SANDIA REPORT

SAND2016-9438
Unlimited Release
Printed September 2016

Advanced Uncertainty Quantification Methods for Circuit Simulation: Final Report, LDRD 2016-0845

Eric R. Keiter, Laura P. Swiler, Ian Wilcox

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

SAND2016-9438
Unlimited Release
Printed September 2016

Advanced Uncertainty Quantification Methods for Circuit Simulation: Final Report, LDRD 2016-0845

Eric R. Keiter
Electrical Models and Simulation
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1177

Laura P. Swiler
Optimization and Uncertainty Quantification
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1318

Ian Wilcox
Component and System Analysis
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1177

Abstract

This report summarizes the methods and algorithms that were developed on the Sandia National Laboratory LDRD project entitled “Advanced Uncertainty Quantification Methods for Circuit Simulation”, which was project # 173331 and proposal # 2016-0845. As much of our work has been published in other reports and publications, this report gives an brief summary. Those who are interested in the technical details are encouraged to read the full published results and also contact the report authors for the status of follow-on projects.

Acknowledgments

This work was funded by the Computing and Information Sciences (CIS) Investment Area within the Laboratory Directed Research and Development (LDRD) Program at Sandia National Laboratories. The authors thank the CIS committee and Program Manager for their support of this project. We thank all of our managers for their support of this activity. We also gratefully acknowledge the work and support of our colleagues on the Xyce and Dakota development teams, respectively.

Contents

List of Figures

1. Introduction

This report contains results of an investigation into advanced methods for uncertainty quantification (UQ) for circuits. This work was performed under LDRD2016-0845. The focus of the work has been on advanced UQ methods which can be posed as optimization problems which can be enhanced with gradient information. As such, steady-state and transient sensitivity analysis methods were developed and added to Xyce. For this work, the Xyce circuit simulator [1, 2] and Dakota software [3] were used.

The major accomplishments of this LDRD project include:

1. Xyce: Steady-state and transient sensitivities analytically calculated in Xyce, using both direct and adjoint methods.
2. Dakota: Field responses and random field model representations incorporated in Dakota.
3. Xyce-Dakota: Use of derivative-based UQ methods, including compressive-sensing based Polynomial Chaos Expansions and reliability methods.

Because much of our work has been documented in other reports, the purpose of this report is to provide an overview of the publications and presentations and provide a summary of the methods with pointers to the more detailed descriptions in other publications.

1.1 Research Products

Under this LDRD, we generated the following conference presentations, internal documentation, and papers:

- "Advanced Uncertainty Quantification Methods for Circuit Simulation: Interim Report." SAND2014-20238. Eric Keiter and Laura Swiler [4].
- "Sensitivity Analysis in Xyce." SAND2016-9437 Eric Keiter, Laura Swiler, Tom Russo and Ian Wilcox. [5]
- "Gradient-Enhanced Uncertainty Quantification Methods for Circuit Simulation." Eric Keiter, Laura Swiler, and Ian Wilcox. Submitted to *IEEE Transactions of CAD of Integrated Circuits and Systems*. [6]

- "Gradient-Enhanced Polynomial Chaos Methods for Circuit Simulation." Eric Keiter, Laura Swiler, and Ian Wilcox. Submitted to *Proceedings of 11th International Conference On Scientific Computing In Electrical Engineering* [7]
- SIAM UQ Presentation: "Methods for Data Reduction in Uncertainty Quantification: Reconciling parameters from compact modeling of electrical circuits." Presentation at the Society for Industrial and Applied Mathematics Uncertainty Quantification Conference, Savannah GA, April 2014.
- CIS Poster: "Uncertainty Quantification Methods for Electrical Simulation." Poster presentation to the External Review Committee for the Computational and Information Sciences IAT, May 2015.
- UQ Colloquium: "Advanced Uncertainty Quantification Methods for Circuit Simulation." Presentation at the ASC Verification, Validation, and Uncertainty Quantification Colloquium, April 2016.

1.2 Document Organization

We have documented the algorithms and approaches developed under this LDRD in several reports and articles. In this report, we provide a summary description of each major technical accomplishment with the detailed references that provide the actual algorithm development and implementation details. Chapter 2 covers steady-state and transient sensitivities in Xyce, Chapter 3 presents UQ approaches that can use analytic sensitivities, Chapter 4 covers field responses, and Chapter 5 provides a summary.

2. Xyce Sensitivities

Many of the optimization and UQ techniques of interest to this LDRD can be enhanced if the application code is able to produce parameter sensitivities with respect to objective functions of interest. Therefore to expedite this work, steady-state and transient sensitivity capabilities have been added to Xyce. Typically sensitivities are computed with respect to an output of interest. For example:

$$\frac{dO}{dp} = \frac{\partial O}{\partial x}\left(\frac{\partial F}{\partial x}\right)^{-1}\frac{\partial F}{\partial p} + \frac{\partial O}{\partial p} \tag{2.1}$$

Where O is the objective function (or output), p is a parameter, F is the residual equation and x is the solution vector. $\partial F/\partial x$ is the Jacobian matrix. A typical objective function, O, could be something like a circuit output current, or possibly a comparison of that current to measured data for purposes of calibration. The parameter, p, is a compact model parameter such as saturation current.

Sensitivities can be computed using two different methods; the direct method and the adjoint method. The difference between direct and adjoint is related to the order in which the terms of equation 2.1 are evaluated [8].

2.1 Direct Sensitivity

The general form for direct sensitivities is given by equation 2.2, in which square brackets have been added to equation 2.1 to indicate that $\partial x/\partial p$ is determined first by way of a linear matrix solve of the Jacobian matrix.

$$\frac{dO}{dp} = \frac{\partial O}{\partial x}\left[\left(\frac{\partial F}{\partial x}\right)^{-1}\frac{\partial F}{\partial p}\right] + \frac{\partial O}{\partial p} \tag{2.2}$$

$$\frac{dO}{dp} = \frac{\partial O}{\partial x}\frac{\partial x}{\partial p} + \frac{\partial O}{\partial p} \tag{2.3}$$

2.2 Adjoint Sensitivity

The general form for adjoint sensitivities is given by equation 2.4, in which square brackets have been added to equation 2.1 to indicate that $\partial O/\partial F$ is determined first by way of a linear matrix

solve of the transpose Jacobian.

$$\frac{dO}{dp} = \left[\frac{\partial O}{\partial x}\left(\frac{\partial F}{\partial x}\right)^{-1}\right]\frac{\partial F}{\partial p} + \frac{\partial O}{\partial p} \tag{2.4}$$

$$\frac{dO}{dp} = \frac{\partial O}{\partial F}\frac{\partial F}{\partial p} + \frac{\partial O}{\partial p} \tag{2.5}$$

Direct and adjoint formulas, given by equations 2.2 and 2.4 can be expanded to specifically apply to various kinds of analysis. The derivations, especially for transient sensitivities, are quite complicated. We have prepared a SAND report that focuses solely on the formulas and calculations for the different types of sensitivities: direct steady-state, adjoint steady-state, direct transient, and adjoint transient. This SAND report is [5]. Some of the derivations are also explained in [4] and [6].

2.3 Transient Direct Sensitivity

Transient direct sensitivities can be derived following the approach described by Hocevar. [9]. In transient, the derivation is slightly more complicated than steady-state, as the time derivative term, $\dot{q}$, must be accounted for. To be consistent with the original DAE solve, it is necessary to derive forward direct formulas for all the time integration methods of interest. More detailed derivations of the transient direct equations are given in reference [5].

For any integration method, a transient direct equation can be solved by starting with the differential algebcaic equation (DAE) form, which is given by:

$$F = \dot{q} + j - b = 0 \tag{2.6}$$

This equation is minimized at every time step in transient using a Newton solver. To obtain the direct sensitivity equation, equation 2.6 must be differentiated with respect to a parameter, p, and then re-arranged to give a linear system to be solved at each time step:

$$\frac{dF}{dp} = \frac{d}{dp}\left(\dot{q} + j - b\right) = 0 \tag{2.7}$$

$$\frac{dF}{dp} = \frac{\partial \dot{q}}{\partial p} + \left[\frac{\partial j}{\partial x}\frac{\partial x}{\partial p} + \frac{\partial j}{\partial p}\right] - \frac{\partial b}{\partial p} \tag{2.8}$$

For Backward Euler integration, the time derivative form for $\partial q/\partial p$ is given by:

$$\frac{\partial \dot{q}}{\partial p} = \frac{1}{h}\left(\frac{\partial q}{\partial p}\left(x_n\right) - \frac{\partial q}{\partial p}\left(x_{n-1}\right)\right) \tag{2.9}$$

Using Backward Euler, substitute in the equation for dq/dp, which ultimately gives:

$$J\frac{\partial x}{\partial p}_n = -FD + CR \tag{2.10}$$

Where J is the original Jacobian given by:

$$J = \left[\frac{1}{h}\frac{\partial q}{\partial x} + \frac{\partial j}{\partial x}\right] \tag{2.11}$$

FD is the "function derivative", or the partial derivatives that come directly from the device models, and is given by:

$$FD = \frac{1}{h}\left[\frac{\partial q}{\partial p}_{n} - \frac{\partial q}{\partial p}_{n-1}\right] + \frac{\partial j}{\partial p} - \frac{\partial b}{\partial p} \tag{2.12}$$

The remaining term, CR, in this paper is referred to as the chain-rule term, given by:

$$CR = \frac{1}{h}\left[\frac{\partial q}{\partial x}\right]\frac{\partial x}{\partial p}_{n-1} \tag{2.13}$$

Note that the chain rule term is using the Q-matrix ($\frac{\partial q}{\partial x}$) from the previous time step, $n-1$. So, for the implementation to be correct, it is necessary to either store store previous matrix-vector muliplication results of $\frac{\partial q}{\partial x}\frac{\partial x}{\partial p}$.

2.4 Transient Adjoint Sensitivity

Transient adjoint sensitivities [8, 10, 11] can be broadly classified in broadly classified into two categories: *discrete adjoint sensitivities* and *continuous adjoint sensitivities*. Discrete adjoint sensitivities are obtained when one applies the adjoint operator after discretizing the original DAE system, and continuous adjoint sensitivities are obtained from applying the adjoint operator first, and then discretizing the result. For this project, only the discrete form of transient adjoint sensitivities were implemented in **Xyce**. The following discrete adjoint derivation follows the description of [8] and [10]. More details of the transient adjoint derivation are given in reference [5].

When considering the transient case, it is convenient to consider the full transient in block form. If a transient simulation consists of N time points, then all the time points can be considered in a single block matrix equation, which has the same general form as equation 2.6:

$$\mathbf{\Psi} = \dot{\mathbf{Q}} + \mathbf{J} - \mathbf{B} = \mathbf{0} \tag{2.14}$$

where $\mathbf{\Psi}$ is the block residual vector given by $\mathbf{\Psi} = [F_0, F_1, ..., F_N]$. The other terms in the equation: $\mathbf{X}$,$\dot{\mathbf{Q}}$, $\mathbf{J}$, and $\mathbf{B}$ are block analogies of the original DAE equation terms: x, q, j, and b, respectively. Similarly, $\mathbf{\Theta}$ and $\mathbf{\Theta}^{\star}$ are the block forms of dx/dp and dO/dj. For conventional time integration methods, the block Jacobian takes on the form of a lower triangular matrix:

$$\frac{\partial \mathbf{\Psi}(\mathbf{X})}{\partial \mathbf{X}} = \begin{bmatrix} \left(\frac{\partial F_0}{\partial x_0}\right) & & & \\ \left(\frac{\partial F_1}{\partial x_0}\right) & \left(\frac{\partial F_1}{\partial x_1}\right) & & \\ \vdots & \vdots & \ddots & \\ & & & \left(\frac{\partial F_N}{\partial x_N}\right) \end{bmatrix} \tag{2.15}$$

Where the linear block system is:

$$\frac{\partial \mathbf{\Psi}}{\partial \mathbf{X}}\Theta = \frac{d\mathbf{\Psi}}{dp} \tag{2.16}$$

Where $\mathbf{\Theta} = [\Theta_0, \Theta_1, ..., \Theta_N]$ is the derivative of the block solution $\mathbf{X} = [x_0, x_1, ..., x_N]$ with respect to a parameter value p. ie, $\Theta_0 = dF_0/dp$. The block matrix is upper triangular and banded depending

on the integration method. If the method is Backward Euler (BE), then, the equivalent equation corresponding to row n of matrix 2.15 is:

$$\left(\frac{\partial F_n}{\partial x_n}\right)\frac{dx_n}{dp} = -\left(\frac{\partial F_n}{\partial x_{n-1}}\right)\frac{dx_{n-1}}{dp} + \frac{dF_n}{dp} \tag{2.17}$$

The left hand side of 2.17 is the original Jacobian matrix given by equation 2.11 at time point n multiplied by dx_n/dp.

$$\left(\frac{\partial F_n}{\partial x_n}\right) = \left[\frac{1}{h}\frac{dq}{dx_n} + \frac{dj}{dx_n}\right] \tag{2.18}$$

The first term on the right hand side of equation 2.17 (involving an off-diagonal Jacobian block) is equivalent to equation 2.13.

$$\left(\frac{\partial F_n}{\partial x_{n-1}}\right) = -\frac{1}{h}\left[\frac{dq}{dx_{n-1}}\right] \tag{2.19}$$

The remainder of the right hand side of equation 2.17 is the similar to equation 2.12.

$$\frac{dF_n}{dp} = \frac{1}{h}\left[\frac{dq_n}{dp} - \frac{dq_{n-1}}{dp}\right] + \frac{dj_n}{dp} - \frac{db_n}{dp} \tag{2.20}$$

The transpose block Jacobian takes on the form of a upper triangular matrix:

$$\left(\frac{\partial \mathbf{\Psi}(\mathbf{X})}{\partial \mathbf{X}}\right)^T = \begin{bmatrix} \left(\frac{\partial F_0}{\partial x_0}\right)^T & \left(\frac{\partial F_1}{\partial x_0}\right)^T & & \\ & \left(\frac{\partial F_1}{\partial x_1}\right)^T & & \\ & & \ddots & \vdots \\ & & & \left(\frac{\partial F_N}{\partial x_N}\right)^T \end{bmatrix} \tag{2.21}$$

Where the linear block system is:

$$\left(\frac{\partial \mathbf{\Psi}}{\partial \mathbf{X}}\right)^T \Theta^\star = \frac{\partial O}{\partial \mathbf{X}} \tag{2.22}$$

Where $\Theta^\star = [\theta_0^\star, \theta_1^\star, ..., \theta_N^\star]$ is the derivative of the objective function O with respect to the block residual vector $\mathbf{\Psi} = [F_0, F_1, ..., F_N]$. Once $\Theta^\star$ has been computed it can be used (analogous to the steady-state case) to obtain dO/dp by taking the dot product with $d\mathbf{\Psi}/dp$. As the adjoint block linear system is a lower-triangular system, it necessary to compute the sensitivity by performing the time integration backwards in time, starting at the final time point N.

2.5 Device Model Support

Production-level circuit simulator contain many different electrical component models, often referred to as “compact models”. Some of these models, such as linear resistors, are very simple; while others, such as those supporting modern semiconductor technology nodes can be quite complex. To support parameter sensitivities, the terms dj/dp, dq/dp and db/dp (used by both adjoint and direct methods) must be provided by the individual models. Given the complexity of modern compact models, as well as the volume of models typically supported by a simulator like **Xyce**, this is a prohibitive undertaking. Fortunately, there are modern tools available that can be

leveraged to make this more feasible, including automatic differentiation (AD) and model compilers.

Many **Xyce** models have recently been modified to provide derivatives using the Sacado [12] automatic differentiation (AD) library. However, for legacy models that have already been established, and were not originally designed to support the templating required by the Sacado API, it can be difficult to retrofit the model to use Sacado. A few models in **Xyce** have been manually modified in this manner, but doing so required a nearly complete rewrite of each model. Fortunately, for many models one can circumvent this issue by using model compilers.

Modern complex device compact models are often developed using a high-level modeling language such as Verilog-A. Verilog-A is the current industry standard for compact modeling, as established by organizations such as the compact model coalition (CMC) [13]. To use a model written in Verilog-A it is necessary to use a model compiler. Model compilers are software tools which parse Verilog-A input files, and store the model in an abstract data structure. Once stored as abstract data, the model can then be written in simulator-specific source code. The model compiler takes care of many implementation details, such as Jacobian derivatives. The resulting source code can then be compiled and linked into the simulator.

There are numerous benefits to using this approach, which have been well-documented in the compact modeling literature. It allows the model developer to focus on the models, rather than the implementation details. Additionally, once a model has been developed, it doesn't have to be re-developed for subsequent simulators. However, one benefit of model compilers has mostly been ignored in the literature. That benefit is the application of AD to support analytic parameter derivatives. This particular benefit has been explored as part of this project and represents a unique research result, and a unique feature of **Xyce**.

One model compiler, the Automatic Device Model Synthesizer (ADMS) [14] is open-source and has been modified for use with **Xyce**. This compiler has been used to implement many recent industry-standard compact models [15, 16, 17, 18, 19, 20, 21]. For this project, the **Xyce**-specific back end to ADMS was extended to produce parametric sensitivities using Sacado AD. Once this modification was applied, it was possible to regenerate the **Xyce** source for all of the ADMS-based models, with sensitivity support included. As a result nearly all the modern transistor models in **Xyce** support analytic sensitivities. This is a unique feature of **Xyce**, compared to other circuit simulators.

2.6 Impact of Sensitivities in **Xyce**

As noted, analytic parametric sensitivities are a unique feature of **Xyce**. It is only with the combination of model compilers and automatic differentiation that implementation in a production simulator is feasible. As such, this capability does not appear to be available in any rival open-source simulator [22, 23]. It is more difficult to assess if this feature is available in any commerical circuit simulators, but it is not a commonly available feature.

As demonstrated in section 3, analytic parameter derivatives can have a significant impact on UQ, by providing additional information to the chosen UQ algorithm. They can also be extremeley useful in circuit optimization, which is the topic of a follow-on project.

3. Gradient Enhanced UQ Methods

There are two classes of uncertainty quantification methods which can benefit from analytic sensitivities: reliability methods and stochastic expansion methods. We provide a very brief overview here. These methods are both outlined in more detail in [4] and [6]. Reliability and stochastic expansion methods are both "forward UQ" methods because they propagate uncertain input parameters defined by probability distributions through a simulation code to obtain the resulting probability distribution on outputs or responses of interest. There are also "backward UQ" methods, also called inverse UQ or parameter estimation methods. These approaches take uncertain data on outputs (e.g. from physical experiments) and use that data to infer what the corresponding uncertainty in the inputs must have been to result in those uncertain outputs. Initially, we focused more on reliability methods in this LDRD, because they also are used in a particular form of backward UQ called "backward propagation of variance" or BPV [24]. In the second half of the LDRD, we have focused more on stochastic expansion methods for two reasons: (1) they tend to be more accurate and consistent and (2) Xyce has future plans to calculate the stochastic expansions in an "intrusive" or embedded manner.

In this chapter, we provide a very brief overview of the methods and show an example with a comparison of the methods.

3.1 Reliability Methods

Reliability methods are a popular uncertainty quantification approach which can be less computationally demanding than sampling, particularly when quantifying uncertainties with respect to a particular response or probability level. The goal of reliability methods is to estimate failure probabilities, i.e., the probability that an output response exceeds or falls below a threshold value. These approaches preferentially evaluate points around the area of interest, namely the failure region, and may use successive approximations to refine failure esti- mates. Reliability methods can be more efficient than Monte Carlo sampling when computing statistics in the tails of the response distributions (low probability events). These methods transform the UQ problem to an optimization problem and use gradients to solve the optimization problem, which is why the analytic sensitivities from Xyce are so important. The interested reader is encouraged to start with [25] for an excellent overview of local reliability methods.

3.2 Stochastic Expansion Methods

Stochastic expansion UQ methods approximate the functional dependence of the simulation response on uncertain model parameters by expansion in a polynomial basis. The polynomials used are tailored to the characterization of the uncertain variables. Polynomial chaos expansion (PCE) is based on a multidimensional orthogonal polynomial approximation. There are many variations of PCE. The one we have focused on involves sampling the simulation at random input points and also using the analytic sensitivities at these points to obtain a regression-based formulation for solving for the coefficients of the expansion. This "derivative-enhanced regression PCE" has been a main focus of interest in the LDRD. For more details about PCE, the user is encouraged to review [26] and [27].

3.3 Results for CMOS Inverter Circuit

In this section, we demonstrate the use of the gradient-enhanced UQ methods on a five-stage CMOS inverter which involves transient sensitivities calculated by Xyce. We model a simple CMOS five-stage inverter circuit, which uses 10 instances of the BSIM6 [28] compact model. This circuit is meant to mimic applications where signal delay is the important metric. Each inverter stage adds to the signal delay. The CMOS circuit is shown in Figure 3.1. The PMOS and NMOS oxide thicknesses are thus critical uncertain parameters. We model these as normal uncertainties, centered around a nominal value with a standard deviation equal to 10% of nominal.

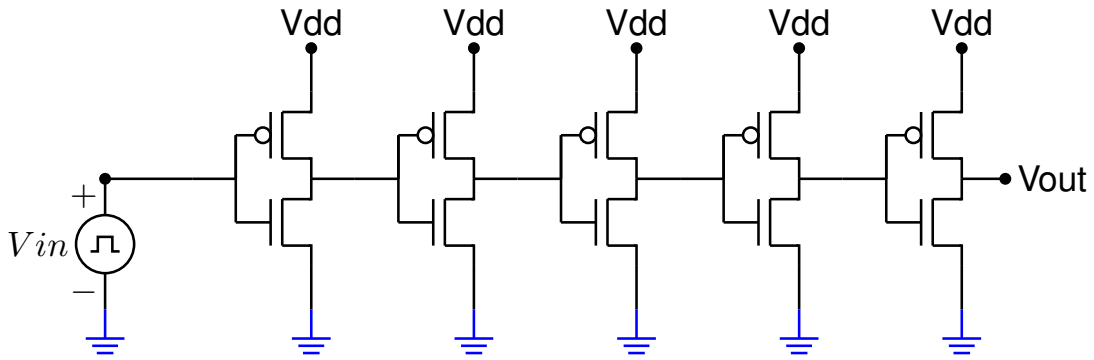

Figure 3.1. CMOS circuit with five inverters.

The circuit is driven by a step input, and the output of interest is the output voltage V_{out}. This is shown in Figure 3.2.

Since the output voltage is a transient signal, we used a generalized Elmore delay, similar to that given by [29], as our objective function of interest. The Elmore delay is given by:

$$O = \texttt{Elmore Delay} = \frac{\int_0^T g'_A(t) \cdot t \cdot dt}{\int_0^T g'_A(t) dt} \tag{3.1}$$

and represents the approximate time for the signal rise or fall. Note that $g_A(t) = V_{out}$.

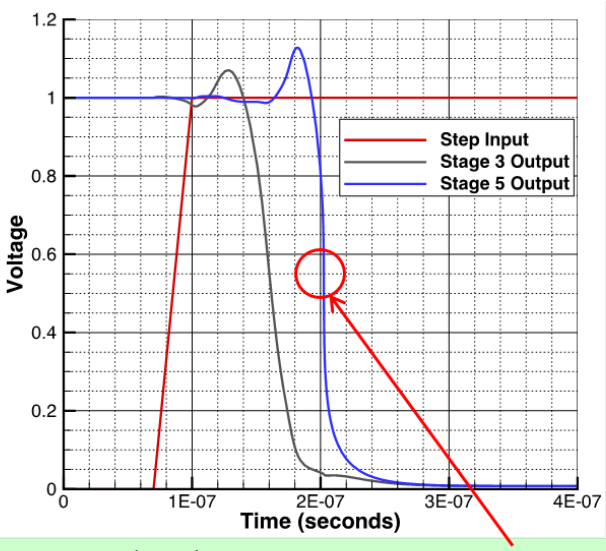

Figure 3.2. Overall behavior of CMOS circuit with Elmore delay highlighted

Xyce returns the transient sensitivities (e.g. the derivatives of the Elmore delay objective with respect to the thickness of the NMOS and PMOS oxide layers as a function of time). This is shown in Figure 3.3.

We performed uncertainty quantification on the CMOS circuit using a variety of UQ techniques. As a baseline, we performed Latin Hypercube Sampling (LHS) with 100 and 1000 samples. LHS is a stratified sampling method which has good space-filling properties and generally gives better results than plain Monte-Carlo sampling (e.g. results which have lower variance on statistical estimators such as the mean). Then, we performed polynomial chaos expansion using a full tensor product quadrature of order 5 for each of the two input parameters, resulting in a total of 25 sample points. Finally, we performed two types of regression-based PCE. In the first, we used 30 samples but did not include the gradients. In the second, we used 10 samples. For each sample, we had two gradient values representing the derivative of the Elmore delay with respect to the two input parameters. Thus, the last PCE calculation used 30 pieces of information and was comparable to the 30 sample regression PCE with no gradients, but it only required 10 samples.

Note that for all of these sample runs, the two input parameters were the NMOS and PMOS oxide layer thickness.They were varied according to a normal distributions with means of 1.74E-9m and 2.34E-9m, and standard deviations of 1.75E-10 and 1.34E-10, respectively. After each Xyce run, the Elmore delay was calculated and used as the quantity of interest in these analyses. All of the runs were performed using Dakota for the UQ methods and Xyce as the circuit simulator.

The use of sensitivities in performing uncertainty analysis is highlighted in Figure 3.4 and Table 3.1. As shown in the figure, the cumulative distribution function (CDF), which gives the probability that the Elmore delay is less than a particular value, is almost the same for an LHS sample of size 1000 and all of the PCE methods. It is very hard to see differences: the CDF curves for LHS 1000

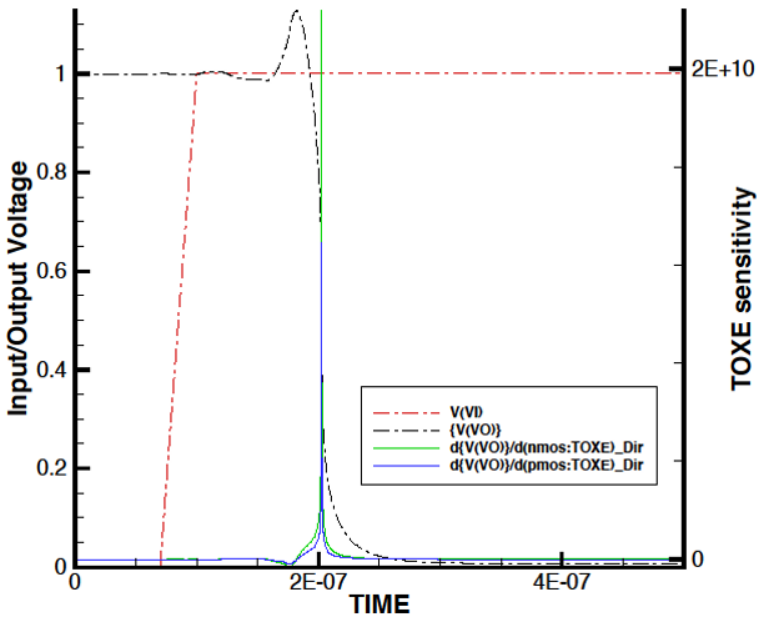

Figure 3.3. Transient sensitivities from the CMOS circuit highlighted

Table 3.1. Comparison Results from UQ Approaches

Number of samples and UQ Method	Mean	Std Dev.
100 LHS	2.0781E-7	6.6309E-9
1000 LHS	2.0782E-7	6.6935E-9
25 PCE Quadrature	2.0783E-7	6.6954E-9
30 PCE Regression	2.0783E-7	6.7131E-9
10 PCE Regression with derivatives	2.0782E-7	6.7035E-9

and all of the PCE variants overlay each other. The one that is different is LHS based only on 100 samples. Figure 3.4 shows that this CDF is not as resolved as the others. Table 3.1 shows that the mean values of the Elmore delay are very similar, differing only in the fifth significant figure. Finally, the standard deviations show a little more variability, but again are reasonably close. We conclude that a polynomial chaos expansion using sensitivities from Xyce (the 10 PCE with regression case) performs comparably to 1000 samples from LHS.

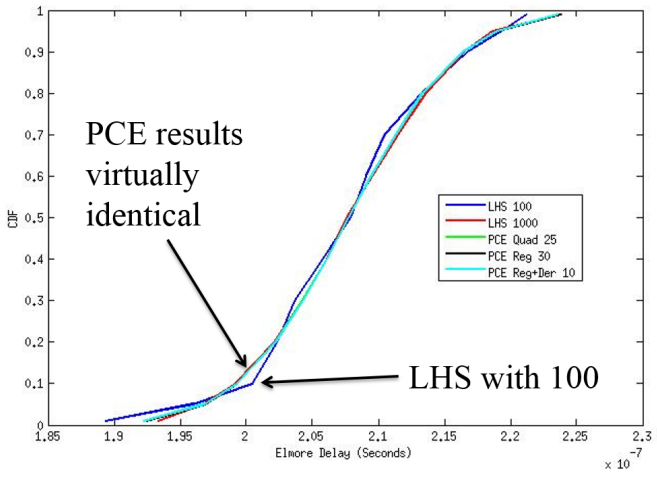

Figure 3.4. Cumulative Distribution Function of Elmore Delay using Various UQ Approaches

4. Field Data

Field data refers to responses that are a function of independent coordinates such as time and/or spatial coordinates. For example, circuit responses such as voltage or current can be a function of time. In these cases, the output from Xyce is a table of timesteps and corresponding current or voltage values. For many years in Dakota, responses were treated as scalar values. That is, if Xyce returned voltages at 1000 timesteps, these were treated as 1000 separate responses.

Under this LDRD, we added the notion of "field responses" in Dakota. The treatment of field responses is different from the treatment of separate, scalar responses at each independent coordinate: one treats the correlations in the field. In the example above with 1000 timesteps and 1000 corresponding voltages, this can now be handled as one field response in Dakota with length 1000. Another common example of field responses in circuit simulations are I-V curves: the dependent variable current is a function of the independent variable voltage.

The field response capability is described in the Dakota User's manual [3] and the Dakota Reference manual [30]. The addition of field responses facilitated other capabilities. Specifically, we added a capability to interpolate field responses from simulations to field data from physical experiments. This allows Dakota to calculate the residual terms (difference between simulation result and experiment) for the sum-of-squares error objective in calibration problems. For example, previously if a Dakota user was calibrating an emitter current in a VBIC transistor model to some experimental data, he or she had to interpolate the Xyce current around nearby voltage values to return the current exactly at the experimental voltage to construct the residual term for the least-squares calibration. Now, Dakota will take the I-V curve from the simulation and interpolate it to the I-V experimental data points to construct the residuals. Further, we developed methods (both finite difference and analytic) for handling gradients in the interpolated case since we need gradients for gradient-based calibration methods such as Gauss-Newton solvers. We also allow for mixed calibration to both scalar and field data. For example, one may have some I-V data to use in calibration but also want to match a particular temperature or pressure condition of the experiment given as a scalar value. Finally, we allow for more sophisticated treatment of experimental measurement error: separate measurement error can be given at each point in the field and/or a full covariance matrix specifying the error across the field can be given.

4.1 Random Field Model

The field response capability outlined above was initially designed to use analytic sensitivities from Xyce in calibration with interpolation. However, the field responses also can be used in uncertainty quantification. Specifically, if we do a forward propagation of uncertain input parameters (such as

IS, BF, and other transistor parameters), one generates an ensemble of results. A random field model can be thought of as a mathematical model (e.g. an abstraction) of an ensemble of field responses. There are two main classes of random field models we use in Dakota:

1. Karhunen-Loeve expansion (K-L)
2. Principal Component analysis (PCA)

These are very similar: both K-L and PCA represent a random field as a sum of basis functions weighted by coefficients. The difference is in how the uncertain coefficients are represented. Under this LDRD, we focused on PCA and created a predictive PCA where the coefficients are Gaussian process models that are functions of the uncertain circuit parameters such as transistor parameters IS, BF, etc. The PCA can be used to generate more field realizations and/or can be used as a surrogate field model. Our approach is outlined in the following section.

4.2 Principal Component Analysis

We use Principal Component Analysis as a dimension reduction method for field responses. There are many good references on PCA [31]. For the application of PCA to field or functional data, we encourage the interested reader to review the book titled "Functional Data Analysis" by J.O. Ramsay and B.W. Silverman [32]. This book goes into more treatment about functional data (e.g. treating the covariance matrix of the data as a function of time), but we are essentially treating the data as discretized values and using a standard PCA with our approach.

In PCA, the idea is to take possibly correlated data and use an orthogonal transformation to obtain uncorrelated components which explain the majority of the variance seen in the original data. The principal components are constructed such that the first component explains the largest amount of variability in the data, the second component explains the second largest amount of variability in the data subject to being orthogonal to the first component, etc. The principal components are the eigenvectors of the covariance matrix of the data. Typically, the original data set is mean-centered (e.g. each column has the mean subtracted from it). Then, an eigenvalue decomposition is performed on the covariance matrix of the data. The eigenvectors corresponding to the largest eigenvalues are the principal components. In PCA, the results are often presented in terms of factor scores and loadings. Factor scores are the transformed variable values corresponding to a particular data point, and loading refers to the weight or value which each original variable should be multiplied by to obtain the factor score. This will be explained in more detail below.

We present more detail about the computation of the principal components for our application. We assume that we have a set or ensemble of I-V curves that have been generated by generating samples on Xyce input parameters and running a simulation repeatedly at different input parameter values. The goal is to identify the principal components responsible for the spread of that data ensemble. For notation purposes, there are d input parameters, N samples, and the field length is L. Focusing only on the output from the simulation, we have a data matrix (the full data matrix) **XF** of dimension $N * L$. For example, if we take 100 samples of a 10000 length response, **XF** is a matrix of 100*10000.

To perform a PCA on **XF**, we do the following:

- Center the data.

  Take **XF** and center it so that the mean of each column is subtracted from that column. That is:

$$\mathbf{X} = \begin{bmatrix} \vdots & \vdots & & \vdots & \\ \vdots & \vdots & & \vdots & \\ XF_{i,1} - \overline{XF_{:,1}} & XF_{i,2} - \overline{XF_{:,2}} & \cdots & XF_{i,L} - \overline{XF_{:,L}} \\ \vdots & \vdots & & \vdots & \\ \vdots & \vdots & & \vdots & \end{bmatrix} \tag{4.1}$$

  where the dimension of **X** is $N * L$, with indexes $i = 1 : N, j = 1 : L$. **X** is the mean-corrected version of **XF**. The notation $\mathbf{X_i}$ denotes the i^{th} row of **X** (a particular observation).

- Perform eigen-decomposition of the covariance matrix.

  The idea in PCA is to construct a linear representation of variable values to highlight the variance present in the data. To find the first principal component, we define f_i as a linear combination of variable values:

$$f_i = \sum_{j=1}^{L} \beta_j x_{ij} = X_i \boldsymbol{\beta} \tag{4.2}$$

  Then, we find the set of (L*1) coefficients $\boldsymbol{\beta_1}$ such that the linear combination $f_{i1} = X_i \boldsymbol{\beta_1}$ has the largest possible mean square (defined as $\sum_i {f_i}^2/N$) subject to the constraint that $\|\boldsymbol{\beta_1}\|^2 = 1$. The optimal $\boldsymbol{\beta_1}$ when applied to all observations will result in a vector Y_1 of dimension N*1: $Y_1 = X_1 \boldsymbol{\beta_1}$. This vector identifies the strongest and most important modes of variation. The process is repeated with subsequent steps, to determine the sets of coefficients $\boldsymbol{\beta_2}$, $\boldsymbol{\beta_3}$ that define the second, third components, etc. These coefficient vectors are called "loadings" or "factor loadings" and the Y vectors are called "factor scores" or "principal component scores."

  In summary, we have a system:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} \tag{4.3}$$

  Typically, the process is only carried out until the M principal components are identified that account for a certain percentage of the variance. To calculate the coefficients $\boldsymbol{\beta}$, it is necessary to calculate the eigenvectors of the covariance matrix of **X**. That is, we calculate the matrix

$$\mathbf{V} = \frac{\mathbf{X^T X}}{\mathbf{N}} \tag{4.4}$$

  The first eigenvector (with the largest eigenvalue) of **V** corresponds to $\boldsymbol{\beta_1}$, the second eigenvector of **V** corresponds to $\boldsymbol{\beta_2}$, etc. Note that the full $\boldsymbol{\beta}$ can be of dimension $L * L$ but typically is smaller, $L * M$. Also, note that to perform a prediction in the original space, we use the loadings and factor scores. Equation 4.5, derived from Equation 4.3, uses the fact that the inverse of an orthogonal matrix is its transpose:

$$\mathbf{X} = \mathbf{Y}\boldsymbol{\beta^T} \tag{4.5}$$

In some circuit examples, a few (e.g. one to three) components are sufficient to represent the data in the sense of explaining 95% or 99% of the variance of the original data. Note that one component is a vector. In the example above, the first component $\boldsymbol{\beta_1}$ would be of dimension 10000*1. We can use Equation 4.5 to obtain the predicted values of the centered data given the factor scores **Y** and the principal components.

Thus far, we have presented the PCA to obtain the principal components for the responses of the circuit simulation (e.g. the I-V curve). The next step is to relate these to the input parameters of the circuit simulation which generated the spread in the data. To do this, we propose to develop a surrogate or emulator model for the functional data. We construct Gaussian process (GP) surrogates for the factor scores of the M principal components. The GP surrogates will be function of the uncertain inputs. The idea is that we have just performed PCA on (for example) the covariance matrix of N samples. Typically, those N samples will be generated by sampling over some d uncertain input parameters we name u, so there should be a mapping from u to the response data, specifically to the loading coefficients and the factor scores. We want to create a surrogate model so that we can predict a new set of data for parameter values u that are not in the particular sample set of the N original samples.

A particular prediction will be considered as:

$$\mathbf{X_{pred}} = \mathbf{GP_1}(\mathbf{u})\boldsymbol{\beta_1} + \mathbf{GP_2}(\mathbf{u})\boldsymbol{\beta_2} + ... + \mathbf{GP_M}(\mathbf{u})\boldsymbol{\beta_M} + \boldsymbol{\mu_X} \tag{4.6}$$

Where $\boldsymbol{\mu_X}$ is a vector containing the column means of the original data **XF**. Note that in this equation, the $\boldsymbol{\beta_i}$ coefficients and the $\boldsymbol{\mu_X}$ vectors are of dimension L*1. The GP surrogates each produce a scalar value, so the prediction is of dimension $L * 1$. If one looks carefully at the relationship between Equations 4.5 and 4.6, the main difference is that Gaussian process models are multiplying the principal components in Equation 4.6, whereas the factor scores are multiplying those principal components in Equation 4.5. To estimate the Gaussian process model coefficients, we use a training set that contains N samples of the d inputs u, and N samples of the factor scores Y_i. The Gaussian process is just a surrogate model mapping the uncertain inputs u to estimates of the factor scores. For example, for the first principal component, if we have 4 uncertain inputs, we have a matrix of N*4 input parameters that we want to map to an N*1 factor score Y_1. We perform this estimation by using a code that calculates the parameters governing the Gaussian process surrogate. This code is in the software framework called Dakota [3], but other Gaussian Process emulators could be used. After the GP is constructed, we have a set of surrogate models $\mathbf{GP_i}$ which are surrogates for the factor scores. Equation 4.6 can be considered a generalization of Equation 4.5. In Equation 4.5, we use the first M principal components to recover the original data from the N samples, but in Equation 4.6 we are using the principal components and surrogates for the factor scores to predict values of the data at new sample points.

PCA Results

This section shows an example of an ensemble of I-V curves, generated in Figure 4.1a. This figure shows ensemble results (the blue lines) of I-V curves generated by varying VBIC parameters IBEI, IBEN, NEI, and NEN. One hundred samples of these uncertain parameters were generated using the Latin Hypercube sampling algorithm in Dakota. The Xyce VBIC model was run at all the sample points to generate the 100 blue I-V curves. The right-hand Figure 4.1b shows the correlation of the

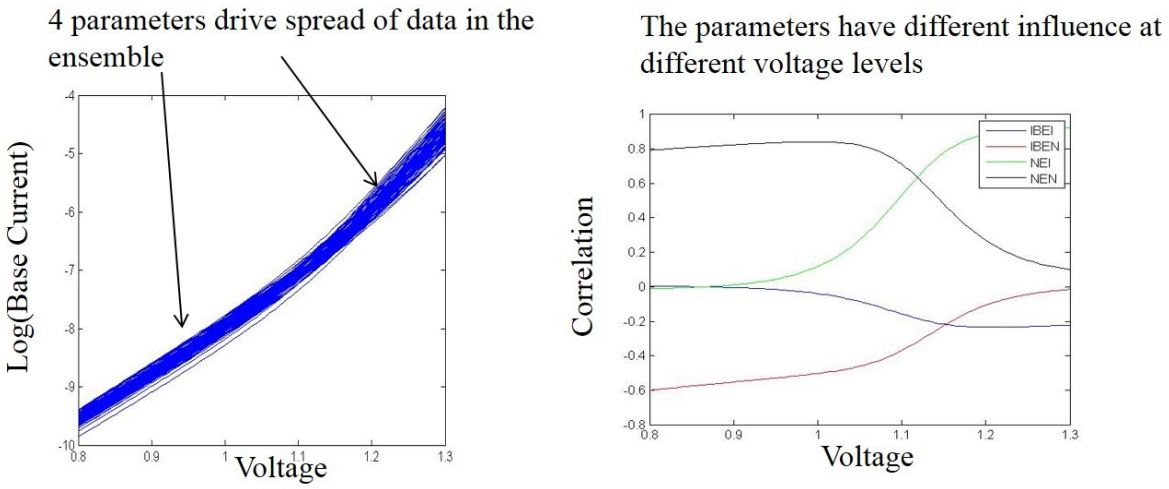

Figure 4.1. Left side (a): Ensemble of I-V curves generated by varying four VBIC parameters; Right side (b): Importance of these four parameters as a function of the voltage level.

input parameters to the output current as a function of voltage. One can see that the parameters have different influence depending on the voltage level: IBEN and NEN are important at lower voltages, and IBEI and NEI are important at higher voltages.

We applied the PCA dimension reduction approach described above. The results are shown in Figure 4.2. The left-hand figure shows that the first principal component captures the variance at higher voltages, but not at lower. It requires two principal components, shown in the right-hand figure, to capture the variance well at both ends of the I-V curves.

Finally, we used the Gaussian process surrogate approach described above coupled with the principal components to generate new sample realizations of I-V curves. That is, we generated new samples of the four inputs, but instead of running them through Xyce's VBIC model, we propagated them through Equation 4.6. The resulting new ensemble I-V realizations are shown in green in Figure 4.3.

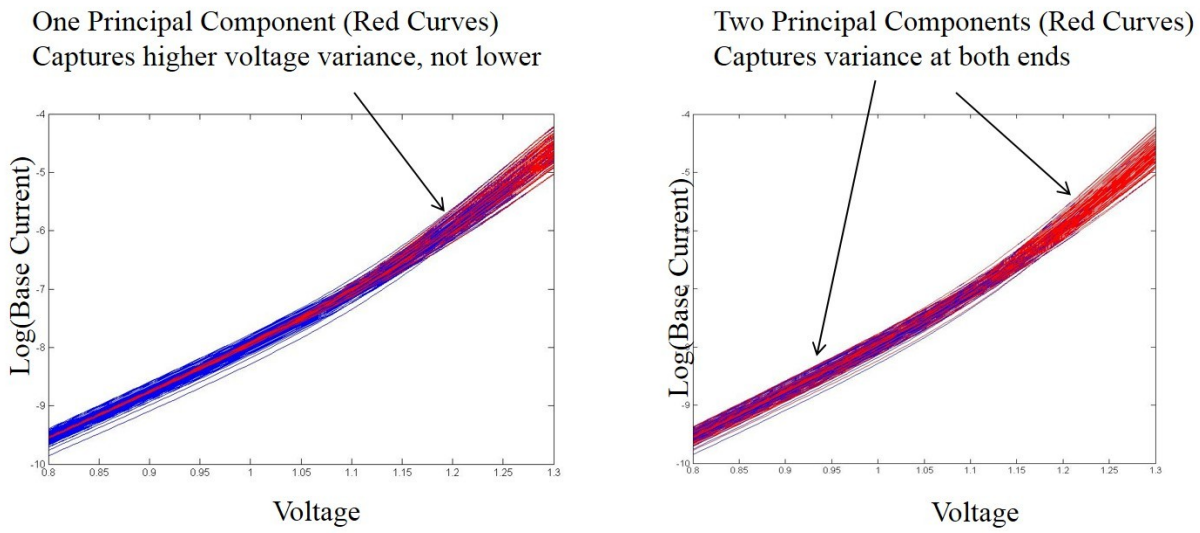

Figure 4.2. Left side (a): One principal component captures the variance at higher voltage levels; Right side (b): Two principal components capture variance at both ends. Original Xyce ensemble samples are shown in blue, with principal component results shown in red.

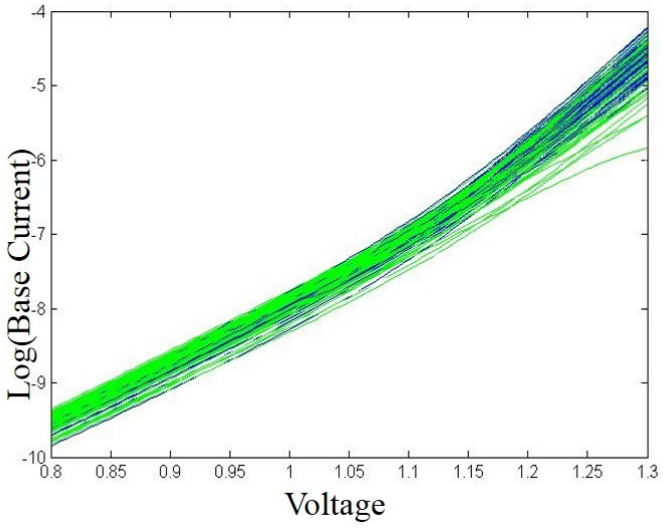

Figure 4.3. New ensemble realizations of I-V curves generated using PCA approach (green) with original ensemble data shown in blue.

5. Summary

To summarize, this LDRD project investigated advanced uncertainty quantification methods that rely on having accurate, analytic sensitivities. The calculation of these sensitivities (both transient and steady-state) have been added to Xyce, so that they are analytically provided using either direct or adjoint methods. These analytic sensitivites provide derivatives of responses of interest with respect to uncertain parameters. These sensitivities are much more accurate than finite difference calculations of the derivatives.

Given that many of UQ and calibration methods involve optimization, they can be enhanced with gradient information. We have observed that for compact model calibrations, accurate derivatives are very important, and thus finite-difference based sensitivities are frequently insufficient. Currently, the sensitivity implementation in Xyce relies on either analytic or finite difference sensitivities on the device level. The type of implementation is dependent on specific device model implementations.

We examined the use of Xyce's analytic sensitivities for UQ on circuit simulation problems using reliability methods and polynomial chaos expansions. In terms of forward uncertainty propagation, we found that reliability methods and PCE worked very well, requiring significantly fewer function evaluations than sampling, especially on linear problems with Gaussian random inputs. For nonlinear problems or problems with non-Gaussian inputs, PCE tended to be more accurate and consistent than the reliability methods. We demonstrated these approaches in a set of comparative studies on a CMOS inverter chain, a common emitter amplifier, and a Gilbert cell mixer circuit [6].

Finally, we developed an approach to handle highly-correlated outputs with large numbers of responses (e.g. voltage-time curves, current-voltage curves). We added a new response type to Dakota called "field response" and used this with a dimension reduction approach called Principal Component Analysis. Further, we augmented the standard PCA so that it can be used in a predictive mode, by making the coefficients of the principal components functions of the uncertain variables used to generate the ensemble.

In summary, this LDRD resulted in new algorithm capabilities in both Xyce and Dakota as well as practical methods for using the new analytic Xyce sensitivities in gradient-enhanced UQ methods in Dakota.

References

[1] Eric R. Keiter, Ting Mei, Thomas V. Russo, Richard L. Schiek, Peter E. Sholander, Heidi K. Thornquist, Jason C. Verley, and David G. Baur. Xyce Parallel Electronic Simulator: User's Guide, Version 6.1. Technical Report SAND2014-18007, Sandia National Laboratories, Albuquerque, NM, September 2014.

[2] Eric R. Keiter, Ting Mei, Thomas V. Russo, Richard L. Schiek, Peter E. Sholander, Heidi K. Thornquist, Jason C. Verley, and David G. Baur. Xyce Parallel Electronic Simulator: Reference Guide, Version 6.1. Technical Report SAND2014-18057, Sandia National Laboratories, Albuquerque, NM, September 2014.

[3] B. M. Adams, L. E. Bauman, W. J. Bohnhoff, K. R. Dalbey, J. P. Eddy, M. S. Ebeida, M. S. Eldred, P. D. Hough, K. T. Hu, J. D. Jakeman, L. P. Swiler, Stephens J. A., D. M. Vigil, and T. M. Wildey. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.0 users manual. Technical Report SAND2014-4633, Sandia National Laboratories, Albuquerque, NM, Updated May 2016. Available online from `http://dakota.sandia.gov/documentation.html`.

[4] Eric R. Keiter and Laura P. Swiler. Advanced uncertainty quantification methods for circuit simulation: Interim report. Technical Report SAND2014-20238, Sandia National Laboratories, Albuquerque, NM, Dec. 2014.

[5] Eric R. Keiter, Laura Swiler, Tom Ruso, and Ian Wilcox. Sensitivity analysis in xyce. Technical Report SAND2016-9437, Sandia National Laboratories, Albuquerque, NM, Sept. 2016.

[6] Eric R. Keiter, Laura P. Swiler, and Ian Wilcox. Gradient enhanced uncertainty quantification methods for circuit simulation. *IEEE Transactions of CAD of Integrated Circuits and Systems*, submitted, 2016.

[7] Eric R. Keiter, Laura P. Swiler, and Ian Wilcox. Gradient enhanced polynomial chaos methods for circuit simulation. *Proceedings of 11th International Conference On Scientific Computing In Electrical Engineering*, submitted, 2016.

[8] Frank Liu and Peter Feldmann. A time-unrolling method to compute sensitivity of dynamic systems. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pages 1–6, June 2014.

[9] Dale E. Hocevar, Ping Yang, Timothy N. Trick, and Berton D. Epler. Transient sensitivity computation for mosfet circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-4(4), October 1985.

[10] Ben Gu, K. Gullapalli, Yun Zhang, and S. Sundareswaran. Faster statistical cell characterization using adjoint sensitivity analysis. In *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE*, pages 229–232, Sept 2008.

[11] Y. Cao, S. Li, L. Petzold, and R. Serban. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint dae system and its numerical solution. *SIAM Journal on Scientific Computing*, 24(3):1076–1089, 2003.

[12] Eric T. Phipps and Roger P. Pawlowski. Efficient expression templates for operator overloading-based automatic differentiation. *CoRR*, abs/1205.3506, 2012.

[13] Silicon integration initiative (si2).

[14] Laurant Lemaitre, Colin McAndrew, and Steve Hamm. ADMS automatic device model synthesizer. *IEEE Custom Integrated Circuits Conference*, May 2002.

[15] G. Gildenblat, X. Li, W.Wu, H. Wang, A. Jha, R. van Langevelde, G.D.J. Smit, A.J. Scholten, and D.B.M. Klaassen. PSP: An advanced surface-potential-based MOSFET model for circuit simulation. *IEEE Transactions on Electron Devices*, 53(9):1979–1993, 2006.

[16] Harshit Agarwal, Sourabh Khandelwal, Juan Pablo Duarte, Yogesh Singh Chauhan, Ali Niknejad, and Chenming Hu. BSIM 6.1.0 MOSFET Compact Model Technical Manual. Technical report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720, 2014.

[17] Sourabh Khandelwal, Juan Duarte, Sriramkumar V., Navid Paydavosi, Darsen Lu, Chung-Hsun Lin, Mohan Dunga, Shijing Yao, Tanvir Morshed, Ali Niknejad, and Chenming Hu. BSIM-CMG 108.0.0 Multi-Gate MOSFET Compact Model Technical Manual. Technical report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720, 2014.

[18] Colin McAndrew, Mark Dunn, Shahriar Moinian, and Michael Schröter. VBIC95, The Vertical Bipolar Inter-Company Model. *IEEE Journal of Solid-State Circuits*, 31(10):1476–1483, 1996.

[19] The SiGe group at Auburn University. Mextram web site, 2015.

[20] Matthias Rudolph. Documentation of the fbh hbt model, 2005.

[21] M. Bucher, D. Diamantakos, A. Bazigos, and F. Krummenacher. Design-oriented characterization and parameter extraction methodologies for the EKV3 MOSFET model. In *NANOTECH 2007 Workshop on Compact Modeling*, May 2007.

[22] Ngspice. Ngspice, spice circuit simulator. `http://ngspice.sourceforge.net`, 2011.

[23] M. E. Brinson and S. Jahn. Qucs: A gpl software package for circuit simulation, compact device modelling and circuit macromodelling from dc to rf and beyond. *Int. J. Numer. Model.*, 22(4):297–319, July 2009.

[24] C.C. Mcandrew, Xin Li, I Stevanovic, and G. Gildenblat. Extensions to backward propagation of variance for statistical modeling. *Design Test of Computers, IEEE*, 27(2):36–43, March 2010.

[25] A. Haldar and S. Mahadevan. *Probability, Reliability, and Statistical Methods in Engineering Design*. Wiley, New York, 2000.

[26] M. S. Eldred, C. G. Webster, and P. Constantine. Evaluation of non-intrusive approaches for wiener-askey generalized polynomial chaos. In *Proceedings of the 10th AIAA Non-Deterministic Approaches Conference*, number AIAA-2008-1892, Schaumburg, IL, April 7–10 2008.

[27] B. M. Adams, L. E. Bauman, W. J. Bohnhoff, K. R. Dalbey, J. P. Eddy, M. S. Ebeida, M. S. Eldred, P. D. Hough, K. T. Hu, J. D. Jakeman, L. P. Swiler, Stephens J. A., D. M. Vigil, and T. M. Wildey. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.0 theory manual. Technical Report SAND2014-4253, Sandia National Laboratories, Albuquerque, NM, Updated May 2016. Available online from `http://dakota.sandia.gov/documentation.html`.

[28] Y. S. Chauhan, S. Venugopalan, M. A. Chalkiadaki, M. A. U. Karim, H. Agarwal, S. Khandelwal, N. Paydavosi, J. P. Duarte, C. C. Enz, A. M. Niknejad, and C. Hu. Bsim6: Analog and rf compact model for bulk mosfet. *IEEE Transactions on Electron Devices*, 61(2):234–244, Feb 2014.

[29] Arie Meir and Jaijeet Roychowdhury. Blast: Efficient computation of nonlinear delay sensitivities in electronic and biological networks using barycentric lagrange enabled transient adjoint analysis. In *DAC '12: Proceedings of the 2012 Design Automation Conference*, New York, NY, USA, 2012. ACM.

[30] B. M. Adams, L. E. Bauman, W. J. Bohnhoff, K. R. Dalbey, J. P. Eddy, M. S. Ebeida, M. S. Eldred, P. D. Hough, K. T. Hu, J. D. Jakeman, L. P. Swiler, Stephens J. A., D. M. Vigil, and T. M. Wildey. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.0 reference manual. Technical Report SAND2014-5015, Sandia National Laboratories, Albuquerque, NM, Updated May 2016. Available online from `http://dakota.sandia.gov/documentation.html`.

[31] I.T. Jolliffe. *Principal Component Analysis, 2nd ed.* Springer Series in Statistics, 2005.

[32] J.O. Ramsay and B.W. Silverman. *Functional Data Analysis, 2nd ed.* Springer Series in Statistics, 2005.

DISTRIBUTION:

1 MS 0899 Technical Library, 9536 (electronic copy)

This page intentionally left blank.


Sandia National Laboratories