Simulation-based Automatic Generation of Signomial and Posynomial Performance Models for Analog Integrated Circuit Sizing

Walter Daems, Georges Gielen, Willy Sansen
Katholieke Universiteit Leuven (Belgium), Department of Electrical Engineering, ESAT-MICAS

Abstract

This paper presents a method to automatically generate posynomial response surface models for the performance parameters of analog integrated circuits. The posynomial models enable the use of efficient geometric programming techniques for circuit sizing and optimization. To avoid manual derivation of approximate symbolic equations and subsequent casting to posynomial format, techniques from design of experiments and response surface modeling in combination with SPICE simulations are used to generate signomial and posynomial models in an automatic way. Attention is paid to estimating the relative 'goodness-of-fit' of the generated models. Experimental results allow to assess both the quality of the generated models as well as the strengths and the limitations of the presented approach.

Keywords

Analog Circuit Modeling, Posynomial and Signomial Response Surface Modeling, Geometric Programming, Design of Experiments

1 Introduction

Automatic sizing of transistor-level analog integrated circuits is currently one of the main research targets in the electronic design automation business. Time to market and limited engineering resources (which yield a higher return-on-investment on a higher design abstraction level) ask for the automation of this traditionally hand-crafted task.

Analog circuit sizing can be decomposed into two subtasks: (1) one "investigates" the circuit's behavior (knowledge acquisition) and (2) one adjusts the design parameters according to the obtained knowledge (knowledge use) [1]. The different solutions to the transistor-level sizing problem found in literature, can be subdivided into three main categories [2]. The distinction is based on the type of circuit knowledge acquisition employed:

- knowledge-based sizing: the knowledge of an expert designer is captured and encoded as a design plan or a set of rules, which are then executed during sizing [3], [4];
- equation-based sizing: the circuit's behavior is captured in a model that consists of a set of equations. These equations can be derived using automated or hand-crafted symbolic analysis [5], [6];
- simulation-based sizing: the circuit's behavior is obtained by numerical SPICE simulations at the transistor level and sizing is performed using optimization techniques [7], [8].

All techniques have their pros and cons. Simulation-based methods have the full accuracy of SPICE as trusted by most designers, whereas symbolic equations can only be derived in manageable CPU times in simplified approximate format. On the other hand, simulation-based circuit optimization is awfully slow, even when parallelizing the process on a farm of workstations, whereas symbolic methods are much faster in execution time.

Recently, it was demonstrated that the optimization of analog integrated circuits, like amplifiers, switched-capacitor filters, LC oscillators, etc., can be formulated as a geometric program [9], [10]. The circuits are characterized with symbolic equations that have to be cast in posynomial format. The advantages of a geometric program are (1) that the problem is convex and therefore has only one global optimum, (2) that this optimum is not correlated with the optimization's starting point and (3) that infeasible sets of constraints can be identified [11]. In addition the geometric program's optimum can be found extremely efficiently, using interior point methods [12], even for relatively large problems. The sizing is so fast (seconds) that design space explorations and process corner analysis can be performed quite easily.

The approach, however, also has a number of limitations today:

1) approximate symbolic equations that characterize the full circuit performance have to be derived. Despite the progress in computer-automated symbolic analysis techniques [6], these are still mainly limited to small-signal characteristics, necessitating the manual derivation of large-signal and transient characteristics;

2) the equations have be to cast in posynomial format. Although [9] and [10] show that many circuit characteristics are posynomial, this is not the case for all characteristics! In this case, the equations have to be approximated by some posynomial model. This is a manual effort with difficult control on the approximation error. Up till now, no automatic posynomial recasting approach has been presented;

3) also the device models have to be approximated in posynomial format [9]. This introduces an extra error, especially for deep submicron CMOS processes. The results will differ from well established and accepted models like BSIM-3v3, used and trusted by designers today.

This paper presents an approach to overcome these limitations. Our approach automatically generates signomial and posynomial models for all circuit characteristics based on numerical SPICE simulations using accurate device models. It keeps the simulation time to a strict minimum by applying techniques from design of experiments and by parallelizing the simulations over the available computing equipment. It also optimizes the use of computing resources by applying a simple but effective load-balancing technique. The strong points of the approach are therefore:

1) no a-priori generation of simplified equations needed: the models are built from SPICE simulations,

2) not only posynomial but more general signomial circuit characteristics and device models are allowed,

3) the full accuracy of SPICE simulations and established device models, such as BSIM-3v3 or MM9, is used to generate the models.

The paper is organized as follows. In section 2 we will discuss performance modeling in general and set out some cornerstones used in our approach. Section 3 describes the fundamentals of our posynomial modeling technique. The modeling algorithm has been implemented in a software prototype that will be highlighted in section 4. Section 5 provides experimental results obtained with the prototype. Finally, conclusions will be drawn in section 6.

2 Performance modeling

Consider a system S transforming an input signal E into an output signal Y (Fig. 1-(a)). The mathematical modeling of this input-output relationship is called *behavioral modeling*. Fig. 1-(b) depicts the same system seen from a designer's point of view: a number of design parameters (X) cause the system to exhibit a particular performance (P). The modeling of this relationship is called *performance modeling*. This paper concentrates on the latter.

2.1 Black-box fitting techniques

In general, a mathematical model is assumed, and the parameters of the model are fitted, so that the model corresponds as closely as possible to

Figure 1. Electronic system seen (a) as a system that relates an input signal E to an output signal Y and (b) as a system for which a set of design parameters X have to be chosen to obtain a specified performance P

simulated or measured data of the real system. Standard mathematical fitting techniques like interpolation or least-squares regression, go back several centuries and are well known. However, the use of these techniques in analog integrated circuit design has always been rather limited. The advantage of black-box modeling is obvious: no intrinsic system knowledge is needed in the fitting process. The model is built from input-output responses only, hence the name "black-box" fitting.
The opposite is true for the model selection: selecting a model that is suited to capture all important effects is a hard task. Understanding the nature of the system is very beneficial in that case. We will assume a quite general model that however satisfies desired mathematical properties.
Notice also that in general such models have a trade-off between generality and accuracy for a given model complexity. This means that a model that has to be valid over a large range of design parameters X typically has a larger fit error for the performances P than a model that only has to be valid for a small range of X.

2.2 Design of Experiments

Techniques from design of experiments provide a mathematical basis to select a limited but "optimal" set of sample points needed to fit a black-box model [13]. Design of experiments considers the fitting process as an estimation of model parameters. E.g. sampling schemes can be composed that guarantee uncorrelated model parameter estimation to a certain degree for a specific model.
Well-known and often-used sampling schemes range from full- and fractional factorial design, over Placket-Burman and Taguchi schemes, to Latin hypercube and even random design. The choice and the literature is vast. A very interesting sampling scheme, however, are orthogonal arrays, more specifically level-2 orthogonal arrays of strength 3 [14]. This scheme places the sampling points on the edges of a fitting hypercube of size $(2dx)^n$ around a center point X^*. The 4-dimensional level-2 orthogonal array of strength three around 0 illustrates this:

$$\begin{bmatrix} -dx & -dx & -dx & -dx \\ -dx & -dx & dx & dx \\ -dx & dx & -dx & dx \\ -dx & dx & dx & -dx \\ dx & -dx & -dx & dx \\ dx & -dx & dx & -dx \\ dx & dx & -dx & -dx \\ dx & dx & dx & dx \end{bmatrix}$$

Usually the center point is provided as well. Orthogonal arrays of strength 3 allow the uncorrelated estimation of linear, quadratic and interaction terms of a second-order polynomial phenomenon. This is a very beneficial property when fitting local models of a more general system.
The combination of design of experiments with a standard fitting technique often is called *response surface modeling (RSM)* [15]. This technique (using orthogonal arrays) will allow us in this paper to efficiently generate models that have a specific desired format.

2.3 Geometric programming basics

Let $X = (x_1, x_2, x_3, \ldots, x_n)^T$ be a vector of real, positive variables. A function f is called *signomial* if it has the form

$$f(X) = \sum_{i=1}^{m} \left(c_i \prod_{j=1}^{n} \left(x_j^{\alpha_{ij}} \right) \right) \tag{1}$$

with $c_i \in \mathcal{R}$ and $\alpha_{ij} \in \mathcal{R}$. If we restrict all c_i to be positive ($c_i \in \mathcal{R}^+$), then the function f is called *posynomial*. If in addition $m = 1$, then the function f is called *monomial*. If all exponents α_{ij} are integer, then the signomial becomes a *polynomial*. Our modeling approach will generate

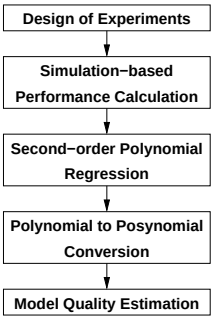

Figure 2. Posynomial model generation flow

signomial and posynomial models. The latter allow to formulate analog circuit sizing as a geometric program.
A (primal) geometric program is the constrained optimization problem:

$$\text{minimize } f_0(X) \tag{2}$$

$$\begin{aligned} \text{with the constraints: } f_i(X) &\leq 1, \quad i = 1, \ldots, p \\ g_j(X) &= 1, \quad j = 1, \ldots, q \\ x_k &\geq 0, \quad k = 1, \ldots, n \end{aligned} \tag{3}$$

with all $f_i(X)$ posynomial and all $g_j(X)$ monomial.
By substituting all variables x_i by $z_k = log\,(x_k)$ and taking the logarithm of the objective function and every constraint, it can readily be seen that the transformed problem is a convex optimization problem. Hence, it has only one global optimum. In addition this optimum can very efficiently be found using interior point methods [12].
Notice that if any f_i or g_j contains a negative constant term (which makes them theoretically speaking non–posynomial), the nature of the geometric program is not harmed. Indeed, the constant term can easily be eliminated from the left-hand side of the constraints. Since shifting the goal function by a constant amount does not influence the location of the optimum in terms of x_k, the same statement holds here as well.

3 Generation of posynomial models

The posynomial performance models (needed for the geometric programming formulation) can be derived using symbolic circuit analysis. In [9] and [10] this hand-crafted process is demonstrated. This is however a non–systematic, tedious process. As far as we know, automated posynomial model generators have not yet been reported in literature. We will present such an approach here. Fig. 2 illustrates the adopted procedure. Based on the results of a well-designed set of simulation experiments, we will fit a second-order polynomial (which at the same time is a kind of signomial). Next, this model will automatically be converted into a posynomial model. Finally the quality of this model is estimated. The next sections will focus on the signomial regression, the posynomial conversion and the quality estimation.

3.1 Second-order polynomial regression

Consider a set of a experiments $X_j, j = 1 \ldots a$ that correspond to a set of a performances $p_j, j = 1 \ldots a$. A model $M(X, c_1, c_2, \ldots c_m)$, linear w.r.t. m fit parameters c_m, needs to be fitted such that the distance between the system's performance and the performance of the model is minimal. Several distance measures can be used for fitting the model parameters. A commonly used measure is the sum of the square performance deviations over all sample points, leading to the well-known *least squares regression*. The least squares fitting of a signomial function $f(X)$

$$f(X) = \sum_{i=1}^{m} c_i t_i \tag{4}$$

with $t_i = \prod_{l=1}^{n} x_l^{\alpha_{il}}$, can be formulated as a minimization problem:

$$\text{minimize } \Psi = \sum_{j=1}^{a} \left(\left[\sum_{i=1}^{m} (c_i t_{i,j}) - p_j \right]^2 \right) \tag{5}$$

with $t_{i,j}$ the actual value of t_i for experiment j. This corresponds to the minimization of a positive-definite general quadratic form:

$$\Psi = \Psi(0) + C^T B + \frac{1}{2} C^T \mathbf{A} C \tag{6}$$

with $B = \nabla\Psi$ and $\mathbf{A}$ represents the Hessian of Ψ. Usually this problem is solved by solving $\mathbf{A}C = -B$ which in our case corresponds to solving:

$$\mathbf{T}\mathbf{T}^t = P \tag{7}$$

Considering that $\mathbf{T}\mathbf{T}^t$ is symmetric, one might be tempted to solve (7) straight away using e.g. a Cholesky decomposition. However, the condition number of $\mathbf{T}\mathbf{T}^t$ is approximately equal to the square of the condition number of $\mathbf{T}$. This results in an ill-conditioned solution of (7), especially when adopting asymmetrical sampling schemes or sampling schemes that are not close to the unity matrix. Since for both signomial and posynomial models the requirement $x_i \geq 0, \forall i$ holds, our sampling scheme becomes most asymmetrical.

If we restrict ourselves to second-order polynomial models, we can easily overcome this burden by first applying an axis translation that makes the sampling symmetrical, then solving (7) and finally restoring the original axes. This procedure leaves the second-order polynomial nature of the model intact. Restricting ourselves to a second-order polynomial model has another advantage: we can convert it without too great difficulty to an *approximate posynomial model*. Besides, it does not put a severe hold on the quality of the resulting models. The reason for this is the fact that a lot of performance parameters for integrated circuits are very near to simple sums and additions of design parameters if the latter are well chosen and proper scaling is applied.

E.g. consider the following approximate expressions for some performance parameters of CMOS operational amplifiers:

$$GBW = \frac{2I_{DS,in}}{V_{GST,in}C_{dom}} \quad \text{and} \quad SR = \frac{I_{bias}}{C_{dom}} \tag{8}$$

When logarithmically scaling the performance space as well as the design space, these parameters become fully linearly signomial.

It should be noted also that the number of fit parameters in a generic n-dimensional k-th order polynomial model is $O(n^k)$. This in general hinders the use of higher-order models.

We will now explain how the signomial model is converted in an approximate posynomial model, so that it becomes useful for geometric programming. Consider the generic n-dimensional second-order polynomial model:

$$f(X) = c_0 + \sum_{i=1}^{n} (c_i x_i) + \sum_{i=1}^{n} \left(c_{i,i} x_i^2\right) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} (c_{i,j} x_i x_j) \tag{9}$$

The terms that have a positive coefficient c_i or c_{ij} are monomial terms. The terms that have a negative coefficient can be approximated around a center point X^* by posynomial approximations as follows:

- negative linear terms:

$$c_i x_i \approx \frac{d_i}{x_i} + b_i \tag{10}$$

d_i and b_i are chosen such that the function value and the first derivative are maintained, i.e.:

$$d_i = -c_i \left(x_i^*\right)^2 \tag{11}$$
$$b_i = 2\, c_i x_i^* \tag{12}$$

- negative interaction terms:

$$c_{i,j} x_i x_j \approx \frac{d_{i,j} x_j}{x_i} + b_{i,j} x_j \tag{13}$$

$d_{i,j}$ and $b_{i,j}$ are chosen such that the function value and the first derivative are maintained, i.e.:

$$d_{i,j} = -c_{i,j} \left(x_i^*\right)^2 \tag{14}$$
$$b_{i,j} = 2\, c_{i,j} x_i^* \tag{15}$$

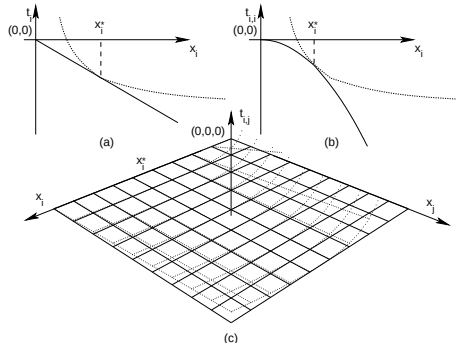

Figure 3. Posynomial approximation of (a) negative linear terms, (b) negative quadratic terms and (c) negative interaction terms. The original terms are plotted in solid lines, the approximate posynomials in dotted lines.

Of course the role of x_i and x_j can be interchanged here. The choice is made based upon which approximation harms the posynomiality of the linear term least (i.e. changing the sign of c_i or c_j). If both choices destroy the posynomiality of the linear term or both linear terms are already non–posynomial, then we substitute the largest variable. If one of the linear terms is still posynomial and we can keep it that way by replacing the other term, then we do so. Otherwise we replace the posynomial term. This procedure guarantees the best possible approximation.

- negative quadratic terms:

$$c_{i,i} x_i^2 \approx \frac{d_{i,i}}{x_i} + b_{i,i} \tag{16}$$

$d_{i,i}$ and $b_{i,i}$ are chosen such that the function value and the first derivative are maintained, i.e.:

$$d_{i,i} = -2\, c_{i,i} \left(x_i^*\right)^3 \tag{17}$$
$$b_{i,i} = 3\, c_{i,i} \left(x_i^*\right)^2 \tag{18}$$

Fig. 3 illustrates these approximations. It is clear that these approximations make a model that favors a good fit at the center point X^*. However, when used during circuit sizing, this center point will move and the model will be updated adaptively [10], [11].

Applying the above approximations in reverse order removes any signomial term from the model, except for the constant term which most likely becomes negative. This poses no problem since when appearing in any geometric program, this constant term can easily be eliminated. The computational complexity of the above procedure is $O(n^2)$.

3.2 Model quality assessment

As model quality parameter we took the root mean square deviation in the existing sampling points as starting point. We normalized this quality figure by dividing it by the performance range of the sample points, leading to a relative quality figure q for model M:

$$q = \frac{\sqrt{\sum_{j=1}^{a} \left(M(X_j) - p_j\right)}}{a\left[c + \left(\max_{j=1}^{a} p_j - \min_{j=1}^{a} p_j\right)\right]} \tag{19}$$

with c a constant to avoid error overestimation when the performance range approaches zero. This figure is:

1) computationally cheap (no extra simulations are needed),

2) easy to assess: a quality larger than 1 suggests a bad fit[1].

In addition it must be said that this figure is rather a pessimistic measure for the posynomial model approximation. Indeed: the use of orthogonal arrays to position the samples within the fitting hypercube, results in a sampling points — except for the center point — located at the extremes of the fitting hypercube, where the approximation error is largest. Remember that the approximation was conceived such as to minimize the modeling error in the center point.

[1] Notice that the fitting technique itself might not always be the cause, also the fitting capabilities of the model for the given circuit at hand are involved.

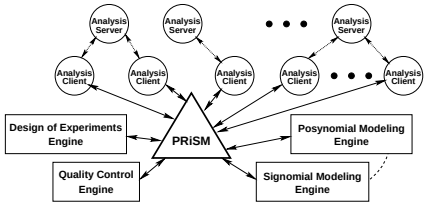

Figure 4. Overall concept of PRiSM

In order to have a more realistic quality assessment for these models, we introduce two extra relative quality figures:
1) the relative deviation in the center point, which we will label q_{oc}
2) the quality figure of (19) evaluated in sampling points located in the interior of the fitting hypercube, which we will label q_{tc} [2]
A drawback of the latter is the need for extra analyses (i.e. circuit simulations). Notice that for the generated posynomial models q can be considered as the worst-case value (we therefore give it an extra subscript: q_{wc}), q_{tc} as the typical-case value and q_{oc} as the optimal-case value.

4 Modeling prototype

The signomial and posynomial fitting techniques have been implemented in PRiSM, our Posynomial Response Surface Modeling prototype. Fig. 4 shows the overall concept of PRiSM. The core engines have been coded using C++, while the analysis clients and servers have been coded in Perl. The total amounts to about 40 000 lines of code. In the design a number of extra design principles were taken into account, based on need observations within our analog integrated circuit design group. The principles are:

- Resource parallelization
- Load balancing
- Analysis abstraction

PRiSM's TCP-based client-server simulation system schedules simulation and extraction jobs on remote workstations over the intra-net or the Internet (*resource parallelization*). Upon completion the duration of each analysis-extraction job is preserved in a server-load database. During the generation of a model (which typically requires hundreds of simulations) the average analysis-extraction duration for a server is taken as its average load. At regular times this load information adjusts the client-server job scheduling plan (*load balancing*).
SPICE simulations are performed using a standard simulation tool chosen according to the availability or preference. At this moment interfaces to ELDO and Berkeley SPICE 3f4 [3] are provided. At the client side an abstraction layer totally hides the specific nature of these simulation tools, allowing easy migration of PRiSM to a new simulator environment.

5 Experimental results

As test case we used the circuit of Fig. 5, a high-speed CMOS OTA which we want to model using a $0.7\mu m$ CMOS technology from Alcatel Microelectronics. The supply voltage is 5V. The nominal threshold voltages of this technology are $0.76V$ for NMOS-devices and $-0.75V$ for PMOS-devices. The circuit has to drive a load capacitance of 10pF.
Thirteen independent design variables were retained as inputs for the models. In Table I an overview of the chosen variables and their bounds is given. Note that currents and transistor drive voltages are chosen as variables, rather than transistor widths, since we use an operating-point driven formulation for analog circuit sizing [16]. The bounded range of variables $v_i \in [lb_i, ub]$ is logarithmically scaled onto $x_i \in [0, 1]$ using

$$x_i = \log\left(\frac{v_i}{lb}\right) \log\left(\frac{lb}{ub}\right) \tag{20}$$

[2]For the experimental results we also used an orthogonal array hypercube with a size 1/3 of the original size.

[3]For this purpose we integrated the BSIM-3v3 model into Berkeley SPICE 3f4.

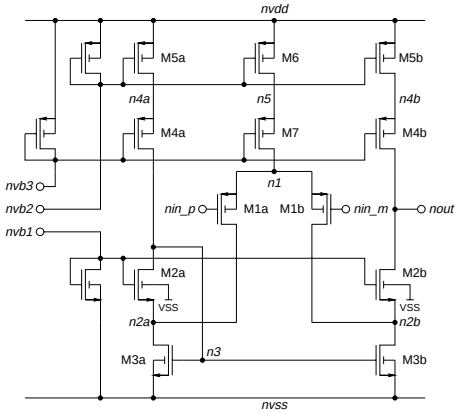

Figure 5. Schematic of a high-speed CMOS OTA

i	v_i	lb	ub	i	v_i	lb	ub
1	$v_{GS,M1}$	-0.75V	-4V	2	$v_{GS,M2}$	0.75V	4V
3	$v_{DS,M2}$	0.1V	4V	4	$v_{GS,M3}$	-0.75V	-4V
5	$v_{GS,M4}$	-0.75V	-4V	6	$v_{GS,M5}$	-0.75V	-4V
7	$v_{DS,M5}$	-0.1V	-4V	8	$v_{DS,M6}$	-0.1V	-4V
9	$i_{D,M1}$	-10uA	-10mA	10	$i_{D,M2}$	10uA	10mA
11	i_{B1}	1uA	100uA	12	i_{B2}	1uA	100uA
13	i_{B3}	1uA	100uA				

Table I. Design variables chosen as model inputs

Six parameters of the OTA will be modeled:
- the low-frequency gain (A_{LF}), the unity-gain frequency (f_u) and the phase margin PM, whose actual values will be determined from an AC simulation,
- the offset at the input (v_{off}) determined from a DC sweep analysis,
- the positive slew rate and negative slew rate (SR_p, SR_n) that will be extracted from a transient simulation.

For all six model parameters, two models are generated around the center point $x_i = 0.5, \forall i$: a second-order polynomial (signomial) model (SM), and a posynomial model with integer exponents -1, 0, 1, 2 containing level-2 interactions (PM). The nominal performance in the center point can be found in table II. In advance all performances p_i are scaled either linearly (A_{LF}, PM, v_{off}, SR_p, SR_n), according to:

$$p_{i,lin} = \pm\frac{1}{1 + |p_{i,typ}|}\left(p_i - p_{i,typ}\right) \tag{21}$$

with $p_{i,typ}$ a typical value of the parameter (e.g. the target specification), or scaled logarithmically (f_u), according to:

$$p_{i,log} = \pm\frac{1}{1 + \log\left(p_{i,typ}\right)} \log\left(\frac{p_i}{p_{i,typ}}\right) \tag{22}$$

The plus sign in (21) and (22) is chosen if a model for minimization of the parameter is required, the minus sign if a model for maximization is required.
Each model is generated twice keeping the same center point for different relative sizes ($dx = 0.1, 0.01$) of the fitting hypercube. The fitting was carried out using an orthogonal array of strength 3 of dimension 243×13, i.e. 243 experiments each requiring three SPICE simulations were carried out to obtain the experimental data. For the case where $dx = 0.1$ for some experiments the DC-solution proved to be not physical or the simulations did not converge. These experiments were retained from the fit process. This of course ruins the desired orthogonality properties of the orthogonal arrays we use. However, as mentioned before, these properties only become relevant as soon as the relative size of the fitting hypercube becomes small.

A_{LF}	36.45 dB	f_u	6.761 MHz	PM	90.28^o
v_{off}	2.646 uV	SR_p	21.7 V/us	SR_n	-26.2 V/us

Table II. Performance of the CMOS OTA for $\mathbf{x_i = 0.5, \forall i}$

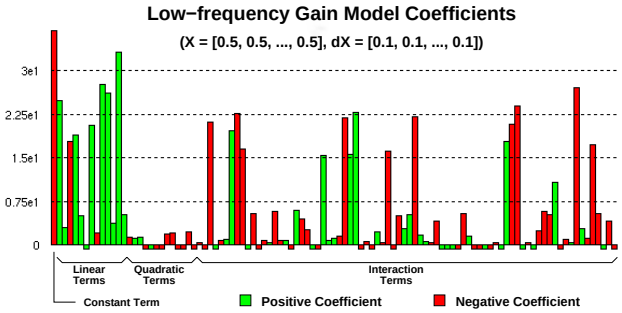

Figure 6. Graphical representation of the model coefficients of the (signomial) model for $\mathbf{A_{LF}}$ for the case $\mathbf{dx = 0.1}$.

param	range	σ	q
$dx = 0.1$			
A_{LF}	0.93662	0.13153e-1	1.404 %
f_u	0.12683	0.59763e-3	0.471 %
PM	0.16144	0.30017e-2	1.859 %
v_{off}	0.10203e-3	0.31287e-5	1.549 %
SR_p	1.11368	0.53327e-1	4.788 %
SR_n	2.00118	0.45433e-1	2.270 %
$dx = 0.01$			
A_{LF}	0.10409	0.25744e-3	0.247 %
f_u	0.14651e-1	0.73839e-5	0.504 %
PM	0.84243e-2	0.49900e-3	5.924 %
v_{off}	5.18427e-6	6.25512e-7	0.595 %
SR_p	0.12833	0.25624e-2	1.997 %
SR_n	0.17249	0.48414e-2	2.807 %

Table III. Properties of the signomial models (SM)

PRIMO was run on an Intel Celeron 466MHz running Linux. The analysis servers ran on 17 UNIX workstations, ranging from a SUN Ultra Sparc I (with a SPECfp95 of 9) to an HP B-1000 (with a SPECfp95 of 42) using their native OS. It took approximately 3 min. to generate all signomial models. The posynomial approximation took another 10 sec.

Fig. 6 illustrates the model coefficients of the signomial model for A_{LF} for the case $dx = 0.1$. It can be seen that in general an unconstrained-fitted model is not posynomial since many negative coefficients occur.

Table III shows the results for the origally fitted signomial models, whereas table IV shows the results for the derived posynomial models. The range, the standard deviation in the fit points (σ) and the quality figure(s) have been given for every modeled parameter. We let $c = 0.001$ (see (19)) which is a realistic value in view of (21) and (22). The parameter c only has significant influence for the offset model. The reason for this is that the operating-point driven formulation makes the offset negligible "by construction". This also explains why the nominal value for the offset (see Table II) is so low.

Several conclusions are to be drawn from these results:

- the signomial fits are in general quite good.
- the posynomial approximated models are useful for limited hypercube sizes. The anomaly of the PM-model for $dx = 0.01$ is due to numerical instabilities[4] during fit and approximation calculations.

param	range	σ	q_{wc}	q_{tc}	q_{oc}
$dx = 0.1$					
A_{LF}	0.93662	0.53034e+1	566.228 %	52.329 %	-1.123 %
f_u	0.12683	0.11825e+2	9324.387 %	890.198 %	1.366 %
PM	0.16144	0.10254e+2	6351.869 %	617.059 %	1.061 %
v_{off}	0.10203e-3	0.77042e-4	38.135 %	4.372 %	-0.981 %
SR_p	1.11368	0.18157e+2	1630.345 %	139.408 %	-14.669 %
SR_n	2.00118	0.10335e+2	516.463 %	48.227 %	0.275 %
$dx = 0.01$					
A_{LF}	0.10409	0.75125e-2	7.217 %	0.760 %	-0.365 %
f_u	0.14651e-1	0.10013e-2	6.834 %	0.835 %	0.070 %
PM (*)	0.84243e-2	0.51649	6131.264 %	559.083 %	32.224 %
v_{off}	5.18427e-6	0.98512e-5	9.451 %	1.871 %	-0.797 %
SR_p	0.12833	0.60875	474.354 %	45.924 %	-2.862 %
SR_n	0.17249	0.36425	211.172 %	23.814 %	-0.081 %

Table IV. Properties of the posynomial models (PM)

6 Conclusions

This paper has presented a technique to automatically generate posynomial performance models from full-accuracy SPICE simulations. Both signomial and posynomial models can be generated. The latter can be employed in automatic circuit sizing using highly efficient interior-point geometric programming algorithms. Our approach reduces the time and the effort needed to generate posynomial models to a strict minimum. The results are promising, but indicate room for improvement a.o. concerning numerical stability and the fitting of more specific (non–polynomial) posynomial models that might enable further reduction of the fit error for large hypercube sizes.

References

[1] L. R. Carley, G. Gielen, R. A. Rutenbar, and W. Sansen, "Synthesis tools for mixed-signal ics: Progress on frontend and backend strategies," in *Proc. Design Automation Conference*, June 1996, pp. 298–303.

[2] G. Gielen and R. A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proc. of the IEEE*, vol. 88, no. 12, pp. 1825–1852, Dec. 2000.

[3] G. R. Degrauwe, "IDAC: an interactive design tool for analog cmos circuits," *IEEE Journal of Solid-State Circuits*, vol. 22, no. 6, pp. 1106–1116, Dec. 1987.

[4] F. M. El-Turky and E. Perry, "BLADES: an artificial intelligence approach to analog circuit design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 6, pp. 680–691, June 1989.

[5] G. Gielen, G. Debyser, K. Lampaert, F. Leyn, K. Swings, G. Van der Plas, W. Sansen, D. Leenaerts, P. Veselinovic, and W. van Bokhoven, "An analogue module generator for mixed analogue/digital asic design," *International Journal of Circuit Theory and Applications*, vol. 23, pp. 269–283, 1995.

[6] F. V. Fernández-Fernandéz, A. Rodríguez-Vázquez, J. L. Huertas, and G. Gielen, *Symbolic analysis techniques – Applications to Analog Design Automation*, IEEE Press, 1998.

[7] F. Medeiro, F. V. Fernández-Fernandéz, R. Domínguez-Castro, and A. Rodríguez-Vázquez, "A statistical optimization-based approach for automated sizing of analog cells," in *Proc. Design Automation Conference*, 1994, pp. 594–597.

[8] R. Phelps, M. Krasnicki, R. A. Rutenbar, L. R. Carley, and J. R. Hellums, "Anaconda: simulation-based synthesis of analog circuits via stochastic patter search," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, pp. 703–717, 2000.

[9] M. Hershenson, S. P. Boyd, and T. H. Lee, "Optimal design of a CMOS op-amp via geometric programming," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 1–21, Jan. 2001.

[10] P. Mandal and V. Visvanathan, "CMOS op-amp sizing using a geometric programming formulation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 22–38, Jan. 2001.

[11] R. J. Duffin, C. Zener, and E. L. Peterson, *Geometric programming: theory and application*, John Wiley & Sons, New York, 1967.

[12] K. O. Kortanek, X. Xu, and Y. Ye, "An infeasible interior-point algorithm for solving primal and dual geometric programs," *Math. Programming*, vol. 76, pp. 155–181, 1996.

[13] George E. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for experimenters: an introduction to design, analysis and model building*, John Wiley & Sons, New York, 1978.

[14] A. S. Hedayat, Sloane N. J. A., and J. Stufken, *Orthogonal arrays: theory and applications*, Springer-Verlag, New York, 1999.

[15] George E. P. Box and N. R. Draper, *Empirical Model Building and Response Surfaces*, John Wiley & Sons, New York, 1987.

[16] F. Leyn, G. Gielen, and W. Sansen, "An efficient dc root solving algoritm with guaranteed convergence for analog integrated cmos circuits," in *Proc. IEEE/ACM International Conference on Computer Aided Design*, Nov. 1998, pp. 304–307.

[4] Increasing the accuracy of the computations results in quality figures comparable to those for the other models.