = \mathbf{8}$ and MCM

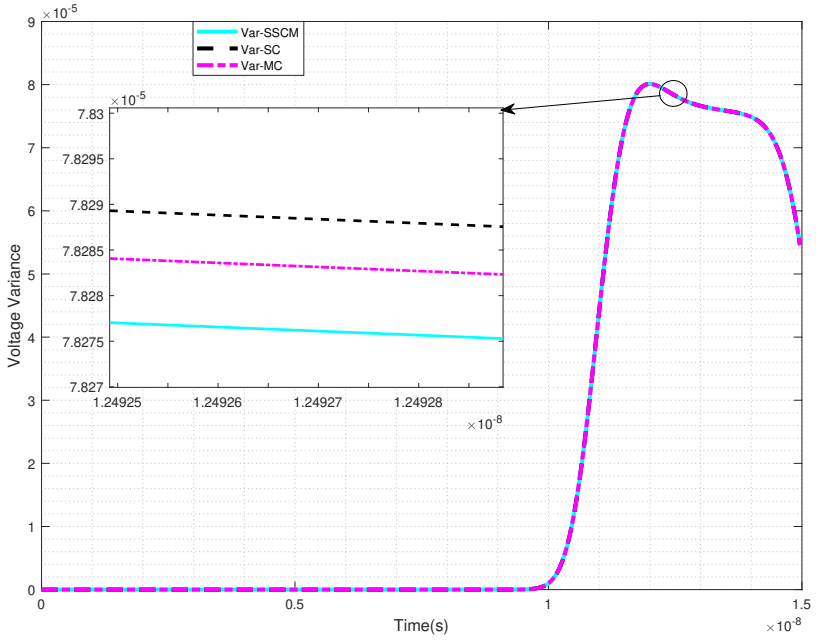

Figure 5–14: Variance comparison of Example 3 with SSCM, SCM at $\boldsymbol{p = 3}$ and $\boldsymbol{N = 8}$ and MCM

Figure 5–15 illustrates the probability density function of example 3 at two different time points of $\mathbf{7.5}\boldsymbol{ns}$ and $\mathbf{11.25}\boldsymbol{ns}$, respectively. We see that in the first subplot, the voltages have a value close to zero because the circuit is at the delay stage at time $\mathbf{7.5}\boldsymbol{ns}$ while the second one lies in the rising edge. Overall, SSCM still has

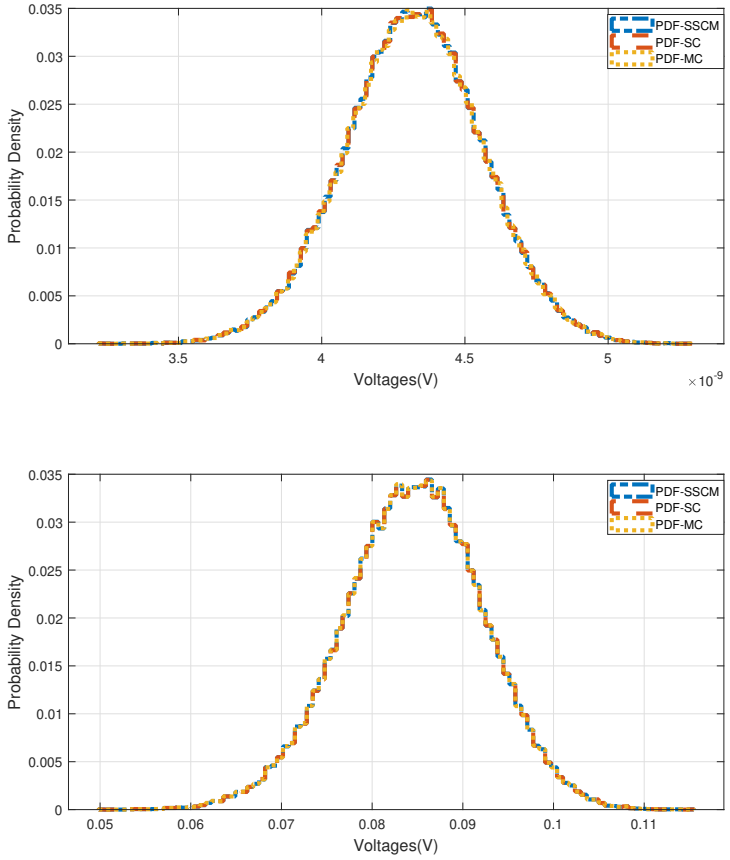

Figure 5–15: Probability density function comparison of Example 3 at time $\mathbf{7.5}\boldsymbol{ns}$ and $\mathbf{11.25}\boldsymbol{ns}$ with SSCM, SCM at $\boldsymbol{p} = \mathbf{3}$ and $\boldsymbol{N} = \mathbf{8}$ and MCM

the graphically indistinguishable probability distribution as the other two methods.

Figure 5–16 gives the 3-sigma limits of example 3. Upon combining Figure 5–13 to 5–16, we see that our SSCM does not have any significant loss but achieves the similar level of accuracy, proving the validity in terms of nonlinear circuits.

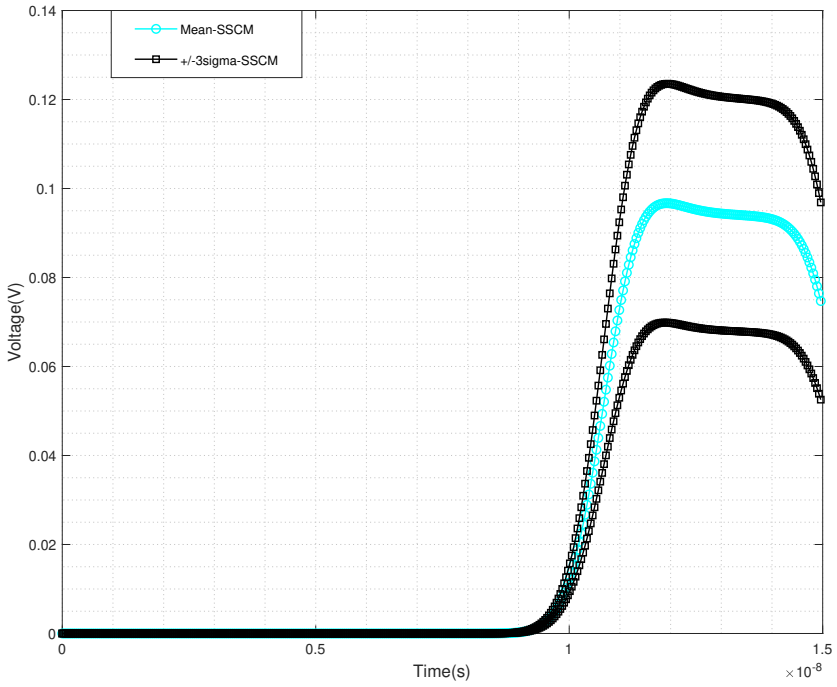

Figure 5–16: 3-Sigma limits of Example 3 with SSCM at $\boldsymbol{p = 3}$ and $\boldsymbol{N = 8}$

As we mentioned previously in linear case, we used Parallel Computing Tool integrated in MATLAB with a number of workers of 4, and this is also applied to nonlinear circuits. Under this circumstances, the time taken for the SCM to converge is 13 mins and 23 secs, which is 3.6 times slower than SSCM of 3 mins and 43 secs.

CHAPTER 6
Conclusion and Future Work

To sum up, this thesis has proposed an efficient approach called Sensitivity Integrated Stochastic Collocation method to solving the variability analysis of integrated circuits. This is done by first generating sampling points with Gaussian Quadrature rule and selecting a certain number of them depending on the number of random variables and the threshold within the algorithm motioned in section 4.1.1, then constructing the new deterministic system with differentiation form of Polynomial Chaos Expansion sequence. Once the responses of circuit sampling with these selected sets of points have been simulated and concatenated with sensitivity at a specific node with respect to all random variables in a set, we can readily solve for coefficients.

What makes our approach advantageous is that SSCM requires quite a few sampling points compared to the stochastic collocation method or linear regression, 20 percent in a general way, that in turn ease the pressure on the difficulty of computation. Such saving on time consumption is not only achieved by using fewer sampling points but also the re-usability of LU decomposition during circuit simulation. More importantly, SSCM maintains the same accuracy as Monte Carlo and Stochastic Collocation, as summarized in Table 6–1. The table shows that with promising accuracy, SSCM has an evident superiority in terms of CPU cost, whether linear or nonlinear circuits, averaging four times faster than SCM.

Table 6–1: Examples Testing Summary

Arguments	Example 1	Example 2	Example 3
Number of Random Variables	10	12	8
Number of Sampling points(SSCM)	57	91	33
Number of Sampling points(SCM)	286	455	165
Highest Total Order	3	3	3
Disparity between MC and SSCM	2×10^{-5}	1×10^{-5}	2×10^{-6}
Simulation time(secs)	128	738	223
Times faster than SCM	4.9	4.3	3.6

As for future work, one interesting point is that we can introduce the concept of Model Order Reduction(MOR) to our method. As we can imagine, with the growth of signal speed and the shrinking feature sizes in digital VLSI circuits, the resulted deterministic system dimension could be massive. Therefore, MOR can be used here for replacing the original RLC-interconnect network with reduced-order models to further reduce the computational complexity. Another intriguing path would be applying Shooting Newton method to work on the periodic steady-state analysis, which is an indispensable component of design process.

References

[1] Lubomír Brančík and Břetislav Ševčík. Fully time-domain simulation of multiconductor transmission line systems: Implicit wendroff and euler methods within modified nodal analysis. In *Proceedings of the Joint INDS'11 & ISTET'11*, pages 1–6. IEEE, 2011.

[2] Nikhil Chhabria, Adhishree Jaiprakash, Karan R Motwani, and Raghuram Srinivasan. Rlc circuit simulation and monte carlo analysis in matlab. In *2016 International Conference on Communication and Electronics Systems (ICCES)*, pages 1–6. IEEE, 2016.

[3] Lidia Daldoss, Paolo Gubian, and Michele Quarantelli. Multiparameter time-domain sensitivity computation. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(11):1296–1307, 2001.

[4] Peter Deuflhard and Andreas Hohmann. *Numerical analysis in modern scientific computing: an introduction*, volume 43. Springer Science & Business Media, 2012.

[5] K Guo, F Ferranti, B Nouri, and M Nakhla. A stochastic collocation technique for time-domain variability analysis of active circuits. In *2016 IEEE 25th Conference on Electrical Performance Of Electronic Packaging And Systems (EPEPS)*, pages 47–50. IEEE, 2016.

[6] Dou Lei and Wang Zhiquan. Sensitivity analysis and optimization for high-speed circuit systems. In *2007 International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, pages 1437–1441. IEEE, 2007.

[7] Enrico Malavasi, Stefano Zanella, Julian Uschersohn, Mike Misheloff, and Carlo Guardiani. Impact analysis of process variability on digital circuits with performance limited yield. In *2001 6th International Workshop on Statistical Methodology (Cat. No. 01TH8550)*, pages 60–63. IEEE, 2001.

[8] Paolo Manfredi. High-speed interconnect models with stochastic parameter variability. *Piemonte: Politecnico di Torino*, pages 81–88, 2013.

[9] Paolo Manfredi, Igor S Stievano, and Flavio G Canavero. Alternative spice implementation of circuit uncertainties based on orthogonal polynomials. In *2011 IEEE 20th Conference on Electrical Performance of Electronic Packaging and Systems*, pages 41–44. IEEE, 2011.

[10] Paolo Manfredi, Riccardo Trinchero, and Dries Vande Ginste. Variability analysis of a boost converter based on an iterative and decoupled circuit implementation of the stochastic galerkin method. In *2018 IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC)*, pages 996–1000. IEEE, 2018.

[11] MH Meyers. Computing the distribution of a random variable via gaussian quadrature rules. *Bell System Technical Journal*, 61(9):2245–2261, 1982.

[12] Tillmann Mühlpfordt, Rolf Findeisen, Veit Hagenmeyer, and Timm Faulwasser. Comments on truncation errors for polynomial chaos expansions. *IEEE Control Systems Letters*, 2(1):169–174, 2017.

[13] Ayhan A Mutlu, Charles Kwong, Abir Mukherjee, and Mahmud Rahman. Statistical circuit performance variability minimization under manufacturing variations. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4–pp. IEEE, 2006.

[14] Gordon Russell, F Burns, and Alex Yakovlev. Varma—variability modelling and analysis tool. In *2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 378–383. IEEE, 2012.

[15] Karanvir S Sidhu, Marco T Kassis, and Roni Khazaka. Efficient regression-based polynomial chaos using adjoint sensitivity. In *2019 IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, pages 1–3. IEEE, 2019.

[16] T Smedes and PGA Emonts. Statistical modeling and circuit simulation for design for manufacturing. In *International Electron Devices Meeting 1998. Technical Digest (Cat. No. 98CH36217)*, pages 763–766. IEEE, 1998.

[17] Domenico Spina, Francesco Ferranti, Tom Dhaene, Luc Knockaert, Giulio Antonini, and Dries Vande Ginste. Variability analysis of multiport systems via polynomial-chaos expansion. *IEEE Transactions on Microwave Theory and Techniques*, 60(8):2329–2338, 2012.

[18] Q Su and K Strunz. Stochastic circuit modelling with hermite polynomial chaos. *Electronics Letters*, 41(21):1163–1165, 2005.

[19] E Jan W ter Maten, Roland Pulch, Wil HA Schilders, and HHJM Janssen. Efficient calculation of uncertainty quantification. In *Progress in Industrial Mathematics at ECMI 2012*, pages 361–370. Springer, 2014.

[20] Dongbin Xiu. Fast numerical methods for stochastic computations: a review. *Communications in computational physics*, 5(2-4):242–272, 2009.

[21] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.

[22] Fei Yuan and Ajoy Opal. Sensitivity analysis of periodically switched linear circuits using an adjoint network technique. In *ISCAS'99. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI (Cat. No. 99CH36349)*, volume 5, pages 331–334. IEEE, 1999.

[23] Zheng Zhang, Tarek A El-Moselhy, Ibrahim M Elfadel, and Luca Daniel. Stochastic testing method for transistor-level uncertainty quantification based on generalized polynomial chaos. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(10):1533–1545, 2013.