Beyond Low-Order Statistical Response Surfaces: Latent Variable Regression for Efficient, Highly Nonlinear Fitting

Amith Singhee and Rob A. Rutenbar
Dept. of ECE, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213 USA
{asinghee,rutenbar}@ece.cmu.edu

Abstract

The number and magnitude of process variation sources are increasing as we scale further into the nano regime. Today's most successful response surface methods limit us to low-order forms -- linear, quadratic -- to make the fitting tractable. Unfortunately, not all variational scenarios are well modeled with low-order surfaces. We show how to exploit latent variable regression ideas to support efficient extraction of arbitrarily nonlinear statistical response surfaces. An implementation of these ideas called SiLVR, applied to a range of analog and digital circuits, in technologies from 90 to 45nm, shows significant improvements in prediction, with errors reduced by up to 21X, with very reasonable runtime costs.

Categories and Subject Descriptors
B.7.2 [Integrated Circuits]: Design Aids

General Terms
Algorithms, Design

Keywords
Response Surface, DFM, Dimensionality reduction, Regression

1. Introduction

Statistical manufacturing variations are of growing concern in the nanoscale regime. There is an urgent need to create robust and efficient models of the impact of these effects on circuit performance. Any solution to these problems must address three large challenges:

- **Dimensionality**: The number of sources of variations can be large. Even for a simple flip-flop, there can be over 50 sources, e.g., random dopant fluctuation (RDF), line edge roughness (LER), poly crystal orientation (PCO) [1], and gate-oxide thickness variation. For larger analog cells the dimensionality can easily be in the hundreds.
- **Large variations**: The relative effect of every variation source is becoming very large. Just considering RDF, predictions indicate that the standard deviation of the threshold voltage (V_t) can be 10% of the nominal V_t at the 70nm node [2], growing to 21% for a 25nm device[3], with 0.3V V_t. If other variations (LER, PCO) are considered, the deviation is even higher [1].
- **Nonlinearity**: Not all performance/variable relationships are simple. A good example is the relationship between device V_t in a flip-flop, and the flip-flop delay. Such nonlinearity is even more pronounced in the case of analog circuits.

Today's most successful Response Surface Models (RSMs) typically assume linear or quadratic model behavior. Linear models have been used extensively in yield optimization [4][5]. However, these models are weak at capturing nonlinear behavior, i.e., they are effective mainly when the variations are small enough to allow a linear approximation. To surmount these problems, quadratic models have been proposed [6]. The PROBE method of [7], and the method of [8] derive reduced rank quadratic models to reduce the problem dimensionality and the quadratic fitting cost. However, as we shall see, even quadratic models cannot always capture the nonlinearity seen in the presence of large manufacturing variations.

We suggest that *Latent Variable Regression* (LVR) techniques are an attractive approach in these scenarios [9]. Roughly speaking, these techniques iteratively extract the next "most important" statistical variable (*Latent Variable* or LV), and minimize the error in fitting the remainder of the unexplained performance variation. Hence, they directly reduce the problem dimensionality. However, these techniques need to be accompanied with flexible, but compact functional forms for the model, thus reducing *a priori* assumptions about the magnitude and behavior of the variations modeled.

The method has a long and interesting track record[1] -- much of it *outside* the realm of silicon applications, in areas ranging from chemometrics [10], to general statistics[11], to bioinformatics [12]. Many LVR methods still assume a linear relationship, or use a low-order nonlinear kernel to explain the assumed nonlinear relationships. Thus, our own interest is on LVR methods that support a more flexible nonlinear framework. Here, Baffi *et al.*, [13], and Malthouse *et al.* [14] are noteworthy. In addition to the single variable iterative extraction philosophy, these show how to use a neural network [15] to capture significant nonlinear behaviors. Of course, neural network models bring with a them host of ancillary problems related to model selection and validation. As a result, they have an (undeserved, in our opinion) reputation as the regressor of last choice in many scenarios. As we shall show, with proper attention in the fitting process, one can employ a very *small* network to mimic linear, quadratic, and more highly nonlinear behaviors, all in a single, *unified* LVR formulation. Moreover, we show a more unified mathematical framework that avoids the speed and reliability problems of [13], and the complexity burdens of [14].

In the rest of the paper, we show how to adapt LVR to the specific problems of modeling large manufacturing variations in circuit-level designs. To emphasize that our interests are in scaled silicon, we refer

1.And, unfortunately, a long and confusing series of related names, including: *Partial Least Squares* (PLS), *Canonical Correlation Regression* (CCR), *Projection on Latent Structures* (PLS), *Reduced Rank Regression* (RRR), and *Ridge Regression* (RR). These methods are similar, differing primarily in the formulation of their optimization problem; see [12][9].

to the technique as SiLVR (for **Si**licon **LVR**). Section 2 reviews the concepts of latent variables and latent variable regression. Section 3 describes SiLVR in detail. Section 4 presents our circuit testcases and experimental results. Section 5 offers concluding remarks.

2. Background: Latent Variable Regression

Suppose we need to model the relationship between k variables, $\boldsymbol{x} = \{x_1, \ldots, x_k\}$, and m circuit responses, $\boldsymbol{y} = \{y_1, \ldots, y_m\}$

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \tag{1}$$

and our model is

$$\hat{\boldsymbol{y}} = \boldsymbol{f}_m(\boldsymbol{x}) . \tag{2}$$

Also assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ have been scaled and centered to have zero mean and unit standard deviation. Classical LVR methods [9] start from a linear form for the model function $\boldsymbol{f}_m$. The standard linear model can be written as

$$\hat{\boldsymbol{Y}} = \boldsymbol{XB} \tag{3}$$

where $\boldsymbol{X}(n \times k)$ and $\hat{\boldsymbol{Y}}(n \times m)$ are matrices containing all the samples of $\boldsymbol{x}$ and the corresponding predictions, respectively. $\boldsymbol{B}(k \times m)$ is a matrix of regression coefficients. LVR modifies this model to

$$\hat{\boldsymbol{Y}} = \boldsymbol{XW}_a\boldsymbol{Z} \tag{4}$$

where $\boldsymbol{W}_a(k \times a)$ is a matrix to project the k-dimensional $\boldsymbol{x}$ vectors onto an a-dimensional space, where $a < k$. $\boldsymbol{Z}(a \times m)$ is a matrix of the regression coefficients for the projected vectors. Hence, the dimensionality of the input space is reduced from k to a and then a smaller linear model is fit from this projected space to the output. The projected vectors ($\boldsymbol{XW}_a$) are called the *latent variables (LVs)*. Comparing (4) to (3), we can see that

$$\boldsymbol{B} = \boldsymbol{W}_a\boldsymbol{Z} \tag{5}$$

the new constraint being that $\boldsymbol{B}$ is not of full rank: $rank(\boldsymbol{B}) = a < k$.

As mentioned earlier, the primary difference in the theoretical framework of the LVR methods is the problem that is optimized to obtain the regression coefficients in $\boldsymbol{B}$. If we pick one column $\boldsymbol{w}_i$ of the projection matrix $\boldsymbol{W}_a$, one possible problem formulation is

$$\max_{\boldsymbol{w}_i} \boldsymbol{w}_i^T\boldsymbol{X}^T\boldsymbol{YY}^T\boldsymbol{Xw}_i \text{, s.t. } \boldsymbol{w}_i^T\boldsymbol{X}^T\boldsymbol{Xw}_i = 1 \text{ , } \boldsymbol{w}_i^T\boldsymbol{X}^T\boldsymbol{Xw}_j = 0, i \neq j \tag{6}$$

where $\boldsymbol{Xw}_i$ are the LVs. The objective function is proportional to the sum of squared covariances of the LV and the outputs. It can be shown [16] that optimizing this objective function for each column of $\boldsymbol{W}_a$ is equivalent to optimizing the following objective function

$$\min_{\boldsymbol{W}_a, \boldsymbol{Z}} \|\boldsymbol{Y} - \boldsymbol{XW}_a\boldsymbol{Z}\|^2 ; \tag{7}$$

that is, we compute the projection vectors, $\boldsymbol{W}_a$, and the reduced regression coefficient matrix, $\boldsymbol{Z}$, that minimize the mean-squared error of the predictions, under the reduced rank constraint. This is exactly what we want to achieve from our modeling attempts. This LVR method is commonly known as *Reduced Rank Regression* [11]. For a comparison with other LVR methods, refer to [9].

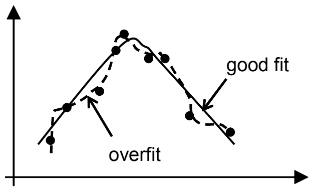

FIGURE 1. The data has a triangular relationship. The good fit is al
to capture the relationship well, while overfitting unnecessarily

The problem of modeling nonlinear behavior, however, remains unsolved by these classical LVR techniques. Kernel-based methods try to address this issue by using the well-known "kernel trick": map the inputs ($\boldsymbol{x}$), using *fixed* nonlinear kernels ($\boldsymbol{f}_K(\boldsymbol{x})$, e.g., a quadratic [8]), to a higher dimensional space, and then create a reduced linear model to the output $\boldsymbol{y}$. This has severe limitations: it increases the problem dimensionality before reducing it, and, more importantly, assumes a *known* nonlinear relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$. Baffi [13] proposes adapting these ideas to use a more flexible neural network [15] formulation, but the model fitting is very slow (a two-step process iterates between model fitting and LV estimation) and unreliable (due to weak convergence of this two-step iteration). Malthouse [14] takes this further, but produces a very complex neural network model (indeed, three separate neural networks, connected together) that can cause undesirable overfitting, especially for small training datasets, and has a large number of unknowns to fit. Also, both these methods solve a problem different from (7). The PROBE method of [7] also uses a projection-based approach, but is restricted to a quadratic form, and also does not explicitly solve for the latent variables, one by one.

SiLVR adopts the central philosophy of the LVR methods: 1) find a projection to a single latent variable (LV); 2) fit a *flexible* nonlinear functional form to the relationship between the LV and the output 3) repeat for the next LV. We shall often refer to such a LV-based model as a *reduced model*. We will now explain SiLVR in detail.

3. SiLVR: a New Latent Variable Regression Strategy

We use a carefully constructed feedforward neural network as our reduced model. Neural networks have remarkable flexibility in modeling strong nonlinearity, but, this flexibility needs to be carefully controlled: if the network is overly complex, or not trained carefully, it can overfit the data: Fig. 1 shows an example. If the network is carelessly constructed it can overuse its modeling flexibility and lose track of the primary relationship (triangular in this case). Such overfitting can be avoided by keeping the network structure as small as possible and intelligently guiding the training process. We will pay careful attention to these issues while constructing our model.

3.1 Structure of the Nonlinear Model

Fig. 2 shows the structure of our model: it takes in all k inputs and predicts one output. In the case of multiple outputs ($m > 1$), we handle each output independently, one at a time. Hence, for simplicity, we will use a single output variable y in the following explanation:

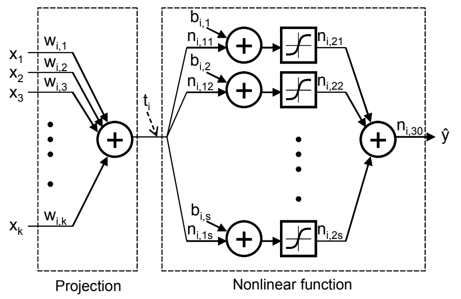

FIGURE 2. Neural network model structure for extracting one latent variable (t_i) and predicting one output

the same methodology is applied for modeling each output. The model has two main parts:

1) **Linear projection** from the input ($\boldsymbol{x} = \{x_1, \ldots, x_k\}$) to a latent variable ($t_i$). $\boldsymbol{w}_i = \{w_{i,1}, \ldots, w_{i,k}\}$ is the projection vector. Hence, the i-th LV is given as

$$t_i = \boldsymbol{x}^T \boldsymbol{w}_i = \sum_{j=1}^{k} x_j w_{i,j} \tag{8}$$

$\boldsymbol{w}_i$ are chosen, so that the projection is maximally *aligned* with y.

2) **A nonlinear function** predicting the output y from t_i, composed of a fixed number (s) of sigmoid functions. The sigmoid function used is $tanh(x)$, which has an output range of [-1,1]. Such a functional form provides welcome flexibility for modeling large varieties of nonlinear behavior. The variables $n_{i,j}$ are internal weights of the network and b_i are the bias values. The values for these weights and biases are chosen to minimize the error between the network prediction $\hat{y}$ and the actual value y.

This structure has been carefully selected to closely match the intuition of LVR, and minimize structural overfitting. We fit the two parts simultaneously, by training the entire network all at once. The fitting problem can be cast as an optimization problem:

$$\min_{\boldsymbol{w}_i, \boldsymbol{n}_i, \boldsymbol{b}_i} \sum_{j=1}^{N} \|e_j\|^2 \text{, where } e_j = y_j - \hat{y}_j \tag{9}$$

where N is the number of training samples. Once a network has been trained to extract one LV, the y values are "deflated" to remove the modeled component:

$$y_j = y_j - \hat{y}_j \tag{10}$$

and (9) is solved again to extract the next LV, and so on. Compared to [13], we have included the projection as a part of our model, and use a *non-iterative* training stage to solve the fitting problem. This gives us significantly improved convergence and training speed.

Note that the number of model parameters to solve for is $k + 3s + 1$, per LV, where s is the number of neurons in the sigmoid layer. s can be kept very small (12 for our experiments). Hence, for each output, the number of model parameters is $O(k)$, with the proportionality constant equal to the number of LVs used. We shall see that the number of LVs needed is very small in most cases (within 2). Hence, this model is very compact and yet, extremely flexible.

3.2 Training the Model

Training the model involves solving the nonlinear optimization problem (9). Of course, we would like the solution method to be fast. However, there is the added, and arguably more important, requirement of the solution being generalizable: the model should not overfit the training data and should have low prediction errors for new samples. Overfitting would make the model practically useless. We use several techniques for fast training and robust model generalization, standard in the data mining community.

Projection vector initialization. The unknown parameters of the model have to be assigned some initial values to start the optimization from. An intelligent guess about the approximate value of these parameters can significantly help the optimizer, since it would start closer to the global minimum. For the projection weights $\boldsymbol{w}_i$, we use Spearman's Rank Correlation (r_s)[17] for this guess. Suppose R_l and S_l are the ranks of corresponding values of some input coordinate x_j and the output, respectively, then

$$r_s^j = \frac{\sum_l (R_l - \bar{R})(S_l - \bar{S})}{\sqrt{\sum_l (R_l - \bar{R})^2} \sqrt{\sum_l (S_l - \bar{S})^2}} \tag{11}$$

is their rank correlation. This measure of correlation is more robust than linear Pearson's correlation, in the presence of non-linear relationships in the data. Hence, r_s^j is a measure of how strong the relationship between x_j and y is, or how important x_j is for predicting y. Recognizing this, we can initialize

$$\boldsymbol{w}_i = \boldsymbol{r}_s / \|\boldsymbol{r}_s\| \text{, where } \boldsymbol{r}_s = \{r_s^1, \ldots, r_s^k\} \tag{12}$$

The other parameters, $\boldsymbol{n}_i, \boldsymbol{b}_i$ are initialized using the Nguyen-Widrow method [18].

Nonlinear optimization. We use the Levenberg-Marquardt (LM) algorithm for training the network [19]. LM is a combination of the Steepest Descent (SD) and Gauss-Newton (GN) algorithms [20]. SD takes steps along the direction of maximum slope of the objective function. GN uses a linear approximation of the local region around the current point to estimate the minimum point, to use for the next iteration. GN is a simplified version of the Newton's method, which has a very desirable quadratic convergence close to the minimum point. However, for non-convex surfaces, GN can get lost far from the global minimum. In such a situation SD is a better choice. The LM algorithm combines the SD and GN into the following stepping formula

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - (J(\boldsymbol{x}^n)^T J(\boldsymbol{x}^n) + \mu I)^{-1} J(\boldsymbol{x}^n)^T e(\boldsymbol{x}^n) \tag{13}$$

Here $\boldsymbol{x}$ is the vector of unknowns being solved for ($\{\boldsymbol{w}_i, \boldsymbol{n}_i, \boldsymbol{b}_i\}$). $\boldsymbol{e}(\boldsymbol{x}^n) = \{e_j\} = \{y_j - \hat{y}_j\}$ is the vector of all the error values at the current point, and $J(\boldsymbol{x}^n)$ is the Jacobian of this error vector at the current point. μ is a parameter that is automatically adjusted during runtime. If μ is small, LM behaves like the GN algorithm and if it is large, it behaves like SD. In spite of its heuristic nature, this algorithm works significantly better than just SD or GN in practice [19].

Bayesian regularization. The LM algorithm alone can often converge to solutions that correspond to overfitting of the training data, since its sole aim is to minimize the training error (9). To guide the algorithm to solutions that are more generalizable, we use the standard technique of *regularization* [15]. For this, we augment the objective (9) as follows:

$$\min_{\boldsymbol{w}_i, \boldsymbol{n}_i, \boldsymbol{b}_i} \alpha E_D + \beta E_P \text{, where } E_D = \sum_{j=1}^{N} \|e_j\|^2 \text{, and}$$

$$E_P = \boldsymbol{w}_i^T \boldsymbol{w}_i + \boldsymbol{n}_i^T \boldsymbol{n}_i + \boldsymbol{b}_i^T \boldsymbol{b}_i \tag{14}$$

We have added another term E_P, the sum of squares of the network parameters, to the original objective E_D. Keeping this term small encourages a smoother (less overfitting) response from the network [21]. The parameters α and β can be automatically determined using a Bayesian framework which maximizes the posterior probability of the values of α and β [21], for the given training data. This framework fits very well with the LM algorithm: it requires the Hessian of the objective function, which is already provided by the LM algorithm. For a detailed explanation, see [22].

K-fold cross-validation. *Cross-validation* (CV) is a standard technique to improve model generalizability (reduce overfitting) [15]. The basic concept is as follows. The training set is divided randomly into K separate sets of equal size. Then, K training runs are performed, each time leaving out one set as the testing set, and the average testing error over all the K runs is computed. This entire procedure is repeated for each candidate model structure, and the

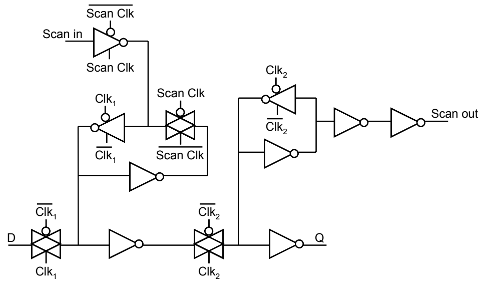

FIGURE 3. Master-Slave Flip-Flop with Scan Chain component.

one with the lowest testing error is selected. SiLVR has a fixed small neural net model structure. Thus, we use CV for a different purpose: to help select the network parameters which give the best predictions. One set of K training runs is run, and the model parameters which give the lowest testing error are selected. The value of K used is 5. This has the further advantage of rejecting solutions that are stuck in local minima.

4. Experimental Results

We have implemented SiLVR in Matlab. In this section we present our experimental results for three testcases, each representing a different family of circuit behavior: 1) Master Slave Flip-Flop with scan chain, 2) two-stage RC-compensated opamp and 3) sub-1V bandgap voltage reference in CMOS. The number of process parameters range from 13 to 122 (both inter-die and intra-die). SiLVR is able to extract good estimates of the LVs, along with the accompanying model, using 1,000 training samples for each case, generated using standard Monte Carlo sampling. We also present comparisons with a straightforward Matlab implementation of an optimal reduced quadratic model, using the PROBE [7] algorithm. The best PROBE results (up to rank 10) are used for graphical comparisons. All models are evaluated on a separate test set of 10,000 Monte Carlo samples. Samples where the circuit does not function, are not used for modeling, but no extra samples are simulated.

4.1 Master-Slave Flip-Flop with Scan Chain

The first testcase is a commonly seen Master-Slave Flip-Flop with scan chain (MSFF) shown in Fig. 3. The design has been implemented using the 45 nm CMOS Predictive Technology Models of [23]. Variations considered are Random Dopant Fluctuation (RDF) for all transistors and one global gate-oxide (t_{ox}) variation. RDF is modeled as normally distributed threshold voltage (V_t) variation:

$$\sigma(V_t) = 0.0135 V_{t0} / \sqrt{WL} \text{ where W,L are in } \mu\text{m} \qquad (15)$$

V_{t0} is the nominal threshold voltage. This results in 30% standard deviation for a minimum-sized transistor. This is large for current technologies: we want to make sure SiLVR is powerful enough for future technologies too. The t_{ox} standard deviation is taken as 2%.

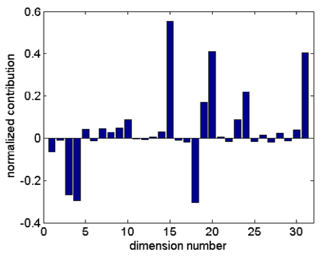

FIGURE 4. Normalized projection vector for the first LV of the MSFF delay

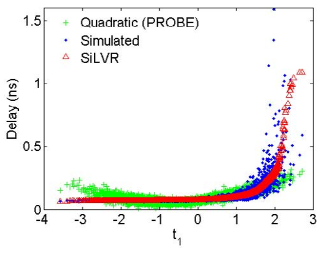

FIGURE 5. Simulated and predicted output, plotted against first LV of MSFF delay

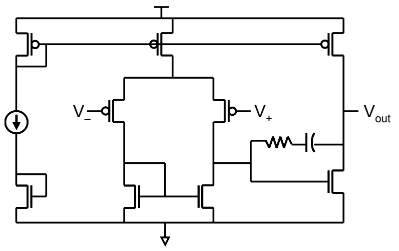

FIGURE 6. Two-stage RC-compensated operational amplifier

The number of input dimensions is 31 and there is one output: the clock-output delay, τ_{cq}.

Fig. 4 shows the projection vector $\boldsymbol{w}_1$ for the first extracted latent variable t_1, and Fig. 5 plots the simulated and predicted delay values against t_1. The latter shows the predictions from SiLVR and also from the best reduced quadratic model. We can clearly observe two things: 1) only 6-8 out of 31 input dimensions (corresponding to transistors in the circuit) affect the output, and 2) SiLVR performs much better than a simple quadratic model. Table 2 compares the errors quantitatively with increasing number of LVs/rank: the best average error is reduced by 2.5x: from 16.3% (PROBE) to 6.4%. Both PROBE and SiLVR perform best with only *one* LV (reduced rank), beyond which they start overfitting.

4.2 Two-Stage RC-Compensated Opamp

This next testcase [24], shown in Fig. 6, is representative of a large class of circuits in the *analog* domain: amplifiers. We test SiLVR on the DC, AC and transient characteristics of the opamp. The opamp has been implemented using models from the Cadence 90nm GPDK library. RDF on all transistors is considered, along with a global t_{ox} variation, and variations on the passives and the current source. All variations are assumed to be normally distributed. The V_t standard deviation is (about 18% of nominal V_t)

$$\sigma(V_t) = 5mV / \sqrt{WL} \text{ where W,L are in } \mu\text{m} \qquad (16)$$

$\sigma(t_{ox})$ is taken to be 2% and each passive and current component has its own normally distributed variation, with a standard deviation of 5%. The input dimensionality is 13, and there are 5 outputs: 1) DC gain, 2) unity gain frequency (UGF), 3) phase margin (PM),

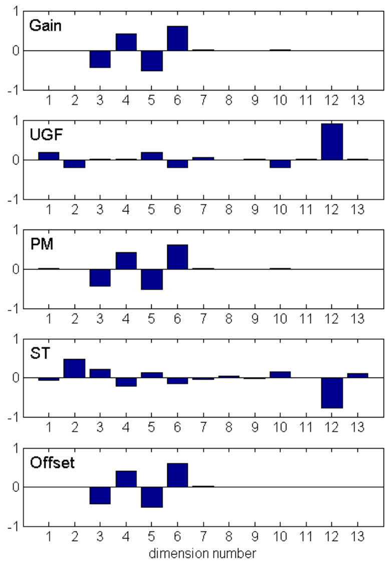

FIGURE 7. Normalized projection vector for the first LVs of the Opamp metrics: we can see the strong relationship between Gain, PM and offset

4) settling time (ST), and 5) DC offset.

Fig. 7 shows the projection vector w_1 for the first extracted LV t_1, for every performance metric, and Fig. 8 plots the simulated and predicted output values against the corresponding t_1. It is clear from these plots that SiLVR is able to extract the LVs effectively and is also sufficiently flexible to explain the circuit behavior. The best (up to rank 10) reduced quadratic model (PROBE) can neither provide explicit LVs, nor explain the actual circuit behavior in many cases. Having the explicit projections provides deep insight into the circuit behavior. First, we can actually *see* the behavior clearly (e.g. step-shaped for offset), removing any need for guess-work. Also, if we look at the projection vectors for Gain, PM and offset in Fig. 7, we can immediately see that these outputs depend almost identically on the same parameter subset (parameters 3-6): these are the driver and load devices in the input differential amplifier. Hence, they are also strongly correlated. This is confirmed by plotting all three simulated metrics against the *gain* LV (Fig. 9). This is where the power of SiLVR is really evident. Table 1 compares the rank correlation (r_s) and linear correlation (r_p) among these metrics, with the dot product of the corresponding normalized LVs (r_l). We refer to this dot product as *Input Referred Correlation* (IRC). We can see that r_s performs better than r_p, but both completely fail to predict the relationship between gain and offset, and gain and PM, while r_l, succeeds.

	$\lvert r_s \rvert$	$\lvert r_p \rvert$	r_l
gain-pm	0.986	0.871	1.000
pm-offset	0.073	0.064	1.000
gain-offset	0.154	0.093	1.000

TABLE 1. Rank and linear correlation compared with IRC as a measure of correlation between strongly correlated Opamp metrics.

Table 2 compares the average errors of PROBE and SiLVR. For the strongly nonlinear metrics (Gain, PM and offset), PROBE has very large errors: 16% to 51% for the best case. SiLVR reduces these by

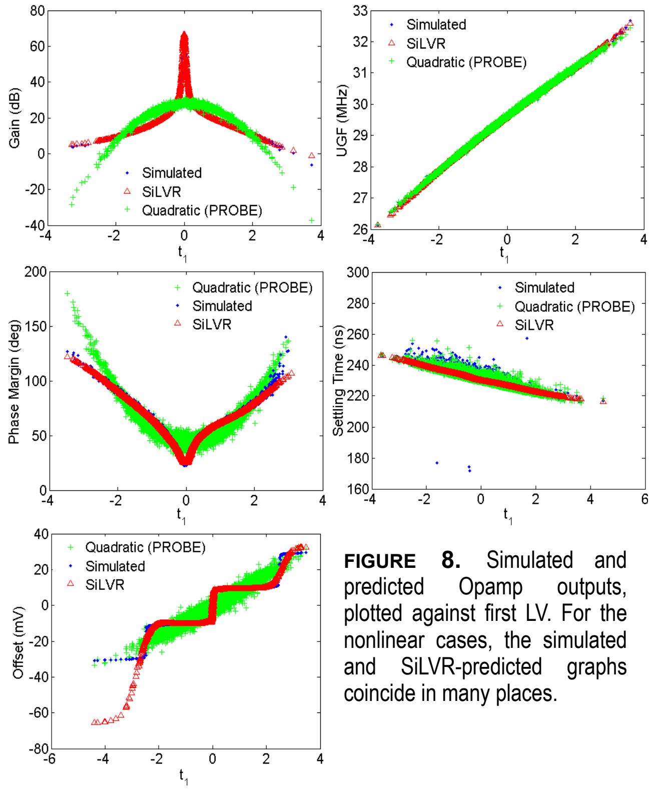

FIGURE 8. Simulated and predicted Opamp outputs, plotted against first LV. For the nonlinear cases, the simulated and SiLVR-predicted graphs coincide in many places.

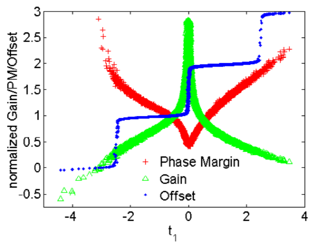

FIGURE 9. Simulated Opamp gain, phase margin and offset plotted against the LV for gain, showing strong correlation among the three.

5x to 21x, using one LV. For UGF and settling time, PROBE shows lower errors, but the SiLVR error is already quite low (<0.5%).

4.3 Sub-1V CMOS Bandgap Voltage Reference

Fig. 10 shows a low-voltage CMOS Bandgap Reference circuit [25]. This bandgap is able to provide reference voltages that are less than 1V, and is built using standard CMOS technology. This circuit was chosen for its relevance in today's low-voltage designs, and also because it has very high dimensionality (122) and strongly nonlinear behavior. The opamp used is the same as in Section 4.2. The circuit has 101 diodes. The transistor device and variation models are the same 90nm CMOS as the Opamp. RDF in the diodes is modeled as normally distributed variations on the saturation current, with standard deviation of 10%. Each resistor and capacitor has its own normally distributed variation source, with a standard deviation of 5%. There are a total of 121 local variation parameters and one global t_{ox} variation. In this case, we measure two metrics: 1) output voltage (V_{ref}), and 2) dropout voltage (V_{do}). V_{do} is the difference between the supply voltage and V_{ref}, when V_{ref} falls by 1% of its nominal value (0.6V): lower V_{do} implies a circuit more robust in the presence of supply variations.

Fig. 11 shows w_1: the 122-dimensional projection vector for the first LV t_1, and Fig. 12 plots the simulated and predicted V_{ref} and V_{do} against their corresponding t_1 s. PROBE performs well for the linearly varying V_{ref}, but completely breaks down for the nonlinearly varying V_{do}. SiLVR is able extract a good estimate of this strong nonlinear behavior. Fig. 13 shows the simulated and predicted V_{do} values against the first *two* LVs (trained using 10,000 points to make the graphical illustration visually obvious). Even though we started with a large dimensionality of 122, only 2 LVs can still explain most of the behavior. Also, the component of normalized LV1 along normalized LV2 is only 1.2e-3, meaning that they are almost orthogonal. This implies that SiLVR can extract all the information from LV1 before looking at LV2.

Table 2 compares the average prediction error of SiLVR with that of the reduced quadratic model (PROBE). In this case, SiLVR performs better for both linear (V_{ref}) and nonlinear behavior (V_{do}). Here too, we can see that improvement achieved by using 2 LVs for

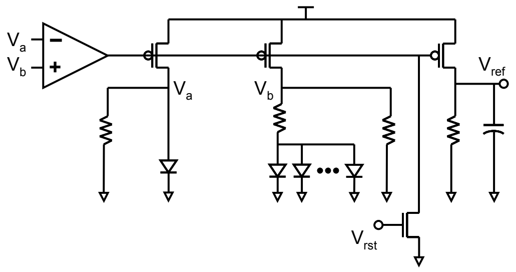

FIGURE 10. Low-voltage CMOS bandgap voltage reference circuit, with a parameter space of 122 dimensions.

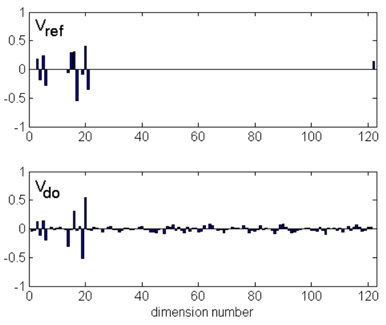

FIGURE 11. Normalized projection vector for the first LVs of the voltage reference metrics.

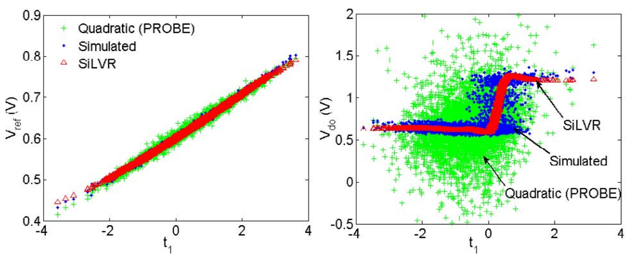

FIGURE 12. Simulated and predicted voltage reference outputs. The quadratic method breaks down for nonlinear behavior.

V_{do} : the SiLVR error decreases from 11.1% to 10.4%.

We also note that the training times to build each LV are quite reasonable, even with the complex cross-validation strategy to improve neural network robustness: each LV requires 13-24 CPU seconds of Matlab computation. This is especially attractive for the for higher dimensional testcases like the voltage reference.

5. Conclusions

SiLVR is a fast and flexible statistical response surface modeling framework, that can efficiently handle the large and highly nonlinear effects from process variations in nanoscale technologies. It makes few assumptions about the problem size or nonlinear functional form, unlike most of the RSM tools available today. It provides the user with explicit latent variables, that maximally explain the observed circuit performance, and deep insight into the circuit behavior. When the relationship we seek to extract is itself of a simple form (e.g., linear, or nearly so), our simple neural network forms mimic a low-dimensional regressor. But when the relationship is more nonlinear, we can explain it with very few extracted latent variables. We believe the SiLVR methodology is an attractive basis for future work on nonlinear yield optimization strategies.

References

[1] M. Hane, T. Ikezawa, T. Ezaki, "Atomistic 3D process/device simulation considering gate line-edge roughness and poly-Si random crystal orientation effects", *IEEE Internat. Electron Devices Meeting*, 2003.

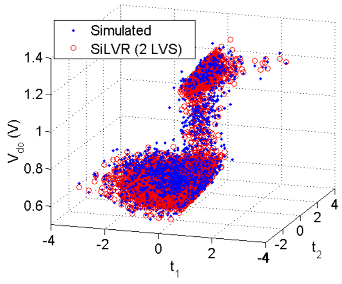

FIGURE 13. Simulated and predicted V_{do} against the first two LVs

[2] T. Ezaki, T. Ikezawa, M. Hane, "Investigation of realistic dopant fluctuation induced device characteristics variation for sub-100nm CMOS by using atomistic 3d process/device simulator", *IEEE IEDM*, 2002.
[3] D.J. Frank et al.,"Monte Carlo modeling of threshold variation due to dopant fluctuations", *Symp. VLSI Tech.*, 1999.
[4] H. Chang, S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal", *ICCAD*, 2003.
[5] Z. Wang, S. Director, "An efficient yield optimization method using a two step linear approximation of circuit performance", *EDAC*, 1994.
[6] A. Dharchoudhury, S.M. Kang, "Worst-case analysis and optimization of VLSI circuit performances", *IEEE Trams. CAD*, 14(4), 1995.
[7] X. Li et al., "Projection-based performance modeling for inter/intra-die variations", *IEEE/ACM ICCAD*, 2005.
[8] Z. Feng, P. Li, "Performance-oriented statistical parameter reduction of parameterized systems via reduced rank regression", *ICCAD*, 2006.
[9] A.J. Burnham, R. Viveros, J.F. MacGregor, "Frameworks for latent variable multivariate regression", *J. of Chemometrics*, 10, 1996.
[10] S. Wold, M. Sjöström, L. Eriksson, "PLS-regression: a basic tool of chemometrics", *Chemometrics and Intelligent Lab. Sys.*, 58, 2001.
[11] P.T. Davies, M.K-S. Tso, "Procedures for reduced-rank regression", *Applied Statistics*, 31(3), 1982.
[12] A-L. Boulesteix, K. Strimmer, "Partial least squares: a versatile tool for the analysis of high-dimensional genomic data", *Brief. in Bioinf.*, 2006.
[13] G. Baffi, E.B. Martin, A.J. Morris, "Non-linear projection to latent structures revisited (the neural network PLS algorithm)", *Computers Chem. Engg.*, 23(9), 1999.
[14] C. Malthouse, A.C. Tamhane, R.S.H. Mah, "Nonlinear partial least squares", *Computers Chem. Engg.*, 21(8), 1997.
[15] B.D. Ripley, "Pattern recognition and neural networks", Cambridge University Press, 1996.
[16] K.E. Muller, "Relationships between redundancy analysis, canonical correlation, and multivariate regression", *Psychometrika*, 46(2), 1981.
[17] G.E. Noether, "Introduction to Statistics: The Nonparametric Way", Springer, 1990.
[18] D. Nguyen, B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights", *Int. Joint Conf. Neural Nets.*, 1990.
[19] M.T. Hagan, M.G. Menhaj, "Training feedforward networks with the Marquardt algorithm", *Trans. Neural Nets.*, 5(6), Nov. 1994.
[20] D.P. Bertsekas, "Nonlinear programming", 2nd ed, Athena Scientific, 1995.
[21] D.J.C. MacKay, "A practical Bayesian framework for backpropagation networks", *Neural Comput.*, 4(3), 1992.
[22] F.D. Foresee, M.T. Hagan, "Gauss-Newton approximation to Bayesian learning", *Int. Conf. Neural Nets.*, 1997.
[23] W. Zhao, Y. Cao, 'New Generation of Predictive Technology Model for sub-45 Design Exploration", *ISQED*, 2006.
[24] D. Johns, K. Martin, "Analog Integrated Circuit Design", Wiley, 1996.
[25] H. Banba et al., "A CMOS Bandgap Reference Circuit with Sub-1-V Operation", *IEEE J. Solid State Cir.*, 34(5), pp. 670-674, 1999.

Rank	MSFF delay		Opamp Gain		UGF		PM		Settling Time		Offset		Bandgap V_{ref}		V_{do}	
(LV)	SiLVR	PROBE	SiLVR	PROBE	SiLVR	PROBE	SiLVR	PROBE	SiLVR	PROBE	SiLVR	PROBE	SiLVR	PROBE	SiLVR	PROBE
1	**6.41**	**16.3**	**1.69**	**35.7**	**0.061**	**0.042**	**1.90**	**16.7**	**0.507**	**0.248**	**10.0**	**51.7**	**0.278**	**0.639**	**11.1**	**35.4**
2	7.24	19.7	1.74	36.4	0.063	0.022	1.97	16.8	0.517	0.240	10.2	52.0	0.341	0.707	10.4	39.9
3	8.15	21.3	1.73	36.9	0.066	0.013	1.94	16.9	0.528	0.243	10.5	52.4	0.374	0.726	11.9	41.9
4	8.17	22.2	1.66	37.2	0.068	0.010	1.99	16.9	0.531	0.244	11.1	55.1	0.375	0.736	11.9	42.3
6	7.79	23.2	1.72	37.0	0.072	0.009	2.03	16.9	0.552	0.245	12.2	55.5	0.374	0.738	11.9	42.4
8	8.08	24.1	1.78	37.1	0.075	0.008	2.11	16.9	0.577	0.245	13.4	55.7	0.403	0.738	12.2	42.4
10	8.60	24.7	1.83	37.1	0.078	0.008	2.16	16.9	0.603	0.245	15.3	55.8	0.459	0.738	11.9	42.4

TABLE 2. Average *percentage* error on a test set of 10,000 Monte Carlo samples