JAISCR, 2011, Vol.1, No.2, pp. 115–134

STATISTICAL PARAMETER IDENTIFICATION OF ANALOG INTEGRATED CIRCUIT REVERSE MODELS

Bruno Apolloni[1], Simone Bassis[1], Cristian Mesiano[1], Salvatore Rinaudo[2], Angelo Ciccazzo[2], Angelo Marotta[2]

[1]*Dipartimento di Scienze dell'Informazione*
via Comelico 39/41, 20135 Milano, Italy
e-mail:{apolloni,bassis,mesiano}@dsi.unimi.it

[2] *STMicroelectronics*
Stradale Primo Sole 50, 95121 Catania, Italy
e-mail:{salvatore.rinaudo,angelo.ciccazzo,angelo.marotta}@st.com, Italy

Abstract

We solve the manufacturing problem of identifying the model statistical parameters ensuring a satisfactory quality of analog circuits produced in a photolithographic process. We formalize it in a statistical framework as the problem of inverting the mapping from the population of the circuit model parameters to the population of the performances. Both parameters and performances are random. From a sample of the latter population we want to identify the statistical features of the former that produce a performance distribution complying with production samples. The key artifact of the solution method we propose consists of describing the above mapping in terms of a mixture of granular functions, where each is responsible for a fuzzy set within the input-output space, hence for a cluster therein. The way of synthesizing the whole space as a mixture of these clusters is learnt directly from the examples. As a result, we have an analytical form of the mapping that approximates complex Spice models in terms of polynomials in the model parameters, and an implicit expression of the distribution law of the induced performances that allows a relatively quick and easy management of the model distribution statistical parameters. This flows into a semiautomatic procedure managing an adaptive composition of different granular modules to cope with the circuit peculiarities. We check the method both on real world manufacturing problems and on ad hoc benchmarks.

1 Introduction

A major challenge posed by new deep-submicron technologies is to design and verify integrated circuits so as to obtain a high fabrication yield, i.e. a high proportion of produced circuits that function properly. By contrast, with a further shrinking of process technology, the on-chip variation is getting worse for each technology node as a result of a prevalence of random effects over the designed functionality. The main sources of these effects are the random distributions of discrete dopants and charged defects, the line edge roughness of the photo resist and the granularity of the materials [1, 2]. Therefore, production yields and circuit figures of merit (such as performance, power, and reliability) have become extremely sensitive to incontrollable statistical process variations. These random and systematic defects as well as parametric process variations have a big influence on the yield of the manufactured circuits, with the consequence of frequent respinning of the whole development and manufacturing chain. This leads to high costs of multiple manufacturing runs and entails extremely high risks of missing a given market window. One way to overcome these draw-

backs is to implement the DFM/DFY paradigm [3] where Design for Manufacturability mates Design for Yield to form a synergistic manufacturing chain to be dealt with in terms of relationships between the statistical circuit parameters matching the production constraints and performance indicators ensuring correctly functioning dies. Model parameters take an intermediate location in this chain, so representing a target of the production process and the root of the circuit performance. In greater detail, the first requirement for planning circuits is the availability of a model relating input/output vectors of the function implemented by the circuit. Meeting this requirement usually involves two phases aimed at searching for a couple of analytic relations: the first between model parameters and circuit performances; and the second, which is connected to the process engineer's experience, linking both design and physical circuit parameters as possibly obtained during production. The final goal is model identification, in terms of designating the input and output parameter values of the analytical relations. Thus, model parameters identification represents a critical instance within the general task of the Statistical Modeling. It is crucial during the circuit design in order to estimate and take into account the fluctuations that might characterize the electrical behavior of the integrated circuits and the manufacturing non-idealities. The objective is to allow the designer to get a clear picture of how the latter react to the model parameters in the actual production process and, consequently, to grasp a guess on their variation impact.

We solve this identification problem on the basis of a huge amount of experimental data collected during the manufacturing process. In line with previous works in the literature [4–7], we frame the problem in terms of random variable transforms - from model parameter space to performance space - with the aim of identifying the best distribution law of the former that induces a satisfactory fulfillment of the performance requisites. In the light of the recent literature [8], we abandon the simplifying independence hypothesis between the random components of each space (where the mapping from the former to the latter is ruled by Spice models [9]). Our specific contributions in this framework are the following:

- our main artifact is to handle the mapping in terms of a granular construct based on a fuzzy partition of the model parameter space. This allows a manageable interpolation of both the Spice functions through polynomials and the performances' joint distribution law as a mixture of the distribution laws [10] affecting the single fuzzy sets;
- moreover, the combined use of function interpolation and statistical parameter identification allows us to be accurate exactly where necessary, i.e. in the circuit operational domain;
- thirdly, we make use of an adaptive composition of different granular methods in response to the different morphology of the function landscape we want to identify.

In this way we get both a quick and accurate interpolation of the Spice functions and a satisfactory identification of the statistical parameters that proves faster and more efficient than other procedures in the literature [11, 6, 12].

The paper is organized as follows. In Section [2] we explain our procedure to identify the approximation of the Spice model and its adaptation to the experimental data. Section [3] will be devoted to describing the procedure as a whole, with particular emphasis on the choices left open to the user, which improve flexibility while also allowing recovering routines. In Section [4]we toss the benefits and limits of this procedure on specific benchmarks in comparison to other methods. In the concluding section we outline a way to improve these results on the basis of the interpolated performances distribution *that is not constrained to be Gaussian* [13].

2 The Model Identification Tools

We formalize the modeling problem in terms of a mapping g from a random vector $X = (X_1, \ldots, X_n)$ [1] describing what is commonly denoted as model parameters, to a random vector $Y =$

[1] By default, capital letters (such as X, Y) will denote random variables and small letters (x, y) their corresponding realizations; vectorial versions of the above variables will be denoted through bold symbols ($\mathbf{X}, \mathbf{Y}, \mathbf{x}, \mathbf{y}$). Moreover, the set the realizations belong to will be described through gothic symbols ($\mathfrak{X}, \mathfrak{Y}$).

$(Y_1,\ldots,Y_t)$, representing the performances. The statistical features of X, such as mean, variance, correlation, etc., constitute its parameter vector θ_X, henceforth considered to be the statistical parameter of the input variable X. Idem for the parameter vector θ_Y. Hence, $Y = g(X) = (g_1(X),\ldots,g_t(X))$, and we look for a vector θ_X such that the corresponding θ_Y characterizes a performance population where $P(Y \in \widehat{D}_Y) = \alpha$, having denoted with $\widehat{D}_Y$ the domain spanned by the performances which have been measured in a satisfactory production process and with α a set probability value. This goal entails the inverse problem – find the θ_X producing the wanted Y population – which we solve in the statistical environment generated by the observed performances through the following three conceptual blocks (to be explored in greater depth in the next sections):

I. approximate g through an accurate enough function that is easy to invert. We will do so by interpolating g in terms of a polynomial in x;

II. statistically invert this function. Denoting with E and V the mean vector and variance/covariance matrix operators, respectively, we aim in this step either to have both $E[g(X)]$ and $V[g(X)]$, as representatives of θ_Y, very close to the estimate $\widehat{\theta}_Y$ taken from the measured performances, or to reproduce other features of these data;

III. check that the solution of the inversion problem actually generates a population satisfying the condition $P(Y \in \widehat{D}_Y) = \alpha$. We cannot expect the above probability to be exactly met. Rather, the proximity to α represents an indicator of the identification accuracy, in the worst case asking for a better modeling of the circuit at hand.

2.1 Interpolating a Spice Model

The most common tool for modeling an analog circuit is represented by the Spice simulator [14]. It consists of a program which, having in input a textual description of the circuit elements (transistors, resistors, capacitors, etc.) and their connections, translates this description into nonlinear differential equations to be solved using implicit integration methods, Newton's method and sparse matrix techniques. A general drawback of Spice – and circuit simulators in general – is the complexity of the transfer function it implements to relate physical parameters to performances which generally find not analytical forms but only heavy numerical implementations. This hampers intensive exploration of the performance landscape in search of optimal parameters. In our paper we bypass this handicap using polynomial approximations. We identify these polynomials in terms of granular constructs. Hence we adopt a principled philosophy of considering the region D_x where we expect to set the model parameters as an aggregate of fuzzy sets in various respects [15]. In this way we locally interpolate the Spice function g through a polynomial, hence a mixture of monomials that we associate to the single fuzzy sets. Many studies show this interpolation to be feasible, even in the restricted form of using posynomials, i.e. linear combination of monomials through only positive coefficients [16]. While this constraint is crucial for solving efficiently convex optimization problems such as geometric programming [17], for our purposes it is superfluous. The granular construct we formalize is the following.

Given a Spice function g mapping from x to y (the generic component of the performance vector y), we assume the domain $D_x \subseteq \mathbb{R}^n$ into which x ranges to be the support of c fuzzy sets $\{A_1,\ldots,A_c\}$, each pivoting around a monomial m_k. We consider this monomial to be a local interpolator that fits g well in a surrounding of the A_k centroid. In synthesis, we have $g(x) \simeq \sum_{k=1}^{c} \mu_k(x)\mathsf{m}_k(x)$, where $\mu_k(x)$ is the membership degree of x to A_k. The degree is in turn computed as a quadratic function of the shift between $g(x)$ and $\mathsf{m}_k(x)$.

On the one hand we have one fuzzy partition of D_x for each component of y. On the other hand, we implement the construct with many simplifications, in order to meet specific goals. Namely:

– since we look for a polynomial interpolation of g, we move from point membership functions to sets, to a monomial membership function to g, so that $g(x) \simeq \sum_{k=1}^{c} \mu_k \mathsf{m}_k(x)$. In turn, μ_k is a *sui generis* membership degree, since it may assume also negative values;

– since for interpolation purposes we do not need $\mu_k(x)$, we identify the centroids directly with a hard clustering method based on the same quadratic shift.

Here below, we detail these points as follows. We exploit two nice bonuses of our clustering instance. On the one hand the training set is constructed by us; thus it may be as large as we want, being constrained only by the computational time required to run a Spice model on a huge set of instances. Namely, we focus on a superset of D_x, so large as to be confident of really including the latter, and uniformly draw m points x_rs which we pair with the $y_r = g(x_r)$s computed by Spice. On the other hand, denoting $\mathsf{m}_k(x) = \beta_k \prod_{j=1}^{n} x_j^{\alpha_{kj}}$, if we work with logarithmic scales, the shifts we consider for the single (say the i-th) component of y_r are the distances between $z_r = (\log x_r, \log y_r)$ and the hyperplane $\mathsf{h}_k(z) = w_k \cdot z + b_k = 0$, with $w_k = \{\alpha_{k1}, \ldots, \alpha_{kn}\}$ and $b_k = \log \beta_k$, constituting the centroid of A_k in an adaptive metric. Indeed, both w_k and b_k are learnt by the clustering algorithm aimed at minimizing the sum of the distances of the z_rs from the hyperplanes associated to the clusters they are assigned to. The procedure we implement is the following.

Algorithm 1. Adaptive c-means for Mosfet

For a given Spice function g from $D_x \in \mathbb{R}^n$ to $D_y \in \mathbb{R}^t$

1. Initialization.
   1.1. Draw uniformly m points $x_r \in D_x$ and form the training set $T = \{(x_1, g(x_1)), \ldots, (x_m, g(x_m))\}$.
1. Then, for each component y_i of $g(x)$
   2.1. Set the number c of clusters.
   2.2. Randomly draw a set of c hyperplanes $\mathsf{h}_k(z) = w_k \cdot z + b_k$ each lying in $\mathbb{R}^{n+1}$, with $z = (\log x, \log y_i)$.
   2.3. For each z_r:
      2.3.1. compute $d(z_r, \mathsf{h}_k(z_r)) = \left(\frac{w_k \cdot z_r + b_k}{||w_k||}\right)^2$ for each h_k;
      2.3.2. assign z_r to the cluster A_k affected by the minimum over $\widetilde{k}$ of $d(z_r, \mathsf{h}_{\widetilde{k}}(z_r))$;
      2.3.3. update w_k and b_k along the gradient of $d(z_r, \mathsf{h}_k(z_r))$.

In this way we obtain for instance the clustering of a training set drawn in the domains $D_x = \{Vth_0, Tox\}$ and $D_y = \{Vth, Ron\}$, where Vth_0 denotes the threshold voltage for large range devices, and Tox the gate oxide thickness of a totally dielectric isolated BicMOS whose performance is measured in terms of threshold voltage (Vth) and on-resistance (Ron). We use BSIM3v3.3 as a Spice version 3 software simulator [18] to draw the training set and consider separately 3 clusters in the spaces $\{Vth_0, Tox, Vth\}$ and $\{Vth_0, Tox, Ron\}$ in logarithmic scales, so that the points turn out to be grouped around 3 planes in each space, as shown in Figure 1.

With the clustering procedure we essentially learn the exponents α_{kj} through which the x components intervene in the various monomials, whereas the β_ks remain ancillary parameters. Indeed, to get the polynomial approximation of $g(x)$ we compute the mentioned *sui generis* memberships through a simple quadratic fitting, i.e. by solving w.r.t. the vector $\mu = \{\mu_1, \ldots, \mu_c\}$ the quadratic optimization problem:

$$\mu = \arg\min_{\widetilde{\mu}} \sum_{r=1}^{m} \left(g(x_r) - \sum_{k=1}^{c} \widetilde{\mu}_k \prod_{j=1}^{n} x_{rj}^{\alpha_{kj}} \right)^2 \qquad (1)$$

where x_{rj} denotes the j-th component of the r-th element of the training set T, and μ_ks override β_ks. In Figure 2 we report the polynomials approximating the Spice relations in the above BicMOS scenario, together with the output values we sampled to compute them. We mention that, while in Figure 1 we focused on only three clusters in each picture to better render grouped points and related planes, to obtain a satisfactory fitting we partitioned the sampled points into 15 clusters.

2.2 Identifying the Statistical Parameters of the Model

With the previous section we are left with a consolidated model for locally approximating g, namely

$$y_i = \sum_{k=1}^{c} m_{ik}(x) = \sum_{k=1}^{c} \mu_{ik} \prod_{j=1}^{n} x_j^{\alpha_{ikj}} \quad \text{for } i = 1, \ldots, t \qquad (2)$$

Now we use it to solve the inverse problem:

Which statistical features of X ensure a good coverage of the Y domain spanned by the performances measured on a sample of produced dies?

Hiding specific problems connected to inter-die/intra-die effects [19, 20] we may concretize the above question in terms of α-*tolerance regions*, i.e.

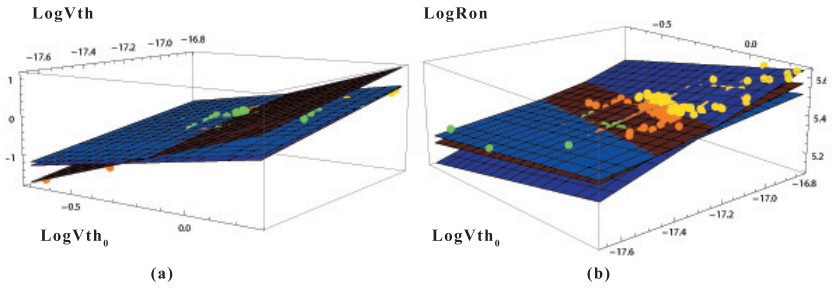

Figure 1. Fuzzy clusters associated to output: (a) Vth, and (b) Ron, in logarithmic scales. Bullet color and radius are a function of cluster membership.

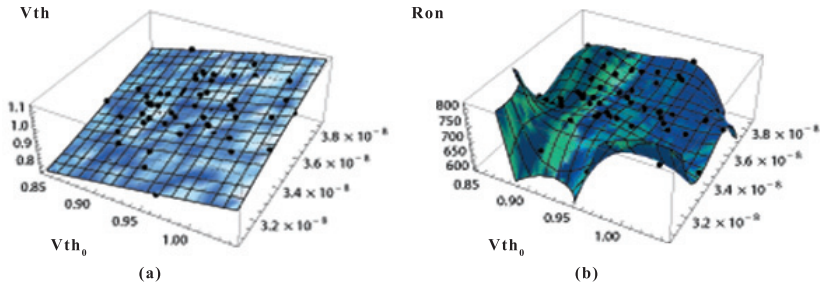

Figure 2. Polynomial approximation, composed of 15 monomials, associated to output: (a) Vth, and (b) Ron. Bullets: output data used to compute polynomials.

regions suitably spread around the mean of the performance vector within which we expect to find the α percent of the performance population. We look for a θ_X defining a Y distribution, via Spice mapping, with a tolerance region containing a good percentage (say $\alpha = 95\%$) of the sampled performances.

A check of the above percentage is at the basis of the third conceptual block we listed at the beginning of this section. We will describe it in Section 4.1. To find a θ_X generating a successful Y population we use a series of partly alternative tools in the province of the second block, which we describe here below. We borrow them from a set of well reputed techniques that are essentially empirical. Thus we will devote a large part of the paper to validating them numerically.

2.2.1 A Suited Interpretation of the Moment Method.

As an early solution of the problem we rely on the first and second moments of the target distribution, which are estimated on the basis of a sample s_Y of sole Y collected from the production lines as representatives of properly functioning circuits. Our goal is to identify the statistical parameters $\widetilde{\theta}_X$ of X that produce through (2) a Y population best approximating the above first and second order moments. The lead strategy is incremental, so that we numerically compute, on the basis of the current instantiation of the candidate solution, parameters and statistics which do not constitute the direct identification target.

In greater detail, X is assumed to be a multidimensional Gaussian variable, so that we identify it completely through the mean vector ν_X and the covariance matrix Σ_X which we do not constrain in principle to be diagonal[21]. The analogous ν_Y and Σ_Y are a function of the former through (2). Although they could not identify the Y distribution in full, we are conventionally satisfied when these functions get numerically close to the estimates of the parameters they compute (directly obtained from s_Y). Denoting with $\nu_{X_j}, \sigma_{X_j}, \sigma_{X_{j,k}}$ and $\rho_{X_{j,k}}$, respectively, the mean and standard deviation of X_j and the covariance/correlation between X_j and X_k, the master equations of our method are the following:

1.
$$\nu_{Y_i} = \sum_{k=1}^{c} \alpha_{ikj} \nu_{M_{ik}} \tag{3}$$
where M_{ik} on the right is a short notation of $m_{ik}(X)$, and $\nu_{M_{ik}}$ denotes its mean.

2. Thanks to the approximations
$$\nu_\Xi \simeq \log \nu_X, \quad \sigma_\Xi \simeq \sigma_X / \nu_X, \quad \rho_{\Xi_{i,j}} \simeq \rho_{X_{i,j}} \tag{4}$$
with $\Xi = \log X$, coming from the Taylor expansion of respectively $\Xi, (\Xi - \nu_\Xi)^2$ and $(\Xi_i - \nu_{\Xi_i})(\Xi_j - \nu_{\Xi_j})$ around (ν_{X_i}, ν_{X_j}) disregarding others than the second order terms, the rewriting of Σ_Y reads

$$\sigma^2_{Y_i} = \sum_{k=1}^{c} \sigma^2_{M_{ik}} + 2 \sum_{\substack{k,r=1 \\ k<r}}^{c} \sigma_{M_{ik,ir}} \tag{5}$$

$$\sigma_{Y_{i,j}} = \sum_{k,r=1}^{c} \sigma_{M_{ik,jr}} \tag{6}$$

with

$$\sigma^2_{M_{ik}} \simeq \nu^2_{M_{ik}} \left(\sum_{j=1}^{n} a^2_{ikj} \frac{\sigma^2_{X_j}}{\nu^2_{X_j}} + 2 \sum_{\substack{j,r=1 \\ j<r}}^{n} \rho_{X_{j,r}} a_{ikj} a_{ikr} \frac{\sigma_{X_j}}{\nu_{X_j}} \frac{\sigma_{X_r}}{\mu_{X_r}} \right) \tag{7}$$

$$\sigma_{M_{ik,ir}} \simeq \nu_{M_{ik}} \nu_{M_{ir}} \sum_{j,w=1}^{n} a_{ikj} a_{irw} \rho_{X_{j,w}} \frac{\sigma_{X_j}}{\nu_{X_j}} \frac{\sigma_{X_w}}{\nu_{X_w}} \tag{8}$$

We numerically solve (3) and (5-6) in ν_X and Σ_X when the left members coincide with the target values of ν_Y and Σ_Y, respectively, and $\nu_{M_{ik}}$ is approximated with its sample estimate computed on samples artificially generated with the current values of the parameters. Solving equations means minimizing the differences between left and right members, so that the crucial point is the optimization method employed. We implemented various algorithms framed into two alternative search philosophies: either stressing a single parameter at a time of the input distribution, or exploring all directions jointly. At a higher level, the domain of the minimization algorithm may be either the entire search space, hence having each component of ν_X and Σ_X as coordinates, or cyclical subspaces singularly related to mean, variances and correlations of the questioned input distribution.

The three building blocks are the following.

2.2.2 The Steepest Descent Strategy.

Using the Taylor series expansion limited to second order[22], we obtain an approximate expression of the gradient components of ν_Y w.r.t. ν_X through

$$\frac{\partial \nu_{Y_i}}{\partial \nu_{X_j}} \simeq \sum_{k=1}^{c} \alpha_{ikj} \left(\frac{1}{\nu_{X_j}} + \frac{\sigma^2_{X_j}}{\nu^3_{X_j}} \right) \nu_{M_{ik}} \tag{9}$$

Thus we may easily look for the incremental descent on the quadratic error surface accounting for the difference between computed and observed means. Expression (9) confirms the scarce sensitivity of the unbiased mean ν_X, and its gradient as well, to the second moments, so that we may expect to obtain an early approximation of the mean vector to be subsequently refined even for values of σ_X possibly far from the correct ones. Hence the pseudocode of this tool is reported in the following Algorithm 2.

Algorithm 2. Gradient descent for ν_X

For a given Spice function g from $D_x \in \mathbb{R}^n$ to $D_y \in \mathbb{R}^t$ and a sample s_Y defined on D_y, sounding initial values for X parameters, and a polynomial fitting of g

1. **for** a suitable number of *epochs*

   1.1. **for** a suitable number of *cycles*

      1.1.1. Generate a (X,Y) sample (s'_X, s'_Y) with current parameters

      1.1.2. Compute the Y sample mean ν'_y and all X sample statistics needed to compute (9)

      1.1.3. Accumulate $\Delta_{\nu_{y_{ij}}} = (\nu_{y_i} - \nu'_{y_i}) \sum_{k=1}^{c} \alpha_{ikj} \left(\frac{1}{\nu_{X_j}} + \frac{\sigma^2_{X_j}}{\nu^3_{X_j}} \right) \nu_{M_{ik}}$, $j = 1, \ldots, n$, for each Y component

      1.1.4. Update $\nu_{X_j} = \nu_{X_j} - \eta_{ij} \Delta_{\nu_{y_{ij}}}$

   1.2. Update η_{ij} according to a suitable learning rule

2. Fit g in the neighborhood of the updated ν_X

The updating of η, or analogous smoothing expedients such as momentum terms[23], are a critical point whose management highly affects the efficiency of the algorithm. As will be shown in a

moment, the routine may be variously iterated until a satisfactory result is obtained.

While analogous to the previous task, the identification of X variances and correlations owns one additional benefit and one additional drawback. The former derives from the fact that we may start with a, possibly well accurate, estimate of the means. The latter descends from the high interrelations among the target parameters which render the exploration of the quadratic error landscape troublesome and very lengthy.

2.2.3 Identification of Second Order Moments.

An alternative strategy for X second moments identification is represented by the evolutionary computation. Given the mentioned computational length of the gradient descent procedures, algorithms of this family becomes competitive on our target. Namely, we used Differential Evolution [24], with specific bounds on the correlation values to avoid degenerate solutions. The routine pseudocode is reported in the following Algorithm 3.

Algorithm 3. Inference of second order moments

For a given Spice function g from $D_x \in \mathbb{R}^n$ to $D_y \in \mathbb{R}^t$ and a sample s_Y defined on D_y, a polynomial fitting of g, and sounding ν_X,

1. Generate a (X,Y) sample with current parameters
2. Compute all X and Y sample statistics necessary to implement (5) to (8)
3. Minimize the square error on the second order moments through general-purpose optimization algorithms
4. Update Σ_X with the best solution found at the previous step
5. Fit g around ν_X with the updated Σ_X

2.2.4 A Brute Force Numerical Variant.

Gradient descent is a very rough local optimization procedure. However, rather than embarking on more sophisticated procedures such as Scaled Conjugate Gradient (SCG)[25] or the Levenberg-Marquardt algorithm (LM) [26], we preferred to move to a still more rudimentary strategy to get rid of the loose approximations introduced in (3) to (9). Thus we: i) avoid computing approximate analytical derivatives, by substituting them with direct numerical computations [27], and ii) adopt the strategy of exploring one component at a time of the questioned parameter vector, rather than a combination of them all, until the error descent stops. This results in a very streamlined version of the Variable Step Search (VSS) algorithm [28] used to optimize complex functions such as neural networks error terms. Spanning numerically one direction at a time allows us to ask the software to directly identify the minimum along this direction. The further benefit of this task is that the function we want to minimize is analytic, so that the search for the minimum along one single direction is a very easy task for typical optimizers, such as the naive Nelder-Mead simplex method [29] implemented in Mathematica [30]. For analogous reasons we abandon the evolutionary algorithm in favor of this strategy even for the second order moment identification. A possible drawback is that the exploration of the search space one component at a time may lead to local minima that are far from the global one. This is a common drawback of local descent strategies. The same holds for global techniques as well which, as for evolutionary algorithms, do not explore the entire search space. However, no one may predict what is the overall overburden, for instance, if we follow either the negative gradient or simply one of its components. Rather, early experimental results seem to promote the hypothesis that the convergence rate toward global solutions improves by employing the latter strategy when we remain in the field of granular computing [28]. Moreover, the computational complexity of following a single component is definitely lower than the one of either descending along the gradient or evolving a population.

We structured the method in a cyclic way, plus stopping criterion based on the amount of overall parameter variation. Each cycle is composed of: i) an iterative algorithm which circularly visits each component direction minimizing the identification error, until no improvement may be achieved over a given threshold, and ii) a fitting polynomial refresh on the basis of a Spice sample in the neighborhood of the current parameter vector. We complete the routine with a last assessment of the parameters that we pursue by running jointly on them all a local descent method such as Quasi-Newton procedure in

one of its many variants [31]. As a conclusion, we may optionally substitute either Algorithm 2 or Algorithm 3, or even both, with the following Algorithm 4.

Algorithm 4. Greedy descent for ν_X (Σ_X)

For a given Spice function g from $D_x \in \mathbb{R}^n$ to $D_y \in \mathbb{R}^t$ and a sample s_Y defined on D_y, sounding initial values for X parameters, and a polynomial fitting of g

1 **for** a suitable number of *cycles*

1.1. Generate a (X,Y) sample (s'_X, s'_Y) with current parameters

1.2. Compute the Y sample mean ν'_Y (sample covariance matrix Σ'_Y)

1.3. Compute the minimum m_j of $(\nu_Y - \nu'_Y)^2$ along any ν_{X_j} direction, for $j = 1,\ldots,n$ (the minimum $m_{i,j}$ of $(\Sigma_Y - \Sigma'_Y)^2$ along any $\sigma_{i,j}$ direction, for $i,j = 1,\ldots,n$)

1.4. Select the direction j^* (the directions (i^*, j^*)) whose correspondent m_j ($m_{i,j}$) achieves the lowest minimum

1.5. Update $\nu_{X_{j^*}}$ (σ_{i^*,j^*}) with the optimal coordinate computed in step 1.3

2 Run a local optimization algorithm to reach a local optimum

3 Fit g in the neighborhood of the updated ν_X (Σ_X)

2.2.5 Fine Tuning Via Reverse Mapping.

Once a good fitting has been realized in the questioned part of the Spice surface, we may solve the identification problem in a more direct way by first inverting the polynomial mapping to obtain the X sample at the root of the observed Y sample, and then estimating θ_X directly from the sample defined in the D_X domain. The inversion is almost immediate if it is univocal, i.e., apart from controllable pathologies, when X and Y have the same number of components. Otherwise the problem is either overconstrained (X components in a number less than Y components) or underconstrained (opposite relation between component numbers). The first case is avoided by simply discarding exceeding Y components, possibly retaining the ones that improve the final accuracy and avoid numeric instability. The latter calls for a reduction in the number of questioned X components. Since X follows a multivariate Gaussian distribution law, by assumption, we may substitute some components with their conditional values, given the others. In greater detail, if X is partitioned in (X_1, X_2) with dimensions t and $n-t$ respectively, from ν and Σ splitted as follows

$$\nu = (\nu_1, \nu_2) \tag{10}$$

with dimensions t and n-t respectively

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \tag{11}$$

with dimensions $t \times t, t \times (n-t), (n-t) \times t$, and $(n-t) \times (n-t)$, respectively, then the distribution of X_1 conditional on $X_2 = x_2$ again follows a multivariate Gaussian with parameters $(\bar{\nu}, \overline{\Sigma})$, where

$$\bar{\nu} = \nu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \nu_2) \tag{12}$$

$$\overline{\Sigma} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \tag{13}$$

the latter representing the Schur complement of Σ_{22} in Σ[32]. The further conditioning of X_1 by Y (since we are reversing the latter on the former) still reduces $\overline{\Sigma}$ by a factor ξ that we may guess from the comparative scatter of the Y' population reconstructed with the inferred θ_X – for instance homogeneously set to $\frac{1}{3}$ for each component in the experiments described in Section 4.

Hence the pseudocode of this part is described in the following Algorithm 5.

Algorithm 5. Reverse mapping

For a given Spice function g from $D_x \in \mathbb{R}^n$ to $D_y \in \mathbb{R}^t$ and a sample s_Y defined on D_y, sounding X parameters, and a polynomial fitting ϕ of g,

1 **if** $n \neq t$ **then**

1.1 **if** $t > n$ **then** hide $t-n$ components of Y (possibly according to a suitable optimality criterion)

1.2 **if** $n > t$ **then** compute conditional mean and variance of a subset X_t of t components of X given its complement $X_{n-t} = \widetilde{\nu}_{n-t}$, through (12-13)

2 **for** each $y_k \in s_Y$

2.1. Get $x_k = \phi^{-1}(y_k)$ obtaining a companion sample s_X

3 Compute ν_X, Σ_X on s_X

4 Fit g in the neighborhood of the current ν_X, Σ_X

3 Profiling the Identification Procedure

The tools we described are in principle non autonomous. Rather we may variously combine them to get results at different accuracy levels. In Algorithm 6 we propose a procedure to realize these computational paths. On the one hand the default sequencing generally leads to a very satisfactory identification of X parameters. However, since all steps are local, possibly relying on approximated optimizers, the procedures allow for both backtracking at any step, with possible selection of alternative optimization tools, and recycling of the entire loop in search of better identification of the current value of the global variables $\{\nu_X, \Sigma_X\}$, whenever necessary.

Algorithm 6. Polynomial identification

For a given Spice function g from $D_x \in \mathbb{R}^n$ to $D_y \in \mathbb{R}^t$, and a sample s_Y defined on D_y.

1 Initialization.

1.1. Set sounding X parameters $\{\nu_X, \Sigma_X\}$

1.2. Fit g polynomially in a $D_x \times D_y$ superset of s_Y through Algorithm 1

1.3 **if** $n = t$ **then goto** [4.1.]

1.4 **if** $n < t$ **then** select n components of Y and **goto** [4.1.]

2 Mean identification.

2.1. According to your selection

2.1.1a. Identify ν_X through Algorithm 4 – default selection

2.1.1b. Identify ν_X through Algorithm 2 – alternative selection

2.2. **if** the identification is not satisfactory **then goto** either [1.1.] or [2.1.]

3 Variance and correlation identification.

3.1. According to your selection

3.1.1a. Identify X variance and covariance matrix Σ_X through Algorithm 4 – default selection

3.1.1b. Identify X variance and covariance matrix Σ_X through Algorithm 3 – alternative selection

3.2. **if** the identification is satisfactory **then stop**, **else goto** either [1.1.] or [2.1.]

4 Fine tuning

4.1 **if** $n \leq t$ **then** inverse map s_Y through Algorithm 5

4.2 **if** $n > t$ **then while** the identification is not satisfactory

4.2.1. Select t components of X, and compute their conditional expectations,

4.2.2. inverse map s_Y through Algorithm 5

4.2.3. compute $\{\nu_X, \Sigma_X\}$

4.3 **if** the identification is not satisfactory **then goto** either [1.1.] or [2.1.] or [3.1.] or [4.1.]

5 **stop**

The two pitfalls of this procedure are the polynomial fitting efficacy and the locality of the optimization procedures.

1. In order to constrain the procedure computational complexity, we stress the use of fitting polynomials in place of Spice routines. Fitting polynomials are the root of our inference. However, their accuracy is local. As mentioned in the introduction, this is a drawback that reverts into a benefit provided that we employ them in a tight surrounding of the parameter values around which the fitting coefficients have been computed. Hence we need to frequently regenerate the polynomials during the inference process. For sure, at the completion of any optimization step, we should use Spice routines to generate a Y sample starting from a sample s_x drawn from a multivariate Gaussian X with the inferred parameters, in order to check the true closeness of the polynomially approximated Y population to the really observed sample. Then we use the joint (X,Y) generated sample to refresh the fitting. We might need similar operations also in intermediate steps of the inference procedure, namely at each backtracking step, to

2*device	model parameter		performance parameter	
	label	description	label	description
pMOS	U_0	Mobility at nominal temperature		
	A_0	Bulk charge effect coefficient	GM	conductance
	VTH_0	Threshold voltage at $V_{BS} = 0$ for large L	ID_{SAT}	source drain current
	K_1	First order body effect coefficient	VTH_{25-25}	saturation voltage
	B_{01}	Bulk charge effect coefficient for channel lenght	VTH_{25-08}	saturation voltage
	B_{11}	Bulk charge effect coefficient for canne width		
nMOS	U_0	Mobility at nominal temperature	GM	conductance
	V_{SAT}	Saturation voltage	ID_{SAT}	source drain current
	VTH_0	Threshold voltage at $V_{BS} = 0$ for large L	VTH_{25-25}	saturation voltage
	K_1	First order body effect coefficient	VTH_{25-08}	saturation voltage
nMOS-DIB12	Bf	Ideal maximum foward Beta	HFE	Current Gain
	Re	Emitter Resistance	VA	Early Voltage
	Is	Transport Saturation Current	I_c	Collector Current
	Vaf	Forward Early Voltage		

Table 1. Model parameters and performances of the identification problems.

avoid injecting biases in the parameter estimation. Moreover, each identification satisfaction check is based on a Y population generated using the current identified parameters and the true Spice mapping; a comparison of this population with the original sample s_Y will decree the satisfaction degree.

2. Local optimization procedures suffer from the general drawback of getting stuck in local minima. As for mean estimation, we found this drawback definitely lighter with Algorithm 4 than with the alternative Algorithms 2 and 3. However, Algorithm 2 is more robust in general, so that we gave the user the possibility of using it in the rare situations where the former fails. As for the covariance matrix, in devising Algorithm 3 we found use of Differential Evolution method suitable. However, to avoid degenerate solutions, we prohibit the correlations to take a value greater than a suitable threshold (actually it is recommended to avoid values higher than 0.6). We remark that the pseudocodes of Algorithms 2-4 are specialized on suitable subspaces of the search space. Nevertheless, in the most difficult cases we found extremely efficient to jointly explore the entire search space, specially through the Algorithm 4. Fine tuning of the above steps is generally obtained through Algorithm 5 that we use substantially in a combinatorial scheme. Having no clear idea of what is the subset of conditional components of X when $n > t$ (or skipped components of Y when $n < t$) which leads to the best estimates, we simply check for all subsets. In the case this scheme proves unbearable because of large differences between n and t, we could either provide a first guess on a subset of the whole sample, or at worst use some rules of thumb to bind the number of subsets.

3. Analogously, we have no assurance about the convergence of the whole optimization loop. Therefore the procedure is equipped with recovering routines to store all current solutions, so that at each time we may select the ones that prove more convincing in order to either stop or start a new iteration. No evidence exists, in particular, that Algorithm 5 always represents an improvement of the identification obtained with the sole statistical tools.

We cannot provide in principle an exhaustive computational complexity analysis of the procedure. Rather, we realize that the single cycles in each procedure are at most quadratic in the number of input variables, with the sole exception of the line optimization routine, whose complexity is well controlled, however, by the optimizer software. Hence, the parameters that mostly affect (linearly with the moderate exception of a few cases) the computational load are: i) the size of the samples used in the various tasks, ii) the number of iterations through which we develop the incremental methods, and iii) the combinatorial scheme in the case that input and output space have different dimensions. We will realize the influence of these quantities in the next section.

4 Numerical Results

The procedure we propose derives from a skillful implementation of granular computing ideas

[15], however without theoretical proof of efficiency. While no worse from this perspective than the general literature in the field per se [11], it needs numerical proof of suitability. To this aim we basically work with three real world benchmarks collected by manufacturers and variants devised to stress the peculiarities of the methods. In Table 1 we describe the benchmarks and the identification solutions that we found.

In Figure 3 we synthesize their operational effects.

Namely, the benchmarks refer to:

1. A unipolar pMOS device realized in Hcmos4TZ technology.

2. A unipolar nMOS device differentiating from the former for the sign (negative here, positive there) of the charge of the majority mobile charge carriers. Spice model and technology are the same, and performance parameters as well. However, the domain spanned by the model parameters is quite different, as will be discussed shortly.

3. A unipolar nMos circuit realized in DIB12 technology. DIB technology achieves the full dielectric isolation of devices using SOI substrates by the integration of the dielectric trench that comes into contact with the buried oxide layer.

We have different kinds of samples for the various benchmarks as for both the sample size which ranges from $14,000$ (pMOS and nMOS) to 300 (nMOS-DIB12) and the measures they report: joint measures of 4 performance parameters in the former two cases, partially independent measures of 3 performance parameters in the latter, where only HFE and VA are jointly measured. Taking into account the model parameters, and recalling the meaning of t and n in terms of number of performance and model parameters, respectively, both the sensitivity of the former parameters to the latter and the different difficulties of the identification tasks lead us to face in principle one balanced problem with $n = t = 4$ (nMOS), and two unbalanced ones with $n = 6$ and $t = 4$ (pMOS) and $n = 4$ and $t = 3$ (nMOS-DIB12). In addition, only 4 of the 6 second order moments are observed with the third benchmark. In particular, the model parameters and the performances considered for the three devices are listed in Table 1.

With reference to Table 2, in column $\widetilde{\theta}_X$ we report the parameters of the input multivariate Gaussian distribution we identify in the aim of reproducing the θ_Y of the Y population observed through s_Y. Of the latter parameter, in the subsequent column we compare the values $\widetilde{\theta}_Y$ computed on the basis of $\widetilde{\theta}_X$ (referring to a reconstructed distribution – in italics) with those $\hat{\theta}_Y$ computed through the maximum likelihood estimate from s_Y (referring to the original distribution – in bold). As a further accuracy indicator, we will consider tolerance regions in the performance space $\mathfrak{Y}$, i.e. domains that are placed around the performances centroid and contain a given percentage $1-\delta$ of the performance population. The procedure to build up these regions on the basis of the identified parameter $\widetilde{\theta}_X$ will be discussed in detail in Section 4.1.. In the last column of Table 2, headed by $(1-\widetilde{\delta})/(1-\delta)$, we appreciate the difference between planned tolerance rate (in bold), as a function of the identified Y distribution, and ratio of sampled measures found in these regions (in italics). We consider single values in the table cells since the results are substantially insensitive to the random components affecting the procedure, such as algorithm initialization. Rather, especially with *difficult* benchmarks, they may depend on the user options during the run of the algorithm. Thus, what we report are the best results we obtain, reckoning the overall trial time in the computational complexity consideration we will do in Section 4.2.

For a graphical counterpart, in Figure 3 we report the scatterplot of the original Y sample and an analogous one generated through the reconstructed distribution, both projected on the plane identified by the two principal components[33] of the original distribution. We also draw the intercept of this plane with a tolerance region containing 90% of the reconstructed points (hence $\delta = 0.1$).

An overview of these data looks very satisfactory, registering a relative shift between sample and identified parameters that is always less than 0.17% as for the mean values, 45% for the standard deviations and 25% for the correlation. The analogous shift between planned and actual percentages of points inside the tolerance region is always less than 2%. We distinguish between *difficult* and *easy* benchmarks, where the pMOS sample falls in

benchmark			solution						
dataset	(n,t)	m	$\bar{\theta}_X$			$\bar{\theta}_Y / \hat{\theta}_Y$			$1-\bar{\delta}$ / $1-\delta$
benchmark			μ_X	σ_X	ρ_X	μ_Y	σ_Y	ρ_Y	
pMOS	(6, 4)	14, 000	$\begin{pmatrix} 233.424 \\ 0.28798 \\ 0.99185 \\ 0.45255 \\ 4.06626 \times 10^{-5} \\ 4.67824 \times 10^{-5} \end{pmatrix}$	$\begin{pmatrix} 3.63673 \\ 0.01806 \\ 0.01083 \\ 0.03275 \\ 4.48106 \times 10^{-6} \\ 9.90006 \times 10^{-6} \end{pmatrix}$	$\begin{pmatrix} -0.16582 \\ -0.46312 \\ -0.41451 \\ -0.49665 \\ -0.35008 \\ -0.12573 \\ -0.47067 \\ -0.07056 \\ -0.39330 \\ 0.09484 \\ -0.16367 \\ 0.21068 \\ 0.49711 \\ 0.22781 \\ 0.48312 \end{pmatrix}$	$\begin{pmatrix} \textit{-0.835824} \\ \mathbf{-0.838496} \\ \textit{-0.971835} \\ \mathbf{-0.969196} \\ \textit{0.000973318} \\ \mathbf{0.00097472} \\ \textit{0.00448103} \\ \mathbf{0.00447346} \end{pmatrix}$	$\begin{pmatrix} \textit{0.0118109} \\ \mathbf{0.0187507} \\ \textit{0.0121665} \\ \mathbf{0.0164674} \\ \textit{0.000029378} \\ \mathbf{0.00029348} \\ \textit{0.000146626} \\ \mathbf{0.00130486} \end{pmatrix}$	$\begin{pmatrix} \textit{0.933746} \\ \mathbf{0.451486} \\ \textit{-0.287658} \\ \mathbf{-0.282512} \\ \textit{-0.389979} \\ \mathbf{-0.387441} \\ \textit{-0.254446} \\ \mathbf{-0.0727698} \\ \textit{-0.367477} \\ \mathbf{-0.174543} \\ \textit{0.900391} \\ \mathbf{0.983658} \end{pmatrix}$	$\begin{matrix} \textit{0.946713} \\ \mathbf{0.9} \\ \textit{0.900398} \\ \mathbf{0.8} \end{matrix}$
nMOS	(4, 4)	14, 000	$\begin{pmatrix} 752.395 \\ 152858.0 \\ 0.68184 \\ 0.521661 \end{pmatrix}$	$\begin{pmatrix} 134.099 \\ 9667.22 \\ 0.0186854 \\ 0.131933 \end{pmatrix}$	$\begin{pmatrix} -0.765278 \\ -0.467972 \\ 0.756786 \\ 0.306389 \\ -0.786377 \\ -0.468842 \end{pmatrix}$	$\begin{pmatrix} \textit{0.552391} \\ \mathbf{0.550715} \\ \textit{0.66383} \\ \mathbf{0.664162} \\ \textit{0.00221691} \\ \mathbf{0.00222077} \\ \textit{0.0100527} \\ \mathbf{0.0100711} \end{pmatrix}$	$\begin{pmatrix} \textit{0.028568} \\ \mathbf{0.0276768} \\ \textit{0.0176982} \\ \mathbf{0.0173677} \\ \textit{0.000830626} \\ \mathbf{0.0006619134} \\ \textit{0.0003551129} \\ \mathbf{0.0002803731} \end{pmatrix}$	$\begin{pmatrix} \textit{0.445093} \\ \mathbf{0.395429} \\ \textit{-0.499279} \\ \mathbf{-0.432434} \\ \textit{-0.637969} \\ \mathbf{-0.640323} \\ \textit{-0.298401} \\ \mathbf{-0.271952} \\ \textit{-0.375841} \\ \mathbf{-0.354887} \\ \textit{0.92015} \\ \mathbf{0.950419} \end{pmatrix}$	$\begin{matrix} \textit{0.9008} \\ \mathbf{0.9} \\ \textit{0.8304} \\ \mathbf{0.8} \end{matrix}$
nMOS-DIB12	(4, 3)	322	$\begin{pmatrix} 138.302 \\ 0.67258 \\ 5.28102 \times 10^{-18} \\ 136.319 \end{pmatrix}$	$\begin{pmatrix} 8.3859 \\ 0.263238 \\ 4.14306 \times 10^{-19} \\ 13.6538 \end{pmatrix}$	$\begin{pmatrix} -0.192107 \\ 0.00139749 \\ -0.477207 \\ -0.980327 \\ 0.167527 \\ -0.0444712 \end{pmatrix}$	$\begin{pmatrix} \textit{113.244} \\ \mathbf{113.242} \\ \textit{0.000065246} \\ \mathbf{0.000065327} \\ \textit{110.164} \\ \mathbf{110.238} \end{pmatrix}$	$\begin{pmatrix} \textit{6.82099} \\ \mathbf{6.95918} \\ \textit{4.96031} \times 10^{-6} \\ \mathbf{4.81021 \times 10^{-6}} \\ \textit{11.1459} \\ \mathbf{11.2166} \end{pmatrix}$	$\begin{pmatrix} \textit{-0.490798} \\ \mathbf{-0.566678} \end{pmatrix}$	$\begin{matrix} \textit{0.9054} \\ \mathbf{0.9} \\ \textit{0.8136} \\ \mathbf{0.8} \end{matrix}$

Table 2. Benchmarks used for testing the proposed procedure and analysis of the identification solution. Rows: benchmarks. Columns: inferred model distribution parameters (indexed by X) and reconstructed performance parameters (indexed by Y),plus comparative levels of the tolerance regions (as a function of δ).

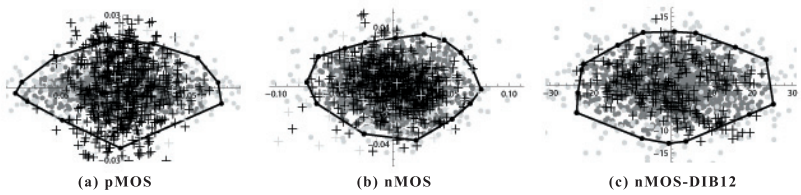

Figure 3. Comparison between output data and reconstruction provided by our procedure for the devices listed in Table 1 when projected on the two principal components of the target. Points: reconstructed population lying within (dark gray) and outside (light gray) 0.90 tolerance region (black curves) identified by black points. Gray crosses: original target output; black crosses: target output addicted with noise terms uniformly spread in the discretization range.

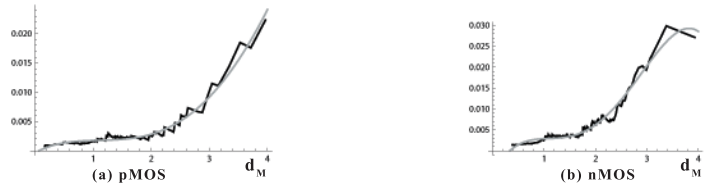

Figure 4. Accuracy of the polynomial approximation versus distance of target points from their mean computed on: (a) pMOS, and (b) nMOS, through the Mahalanobis metric d_M, as for abscissa, and relative error ε, as for ordinate (black curves), together with their polynomial fitting of degree 4 (gray curves).

the first category. Indeed the same percentages referring to the remaining benchmarks decreases to $0.13\%, 10\%$ and 9%. In the rest of this section we will consider these performances both in relation to the peculiarities of our benchmarks and procedures, and in contrast with analogous results in the literature.

4.1 Deepening the Methods

We learn the following lessons from a deeper analysis of the above experiments.

Polynomial fitting. It is a local procedure which generates curves that are accurate in a region centered in the sampled points. This means that, on the one hand, we need to refresh the fitting each time the mean of the sample changes notably. On the other hand, the loss of accuracy with the distance from the center is not dramatic, since it concerns regions of low probabilities. Figure 4 plots a typical trend of the accuracy with the distance from the mean, when the former is measured according to the relative error and the latter is computed in terms of the Mahalanobis metric. The pictures also report a polynomial fitting of the curves, showing in any case the adequacy of the polynomial approximation, provided that a suitable neighborhood of the target region emerged in the learning stage.

Deterministic versus statistic tools. In principle, in the case of balanced samples ($n = t$), the fitting quality is the sole feature responsible for the statistical identification. We start with *sounding* X parameters and draw a first fit. With this fit we revert y_rs on x_rs and estimate the X parameters directly from the latter sample. However, the Y sample generated with these X parameters may not prove satisfactory. Then, we may start a cycle where we use the latter for generating another (X,Y) sample that we fit with a new polynomial, and use the latter for a new reverse of the original s_Y, until we are satisfied. With $t > n$ we obtain the same back-fitting after discarding some Y components. With $n \neq t$ we need to reformulate the problem as discussed in Section 2.2. In particular, in Figure 5(a) we report the analogous result of Figure 3(b) when we remove the third performance parameter in order to artificially unbalance the identification problem, getting in any case comparable results

Figure 5. Robustness of the solution vs. information lack and methods. Same graphs as in Figure 3 when the third perfomance parameter is unavailable and the moment error minimization is pursued thorugh: (a) Variable Step Search, and (b) Steepset Gradient Descent coupled with Differential Evolution methods.

on the remaining components (mean relative error between current and target mean vectors $\approx$ 0.3%, on second order moments $\approx$ 3%) denoting a balance between the decrease in the target difficulty and in the number of available statistics.

While in *easy* cases statistical tools play the mentioned ancillary role w.r.t. the deterministic inversion procedure, on the first benchmark (pMOS) they constitute the sole way of identifying the Y parameters. This is because any attempt to invert s_Y introduces a deterioration of the current parameter identification. Thus we rely on the sole greedy descent routines to get $\widetilde{\theta}_Y$ in Table 1. Furthermore, this accounts for the difficulty of the pMOS benchmark. The related Spice functions are actually so difficult that the average smoothing introduced by the moment estimators results into a necessary benefit to the identification task. From the second to last column in Table 1 we argue that what is most responsible for the relatively poor identification is the correlation between the first and second component of the performance vector. In turn, we note that both components are measured in a coarse discrete scale, while the remaining two are measured in a continuous scale (actually a so fine discrete scale to be confused with a continuous one). This happens also in the second benchmark (nMOS) but without the same negative effects. Thus we guess that the loss of accuracy in the performance measurements hampers the identification accuracy only within a certain threshold.

As a technical remark, the logarithmic scale transform requested by the clustering Algorithm 1 is not applicable on negative values of either model or performance parameters, obviously. This does not constitute a drawback, however, but just requires a wise management of their absolute values.

Descent method. Of the methods to reduce the error of the means identification, the most efficient and accurate is the Variable Step Search descent along single components. The method, originally structured in a cyclic way, find a suitable stopping criterion by measuring the amount of parameter variation. Namely, we circularly visit each components by finding a local minimum in the chosen direction (operation which can be performed analytically in a very efficient way by general-purpose optimizers [30]), and adopt a threshold 10^{-5} on the relative change on each direction to abandon the search. Note that, although no theoretical results guarantee a fast convergence [28], the computation generally ends in two cycles. Thanks to the general speed of above method, we may devote some extra computational time, in any case not influencing the order of speed factor of the method, to perform a descent on the overall error surface for the final assessment of the solution.

We maintain the Steepest Gradient Descent and the Differential Evolution methods because they represent alternative procedures which are robust (hence to be employed in tricky instances) but less accurate and more computational demanding. In Figure 5(b) we represent the same picture as in Figure 5(a) when using the latter two alternative identification tools. The accuracy are almost the same (relative error ≈ 0.005 and 0.05 respectively for mean and second order elements), but the running time is approximately one order greater than the former method's.

The proper sizing of the tools. We have three sizing parameters in the numerical experiments:

the number of model parameters, the number of fitting monomials, and the size of the generated samples.

The Spice model has around 100 input variables (the model parameters) that diminish to 12-13 when we consider the ones that are either tangibly varying during the production and tangibly influencing the circuit performances with their variation. However 12 is still a high number, close to be prohibitive, for the incremental tools implemented in our procedure. Of course it is a moderate size of the input of a neural network, such as a multilayer perceptron, which processes available data incrementally as well. But with the latter device we are facing very regular atomic functions, such as sigmoids[34], possibly in a huge number so that their composition may reproduce complex goal functions [35]. In our case, we work with simple but less regular atomic functions, the monomials, in a number that is limited in order to avoid overfitting, yet with the same goal of reproducing complex functions, and with the additional target of using the fitting to identify the input population. As a matter of fact, practitioners look for 3 to 4 model parameters with benchmarks similar to ours, typically those proving most sensitive to the performances with standard sensitivity tools (for instance those based on the Jacobi matrix[36]). With the pMOS benchmark (the *difficult* instance) we profit from two further components to improve our results by about 20% as for the second order moments, getting exactly the identification we described in Table 1. Note that the additional model parameters revert into additional noise in the procedure, which is the reason why we avoid incrementing the parameters in this benchmark further on.

As for the number of monomials, we elude the mentioned overfitting risk with a trivial trial-and-error procedure based on *elbow* heurstics [37]. Namely, at the beginning of the whole procedure we simply launch a series of fitting routines on the training set, each with a different number of monomials, and check when the slope of the fitting quadratic error computed on a companion test set is such that adding another monomial does not reduce the error tangibly (see Figure 6). Since we are fitting different performances on the same training set, we use obvious artifices to get a same number of monomials on each performance, for simplifying the implementation of the subsequent identification steps.

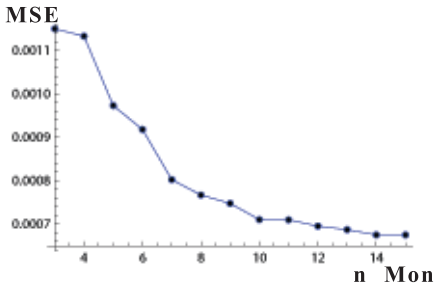

Figure 6. Course of the mean squared interpolation error vs. number of monomials.

Namely, we focus on the maximum number of monomials found by the clustering algorithm on a given output component and increase the number of the other monomials by introducing additional ones with slightly perturbed copies of the former's coefficients. Avoiding both overfitting and an exceeding number of interpolating parameters at the basis of the subsequent inference task is a common problem that has been faced by many researchers [38] and we solve with a local approach. Our additional expedient, mentioned earlier, lies in the fact that, rather than looking for a single function interpolating Spice in the whole model parameter domain, we couple the interpolating task with θ_X statistical identification, so as to concentrate the interpolating requests exactly in a specific neighborhood of the candidate X mean.

As for the training sets, we structured the procedure so as to commensurate the computational effort with the accuracy requirement. We remark that, besides the sample observed on the performance parameters directly from the production plant, we operate in an active learning mode [39] producing specific samples by ourselves. Namely, from the current X population (i.e. the model parameter population as it is identified at the current iteration) we draw two kinds of samples, respectively to: i) identify the interpolating polynomial, and ii) compute run-time statistics. The first sample is more costly to generate, since it requires the computation of Spice functions.

Hence we tend to work with sample sizes of the order of $m = 200$. In Figure 7 we show different degrees of overlap between the empirical cumulative distribution functions (ECDF) of the original performances and companion ECDFs of the reconstructed performances with different sample sizes. Though we have an evident improvement with the sample size, we assume 250 as a number of examples that is adequate to limit the other drifting factors that, as we discussed, hampers the accuracy of the results. The second kind of samples is needed to compute the X moment expected values in (3) to (9). Since their generation is very quick, we do not refrain from using a sample of size $20m$ to get stable estimates of these parameters.

Drawing a tolerance domain. The operational goal of the questioned procedure is to check that a good percentage of the produced circuits will satisfy the requests on performances as they emerge from the current production. Specifically, on the one hand, from a sample s_Y of this production

consisting of correctly working circuits, we have a statistical picture of the tolerance region that we may expect for the circuit. Thus, on the other hand, we may require the Y tolerance region to contain a percentage of, say, 95% of the produced population, and check this property through the percentage of s_Y elements falling in the same region. No matter how we characterize the Y population, we may compute the tolerance region through a contour line of Y density function which includes the population with the mentioned probability. A preliminary exercise is to start from the X parameters we have identified and generate a huge Y population through Spice – a heavy task that we face only once, however. Then we compute the desired contour line with a standard Convex Hull Peeling Depth approach[40–42] and check the s_Y percentage falling inside it. This is exactly what we did in Figure 3. The values in the last column of Table 1 denote that we may use the models we identified in both nMOS and nMOS-DIB12 to compute the production yield with good accuracy. Thus we expect to get reliable data about yield also in case of small changes in the model parameters. With pMOS we have greater shifts between planned and reckoned tolerance rates, as a drawback of the inner model complexity of this circuit. We must also notice that the latter rates are reckoned solely on the third and fourth performance components, since a large part of the first and second components coordinates fall outside the tolerance region, as a consequence of the mentioned overestimation of their correlation. We may view the same as a further consequence of the coordinate-discretization/model-complexity pair, the sole benefit of which is the fact the tolerance region we draw represents a conservative solution denoting a computed yield that is definitely poorer than the actual one. Actually, with pMOS we pay for the low number of Y statistics – possibly not the more relevant ones – we involved in the model identification despite the model complexity. As a further remark, we note that with the *easy* benchmarks, the tolerance regions computed on the distribution reconstructed through the X sample plus polynomial transforms is very close to the one reconstructed through the same sample via Spice mapping, as evident in Figure 8. This suggests a very quick way of using the identified model.

4.2 Contrasting the Literature Results

Our approach to the identification problem is analogous to Expectation Minimization [43]. Expectation is at the basis of parameters inference, minimization of fitting. This allows us to both concentrate the fitting accuracy exactly in the domains involved by the sought parameters, and bootstrap the latter with great accuracy at a low computational cost. Unlike commonly in the literature, we do assume neither the performance parameters distribution to be Gaussian, nor the model parameters to be independent. Both hypotheses have been proven to hamper accurate solutions[44]. *Vice versa* their introduction isolates the merits of the identification principally to the fitting accuracy of the Spice functions [11]. Therefore we are forced to contrast our methods with those proposed in literature exactly in terms of running times and fitting accuracy.

4.2.1 Fitting Accuracy as Necessary.

Given the high computational costs of the Spice models, their approximation through cheaper functions is the first step in many numerical procedures

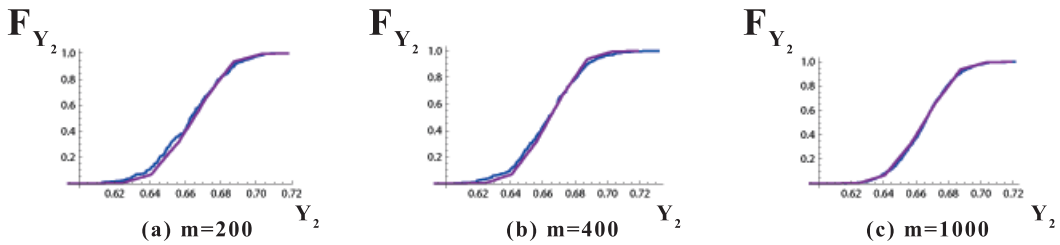

Figure 7. Robustness of the solution vs. information lack and methods. Same graphs as in Figure 3 when the third perfomance parameter is unavailable and the moment error minimization is pursued thorugh: (a) Variable Step Search, and (b) Steepset Gradient Descent coupled with Differential Evolution methods.

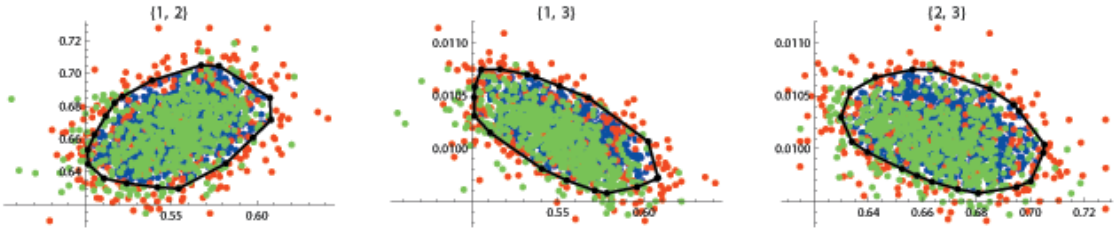

Figure 8. 0.8 tolerance region projected on paired components (identified by plot labels) on the nMOS benchmark with reduced model parameters as in Figure 5 computed via Spice mapping. Blue/red bullets: polynomially approximated performances lying within/outside the tolerance region; green bullets: target output; black bullets: delimiters of tolerance region (black curve).

on microelectronic circuits. Within the vast set of methods proposed by researchers on the matter [4, 17, 45–51] in Table 3 we report a numerical comparison between two well reputed fitting methods and ours (which we refer to as Granular Construct based (GC)).

The methods are Multivariate Adaptive Regression Splines (MARS)[45], i.e. piecewise polynomials, and Polynomial Neural Network (PNN)[52]. Namely, we consider the $\widetilde{\theta}_X$ reported in Table 1 as the result of the nMOS circuit identification. On the basis of these parameters and through Spice functions, we draw a sample of 250 pairs (x_r, y_r) that we used to feed both competitor algorithms and our own. In detail we used VariReg software [53, 54] providing an efficient implementation of both MARS and PNN. To ensure a fair comparison among all methods, we: i) set equal to 6 both the number of monomials in our algorithm and the maximum number of basis functions in MARS, where we used a cubic interpolation, and ii) employ the default configuration in PNN by setting the degree of single neurons polynomial equal to 2. Moreover, in order to understand how the various algorithms scale with the fitting domain, we repeat the procedure with a second set $\widetilde{\theta}'_X$ of parameters, where the original standard deviations have been uniformly doubled. In the table we report the mean squared errors measured on a test set of size 1000, whose values are both split on the four components of the performance vector (in the brackets) and resumed by their average (in italics). The comparison denotes similar accuracies with the most concentrated sample – the actual operational domain of our polynomials – and a small deterioration of our accuracy in the most dispersed sample, as a necessary price we pay for the simplicity of our fitting function.

4.2.2 Time Complexity.

As for the specific fitting procedures, their complexity analysis looks a superfluous task given the involved sizes: at most 1000 sample items, each one described by at most 10 features. All algorithms' executions last at most 1 second. This prevented us from testing Caffeine algorithm [55]. Though well reputed, this hybrid genetic algorithm indeed provides an accuracy that is either comparable or just a

	$\bar{\theta}_X$		$\bar{\theta}'_X$	
	train	test	train	test
GC	*0.0000125623* $\begin{pmatrix} 0.0000350975 \\ 0.0000151476 \\ 3.06034 \times 10^{-10} \\ 3.59774 \times 10^{-9} \end{pmatrix}$	*0.0000242739* $\begin{pmatrix} 0.0000759397 \\ 0.0000211444 \\ 6.62265 \times 10^{-10} \\ 1.10138 \times 10^{-8} \end{pmatrix}$	*0.000228931* $\begin{pmatrix} 0.000751481 \\ 0.000164105 \\ 1.54286 \times 10^{-8} \\ 1.24052 \times 10^{-7} \end{pmatrix}$	*0.000369871* $\begin{pmatrix} 0.00131925 \\ 0.000159924 \\ 2.33858 \times 10^{-8} \\ 2.92353 \times 10^{-7} \end{pmatrix}$
MARS	*8.68173∗10^−6* $\begin{pmatrix} 0.0000246876 \\ 0.0000100344 \\ 2.80773 \times 10^{-10} \\ 4.66935 \times 10^{-9} \end{pmatrix}$	*0.0000168024* $\begin{pmatrix} 0.0000528055 \\ 0.0000143915 \\ 5.92204 \times 10^{-10} \\ 1.19291 \times 10^{-8} \end{pmatrix}$	*0.000124012* $\begin{pmatrix} 0.000401349 \\ 0.0000946271 \\ 5.3722 \times 10^{-9} \\ 6.47147 \times 10^{-8} \end{pmatrix}$	*0.0002805* $\begin{pmatrix} 0.00100927 \\ 0.000112503 \\ 6.07291 \times 10^{-9} \\ 2.22601 \times 10^{-7} \end{pmatrix}$
PNN	*0.0000602061* $\begin{pmatrix} 0.000230822 \\ 0.0000100003 \\ 2.7761 \times 10^{-10} \\ 2.38434 \times 10^{-9} \end{pmatrix}$	*0.0000769737* $\begin{pmatrix} 0.000293665 \\ 0.0000142199 \\ 5.70282 \times 10^{-10} \\ 9.12621 \times 10^{-9} \end{pmatrix}$	*0.000125976* $\begin{pmatrix} 0.000409046 \\ 0.0000948249 \\ 4.14671 \times 10^{-9} \\ 2.84136 \times 10^{-8} \end{pmatrix}$	*0.000280898* $\begin{pmatrix} 0.00101197 \\ 0.000111354 \\ 7.14833 \times 10^{-9} \\ 2.62591 \times 10^{-7} \end{pmatrix}$

Table 2. Performance comparison between fitting algorithms. Rows: algorithms; main columns: benchmark parameterization; subcolumns: experimental environments (training set, test set).

bit higher than competitors, but at a cost of around 3 hours for the sample size managed by us [12].

As for the whole procedure, we reckon overall running times of around half an hour. Though not easily contrastable with computational costs of analogous tasks, this order of magnitude results adequate for an intensive use of the algorithm in a circuit design framework.

5 Conclusions

We solve a complex electronic manufacturing control problem using a granular construct. In spite of the methodology broadness the attribute *granular* may evoke, we obtain a very accurate solution benefitting from strict exploitation of state-of-the-art theoretical results. Starting from the basic idea of considering the Spice function as a mixture of fuzzy sets, we enriched its implementation with a series of sophisticated methodologies for: i) identifying clusters on the basis of proper adaptive metrics defined on functional spaces; ii) descending, direction by direction, along the *ravines* of the cost functions of the related optimization problems; iii) inverting the (X,Y) mapping in case of unbalanced problems through the bootstrapping of conditional Gaussian distributions; and iv) computing tolerance regions through convex hull based peeling techniques. In this way we supply a very accurate and fast algorithm to statistically identify the circuit model. We actually did not exploit all statistical features of the inversion problem, since we basically rely on first and second order moments of both source and destination distributions. Hence, as next step in this research vein, we plan to reintroduce the membership functions $\mu_k(x)$ of points to clusters and maintain the membership grades of the monomial distributions to the Y distribution, as an extension of the *a priori* probability of cluster k, obtaining *corrected* monomial densities. In this way we may come to an analytical expression of the Y distribution through which we aim to obtain a further refinement of the X parameters.

References

[1] Bernstein K., Frank D.J., Gattiker A.E., Haensch W., Ji B.L., Nassif S.R., Nowak E.J., Pearson D.J., Rohrer N.J., *High-performance CMOS variability in the 65-nm regime and beyond*. IBM Journal of Research Development 50:433–449, 2006.

[2] Boning D.S., Nassif S., *Models of process variations in device and interconnect*. In Chandrakasan, A. ed., Design of High Performance Microprocessor Circuits, chapter 6, IEEE Press, 1999.

[3] Buhler M., Koehl J., Bickford, J. Hibbeler, J. Schlichtmann, U. Sommer, R. Pronath, M., Ripp A., *DFM/DFY design for manufacturability and yield - influence of process variations in digital, analog and mixed-signal circuit design*. In: DATE06. 387–392, 2006.

[4] McConaghy T., Palmers P., Gao P., Steyaert M., Gielen G.G.E., *Variation-Aware Analog Structural Synthesis: A Computational Intelligence Approach*. Springer, 2009.

[5] Qu M., Styblinski M., *Parameter extraction for statistical IC modeling based on recursive inverse*

approximation. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 16, 1250–1259, 1997.

[6] Koskinen T., Cheung P., *Statistical and behavioural modelling of analogue integrated circuits*. Circuits, Devices and Systems, IEE Proceedings G 140, 171–176, 1993.

[7] Lee S.H., Choi C.H., Kong J.T., Lee W.S., Yoo J.H., *An efficient statistical analysis methodology and its application to high-density DRAMs*. In: Proceedings of the 1997 IEEE/ACM International Conference on Computer-Aided Design (ICCAD-97), Washington, DC, USA, IEEE Computer Society, 678–683, 1997.

[8] Cheng B., Dideban D., Moezi N., Millar C., Gareth R., Xingsheng W., Scott R., Asen, A., *Benchmarking statistical compact modeling strategies for capturing device intrinsic parameter fluctuations in BSIM4 and PSP*. IEEE Design and Test of Computers 99, 2010.

[9] Quarles T., Pederson D., Newton R., Sangiovanni-Vincentelli A.,Wayne C., Spice http://bwrc.eecs.berkeley.edu/Classes/icbook/SPICE/, 2009.

[10] Rohatgi V.K., *An Introduction to Probablity Theory and Mathematical Statistics*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1976.

[11] McConaghy T., Gielen G., *Analysis of simulation-driven numerical performance modeling techniques for application to analog circuit optimization*. In: Proceedings of IEEE International Symposium on Circuits and Systems, 2005.

[12] McConaghy T., Gielen G.G.E., *Double-strength CAFFEINE: fast template-free symbolic modeling of analog circuits via implicit canonical form functions and explicit introns*. Design Automation and Test in Europe, 269–274, 2006.

[13] Bolt M., Rocchi M., Engel J., *Realistic statistical worst-case simulations of VLSI circuits*. Semiconductor Manufacturing, IEEE Transactions on 4, 193–198, 1991.

[14] Kundert K.S., *The Designers Guide to SPICE and SPECTRE*. Kluwer Academic Publishers, Boston (1998)

[15] Apolloni B., Bassis S., Malchiodi D., Witold P., *The Puzzle of Granular Computing*. Volume 138 of Studies in Computational Intelligence. Springer Verlag, 2008.

[16] Eeckelaert T., Daems W., Gielen G., Sansen W., *Generalized simulation-based posynomial model generation for analog integrated circuits*. Analog Integr. Circuits Signal Process, 40:193—203, 2004.

[17] Hershenson M., Boyd S., Lee T., *Optimal design of a CMOS OP-AMP via geometric programming*. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 20:1–21, 2001.

[18] Liu W., Jin X., Xi X., Chen J., Jeng M.C., Liu Z., Cheng Y., Chen K., Chan M., Hui, K. Huang J., Tu R., Ko P.K., Hu C., *BSIM3v3.3MOSFETModel UsersManual*. Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, 2005.

[19] Bowman K.A., Duvall S.G., Meindl J.D., *Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration*. IEEE Journal of Solid-State Circuits 37:183–190, 2002.

[20] Rao R., Srivastava A., Blaauw D., Sylvester D., *Statistical estimation of leakage current considering inter- and intra-die process variation*. In: ISLPED 03: Proceedings of the 2003 international symposium on Low power electronics and design, New York, NY, USA, ACM.

[21] Eshbaugh K.S., *Generation of correlated parameters for statistical circuit simulation*. IEEE Transactions on CAD of Integrated Circuits and Systems 11:1198–1206, 1992.

[22] Mood A.M., Graybill F.A., Boes D.C., *Introduction to the Theory of Statistics*. McGraw-Hill, New York, 1974.

[23] Ning Q., *On the momentum term in gradient descent learning algorithms*. Neural Networks 12:145–151, 1999.

[24] Price K.V., Storn R.M., Lampinen J.A., *Differential Evolution, A Practical Approach to Global Optimization*. Volume 538 of Natural Computing Series. Springer, 2005.

[25] Moller M.F., *A scaled conjugate gradient algorithm for fast supervised learning*. Neural Networks 6:525–533, 1993.

[26] More J.J., *The Levenberg-Marquardt algorithm: Implementation and theory*. Numerical Analysis 630/1978:105–116, 1978.

[27] Duch W., Kordos M., *Multilayer perceptron trained with numerical gradient*. In: Proceedings of the International Conference on Artificial Neural Networks (ICANN) and International Conference on Neural Information Processing (ICONIP), Istanbul, 106–109, 2003.

[28] Duch W., Kordos M., *Variable Step Search algorithm for feedforward networks*. Neurocomputing 71:2470–2480, 2008.

[29] Nelder J.A., Mean R., *A simplex method for function minimization*. Computer Journal 7:308–313, 1965.

[30] Wolfram Research Inc., Mathematica 7, 2008.

[31] Nocedal J., Wright S.J., *Numerical Optimization*. Series: Springer series in operations research. Springer, New York, 1999.

[32] Zhang F., *The Schur Complement and Its Applications*. Volume 4 of Numerical Methods and Algorithms. Springer, Netherland, 2005.

[33] Jolliffe I.T., *Principal Component Analysis*. Springer Verlag, 1986.

[34] Hinton G.E., Sejnowski T.J., *Learning and relearning in Boltzmann Machines*. In Rumelhart, D.E., McClelland J.L., et al., eds., Parallel Distributed Processing: Volume 1: Foundations. MIT Press, Cambridge, 282–317, 1987.

[35] Cybenko G., *Approximations by superpositions of sigmoidal functions*. Mathematics of Control, Signals, and Systems 2:303–314, 1989.

[36] Leader J.J., *Numerical Analysis and Scientific Computation*. Addison Wesley, 2004.

[37] Aldenderfer M.S., Blashfield R.K., *Cluster Analysis*. Sage, Newbury Park (CA), 1984.

[38] Nikolaev N., de Menezes L.M., Iba H., *Overfitting avoidance in genetic programming of polynomials*. E-Commerce Technology, IEEE International Conference on 2:1209–1214, 2002.

[39] Cohn D., Ghahramani Z., Jordan M., *Active learning with statistical models*. Journal of Artificial Intelligence Research 4:129–145, 1996.

[40] Barnett V., *The ordering of multivariate data*. Journal of Royal Statistical Society Series A 139:319–354, 1976.

[41] Apolloni B., Bassis S., Gaito S., Malchiodi D., *Appreciation of medical treatments by learning underlying functions with good confidence*. Current Pharmaceutical Design 13:1545–1570, 2007.

[42] Liu R.Y., Parelius J.M., Singh K., *Multivariate analysis by data depth: Descriptive statistics, graphics and inference*. The Annals of Statistics 27:783–858, 1999.

[43] McLachlan G.J., Krishnan T., *The EM Algorithm and Extensions*. 2nd edn. Wiley Series in Probability and Statistics. Wiley-Interscience, 2008.

[44] Purviance J.E., Petzold M.C., Potratz C., *A linear statistical FET model using principal component analysis*. Microwave Theory and Techniques, IEEE Transactions on 37:1389–1394, 1989.

[45] Friedman J.H., *Multivariate adaptive regression splines*. Annals of Statistics 19:1–141, 1991.

[46] Daems S., Gielen G., Sansen,W., *Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 22:517–534, 2003.

[47] Taher H., Schreurs D., Nauwelaers B., *Extraction of small signal equivalent circuit model parameters for statistical modeling of HBT using artificial neural networks*. In: Gallium Arsenide Applications Symposium (GAAS 2005) 3—7 ottobre 2005.

[48] Hatami S., Azizi M.Y., Bahrami H.R., Motavalizadeh D., Afzali-Kusha A., *Accurate and efficient modeling of SOI MOSFET with technology independent neural networks*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 23:1580–1587, 2004.

[49] Vancorenland P., Van der Plas G., Steyaert M., Gielen G., Sansen W., *A layout-aware synthesis methodology for RF circuits*. In: ICCAD01: Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design, Piscataway, NJ, USA, IEEE Press.

[50] Ampazis N., Perantonis S.J., *Two highly efficient second order algorithms for training feedforward networks*. IEEE Transactions on Neural Networks 13:1064–1074, 2002.

[51] Ampazis N., Perantonis S.J., *Olmam neural network toolbox for matlab*, 2002.

[52] Elder IV J.F., Brown D.E., *Induction and polynomial networks. network models for control and processing*. In Fraser M., ed.: Intellect, Portland, OR, 143–198, 2000.

[53] Jekabsons G., *Varireg software*, http://www.cs.rtu.lv/jekabsons/, 2010.

[54] Jekabsons G., *Adaptive basis function construction: an approach for adaptive building of sparse polynomial regression models*. Machine Learning, In-Tech 28 In Press, 2010.

[55] McConaghy T., Eeckelaert T., Gielen G., *CAFFEINE: Template-free symbolic model generation of analog circuits via canonical form functions and genetic programming*. Design Automation and Test in Europe, 1070–1075, 2005.