An Efficient Optimization–based Technique to Generate Posynomial Performance Models for Analog Integrated Circuits

Walter Daems
K.U.Leuven, ESAT–MICAS
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium
daems@esat.kuleuven.ac.be

Georges Gielen
K.U.Leuven, ESAT–MICAS
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium
gielen@esat.kuleuven.ac.be

Willy Sansen
K.U.Leuven, ESAT–MICAS
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium
sansen@esat.kuleuven.ac.be

ABSTRACT

This paper presents an new direct–fitting method to generate posynomial response surface models with arbitrary constant exponents for linear and nonlinear performance parameters of analog integrated circuits. Posynomial models enable the use of efficient geometric programming techniques for circuit sizing and optimization. The automatic generation avoids the time–consuming nature and inaccuracies of handcrafted analytic model generation. The technique is based on the fitting of posynomial model templates to numerical data from SPICE simulations. Attention is paid to estimating the relative 'goodness–of–fit' of the generated models. Experimental results illustrate the significantly better accuracy of the new approach.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids; B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms

Performance, Design, Algorithms

Keywords

Performance Modeling for Analog Circuits, Posynomial Response Surface Modeling, Geometric Programming

1. INTRODUCTION

The sizing of transistor–level analog integrated circuits is a time–consuming and thus expensive step in the design of analog and mixed–signal circuits. Automation of this process is currently an important research target in the electronic design automation community.

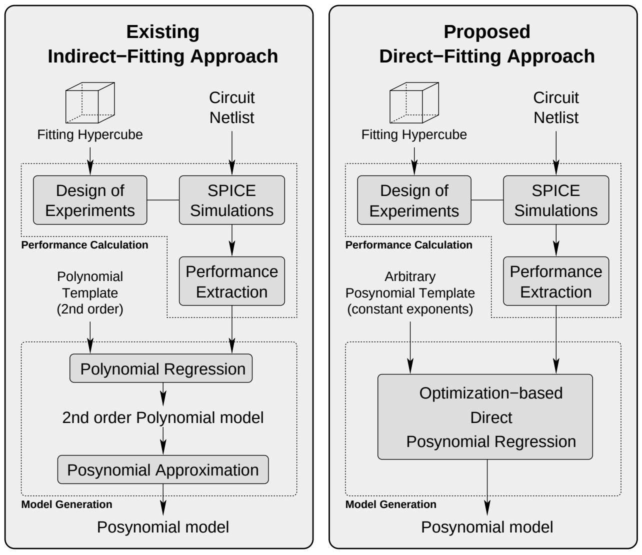

Figure 1: Comparison of the existing approach from [5] and the novel proposed direct–fitting approach

In the course of that research, it was demonstrated that the sizing of analog integrated circuits, like amplifiers, switched–capacitor filters, LC oscillators, etc., can be formulated as a geometric program [1, 2]. In that approach, the performance characteristics of the circuits are represented as symbolic equations written in posynomial format.

The advantages of a geometric program are multiple [3]:

1. The problem is convex and has only one global optimum.
2. The optimization is not dependent on the starting point.
3. Infeasible sets of constraints can be identified.
4. The geometric program's optimum can be found extremely efficiently, using interior point methods [4], even for relatively large problems.

The problem with the approaches in [1] and [2], however, is that the symbolic equations have to be derived manually, and that non-posynomial expressions have to be approximated manually into posynomial format. Both steps are time–consuming and can result in large inaccuracies, since e.g. also the device models have to be cast in posynomial format [1].

Recently, a method to automatically generate posynomial models for linear and nonlinear performance characteristics has been presented in [5]. The outline of the method has been sketched on the left–hand side of Fig. 1. The method first calculates the performance of a circuit for a set of samples that is composed using techniques from *design of experiments*. The data points can be generated using full–accuracy SPICE simulations. Then, a second–order polynomial template is fit to the sampled performance data. Finally, this model is approximated to render it posynomial. In this way it is possible to generate posynomial models of the form

$$p(X) = \sum_{k=-1,0,1} \sum_{\substack{l=-1,0,1 \\ \neg((k=-1)\wedge(l=-1))}} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \left(c_{i,k,j,l} x_i^k x_j^l \right) \right) \qquad (1)$$

with n the number of design parameters and all $c_{i,k,j,l}$ positive constants.

However the method presented in [5] exhibits some limitations and problems:

1. Only models of the type of (1) can be generated, i.e. the exponents are restricted to integers between -2 and 2.
2. The fit quality for larger fitting hypercube sizes is poor.
3. Problems concerning numerical stability occur.

This paper describes an alternative direct–fitting approach that overcomes these problems. The idea behind the proposed method can be seen on the right–hand side of Fig. 1. Again, the performance of the circuit is sampled using numerical simulations. But instead of first fitting a polynomial model and then approximating it, we immediately fit the wanted posynomial model using an optimization–based fitting technique. The strong points of the proposed method are:

1. its generality: any posynomial model with arbitrary constant exponents for any linear and nonlinear circuit characteristic can be fitted;
2. its superior fit quality;
3. its capability of generating sparse (i.e. compact) models.

The paper is organized as follows. Section 2 will provide some theoretical background concerning *posynomial performance modeling*. In section 3, we will concisely indicate the principles of the indirect–fitting approach of [5], followed by a detailed description of our novel direct–fitting approach. To show the greatly improved performance, the experimental results obtained with our method will be compared in section 4 to the ones obtained in [5]. Finally, section 5 draws some conclusions and provides some ideas for future research.

2. POSYNOMIAL PERFORMANCE MODELING

2.1 Performance modeling

Consider a system S transforming an input signal E into an output signal Y (Fig. 2). The system S is governed by a set of design parameters X that influence its behavior.

$$Y = S(E, X) \qquad (2)$$

The mathematical modeling of this input–output relationship is called *behavioral modeling*. In this paper, we are not directly interested in this relationship as such. However, the combination (Y, E) allows us to determine a particular performance p_i of the system, subject to excitation Y. Gathering a set of particular performances p_i into a vector gives us an idea of the total performance of the system.

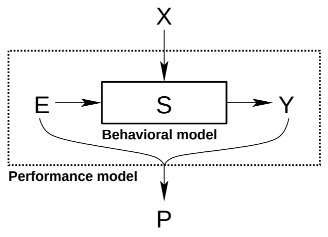

Figure 2: Electronic system S parametrized by X with excitation E and response Y leading to a performance P

Therefore, seen from a designer's point of view, the key relationship is:

$$P = F(X) \qquad (3)$$

The mathematical modeling of this relationship is called *performance modeling*.

2.2 Posynomials & Geometric Programming

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

$$f(X) = \sum_{i=1}^{m} \left(c_i \prod_{j=1}^{n} \left(x_j^{\alpha_{ij}} \right) \right) \qquad (4)$$

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 expression f is called *posynomial*. A posynomial consisting of a single term is called a *monomial*. The signomial fitting problem now can be defined as:

Signomial fitting problem:

Given a set of performance data samples

$$\{(X_1, p_{i,1}), (X_2, p_{i,2}), \ldots, (X_a, p_{i,a})\}$$

and a model template of type (4) with given exponents $\alpha_{ij} \in \mathcal{R}$,
determine the coefficients c_i, with $c_i \in \mathcal{R}$, such that

$$\left\| \left[f(X_1), f(X_2), \ldots, f(X_a) \right]^T - \left[p_{i,1}, p_{i,2}, \ldots, p_{i,a} \right]^T \right\|_2 \qquad (5)$$

is minimal. ■

We use an Euclidean norm in (5) for three reasons:

1. it renders (5) smooth;
2. it causes a good centered spread of the sampling data around the model; and
3. it allows to solve the signomial modeling problem as the least–squares solution of an overdetermined system, which in turn reduces to solving a set of linear equations.

The posynomial fitting problem then reduces to:

Posynomial fitting problem:

Solve the signomial fitting problem with the extra constraint

$$c_i \geq 0, \quad \forall i \in [1:m] \qquad (6)$$

■

Whereas the signomial form has better fitting properties, the posynomial form allows to formulate analog circuit sizing as a geometric program [1, 2].

A (primal) geometric program is the constrained optimization problem:

$$\begin{array}{rrcll} \text{minimize} & f_0(X) & & & \\ \text{with the constraints:} & f_i(X) & \leq & 1, & i = 1, \ldots, p \\ & g_j(X) & = & 1, & j = 1, \ldots, q \\ & x_k & \geq & 0, & k = 1, \ldots, n \end{array} \quad (7)$$

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 f_0 and every constraint f_i, g_j, it can readily be seen that the transformed problem is a convex optimization problem. Because of this, it has only one global optimum. In addition, this optimum can be found very efficiently even for large problems, using interior point methods [4].

In view of the canonical geometric programming formulation, all performance constraints must be modeled in the normal form of (7). Therefore, we will scale the performance variables p_i in one of the two following ways:

- Linear scaling:

$$p_{i,scaled} = 1 \pm \frac{1}{W}\left(p_i - p_{i,spec}\right) \quad (8)$$

  with W an arbitrary weight factor.

- Logarithmic scaling:

$$p_{i,scaled} = 1 \pm \frac{1}{W}\log_{10}\left(\frac{p_i}{p_{i,spec}}\right) \quad (9)$$

  with W an arbitrary weight factor.

The scaling formulae (8) and (9) suggest a dependence of the models on the design specifications. This is only partly true. Indeed, the direction of the inequality may force us to fit $-p_i$ instead of the original performance parameter p_i.[1] The reason for this is obvious: changing the sign of a posynomial function does not preserve its posynomiality. However, the exact value of the specification (e.g. $GBW \geq$ 1GHz) is not an intrinsic part of the fitted model as the normalization does not harm the posynomiality of the models.

3. POSYNOMIAL MODEL GENERATION

In this section we will describe two techniques to fit posynomial models $p_i = f(X)$ (see (4)) with given constant exponents α_{ij} to a set of sample data $(X_k, p_{i,k})$:

1. the *indirect–fitting method*, as it is presented in [5], and
2. a *direct–fitting method*, the subject of this paper.

Both methods try to generate a posynomial response surface model [6] based on the generated performance samples. In section 4, the two methods will be compared using experimental results.

As the performance generation part of the two techniques is identical (see Fig. 1), we will briefly treat it in advance.

Fig. 3 illustrates how the set of performance samples is composed. The input samples X_k are generated using techniques from *Design of Experiments* in order to optimize the effectiveness of the sample set in the fitting process [7]. This input vector controls the values of (operating point) device currents and voltages. The *Operating Point Driven Solver* transforms these values into the corresponding transistor geometries and bias currents and voltages. Together with the analysis cards needed for the performance extraction, this results in a fully specified SPICE netlist. The consecutive

[1]This explains the ± sign in (8) and (9)

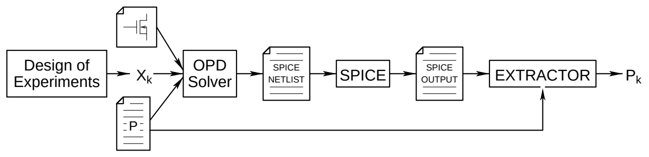

Figure 3: Performance generation outline

SPICE simulation results in an output file containing the simulation data. Afterwards, an automatically generated extractor script can distill the performance data. This concludes the part common to both methods. Note also that all these required simulations can be executed in parallel on a network of computers.

3.1 Indirect–fitting method

The indirect–fitting method is based on the fact that the signomial fitting problem reduces to solving an overdetermined set of linear equations in the least–squares sense when using an Euclidean norm in (5). The outline of the indirect fitting method is depicted in Fig. 4.

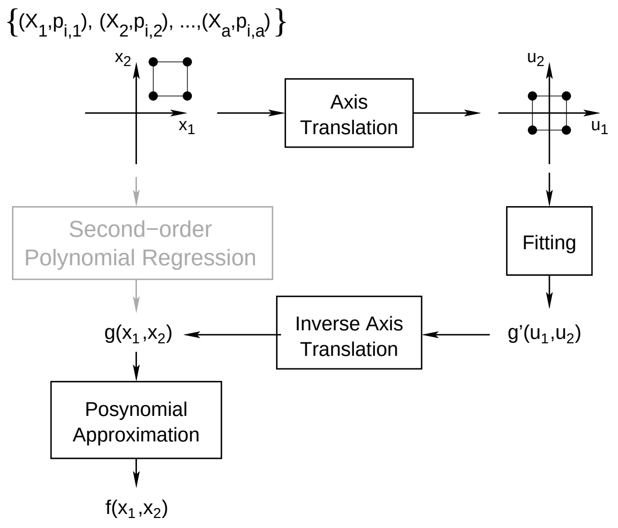

Figure 4: Outline of the indirect fitting method

The data set $\{(X_1, p_{i,1}), (X_2, p_{i,2}), \ldots, (X_a, p_{i,a})\}$ is first transformed into a data set that is located symmetrically around the origin of the U plane. Afterwards a second–order polynomial is fit to these data, such as to minimize the least squares error in the sampling points. Finally, the second–order polynomial is approximated by a posynomial form, i.e. the resulting model. The nature of the posynomial approximation step is to minimize the (nominal and first derivative) error in the center of the fitting hypercube. The details of this approximation can be found in [5].

Notice that the axis transformation was introduced to avoid the unstable solution of the least–squares fitting method that occurs in the case of asymmetrical data. Because of the axis translation, the indirect–fitting method can only generate models with integer exponents.

3.2 Direct–fitting method

The direct–fitting method solves the *posynomial fitting problem* directly. For this purpose, we developed a constrained minimization algorithm. The poysnomial fitting problem can be rewritten as a single–objective convex optimization problem:

$$\text{minimize } \psi(C) = \sum_{t=1}^{a} \left(\sum_{i=1}^{m} \left(c_i \prod_{j=1}^{n} \left(x_{j,t}^{\alpha_{ij}} \right) \right) - p_{i,t} \right)^2 \tag{10}$$

$$\text{subject to } c_i \geq 0, \quad \forall i \in [1 : m] \tag{11}$$

with $x_{j,t}$ and $p_{i,t}$ respectively the value of x_j and the value of p_i at the t^{th} experiment. The goal function (10) is a positive semi–definite second–order polynomial restricted to a convex constraint set. Therefore, the optimization problem is convex.

The entire fitting algorithm can be found in Fig. 5. This algorithm makes use of a boolean variable, *isposy*, a counter, *relcntr*, and a real variable, ψ_{prev}. It also assumes that c_1 is the coefficient of the model's constant term.

```
1.  compose $\psi(C)$
2.  set $\psi_{prev} = +\infty$
3.  do
    3.1.  do
          3.1.1.  minimize $\psi(C)$ using a conjugate–gradient descent until $C$ contains a negative component or until a local stop criterion has been fulfilled
          3.1.2.  set $isposy = true$
          3.1.3.  for $j = [2 : m]$, do
                  3.1.3.1.  if $(c_j < 0)$, then
                            3.1.3.1.1.  $c_j = 0$
                            3.1.3.1.2.  set $isposy = false$
          3.1.4.  set $relcntr = 0$
          3.1.5.  for $j = [2 : m]$, do
                  3.1.5.1.  if $((c_j == 0) \wedge (\nabla_{C,j}(\Psi) < 0))$, then
                            3.1.5.1.1.  release $c_j$
                            3.1.5.1.2.  $relcntr = relcntr + 1$
                  3.1.5.2.  else
                            3.1.5.2.1.  fix $c_j$
          until $isposy$
    3.2.  if $(relcntr == 0)$, then
          3.2.1.  stop
    3.3.  if $((\psi(C) - \psi_{prev}) < \varepsilon)$, then
          3.3.1.  stop
    3.4.  else
          3.4.1.  set $\psi_{prev} = \psi(C)$
          3.4.2.  release $c_j$ that offers the largest axis–wise descent
    forever
```

Figure 5: The direct–fitting algorithm

This algorithm has proven to be better than any other algorithm (interior–point algorithms with both external and internal barrier functions) we tested.

Besides the improved fitting capabilities (that will be demonstrated in section 4), this algorithm exhibits two clear advantages over the indirect approach:

- it is capable of fitting model coefficients of posynomial models with *arbitrary* real exponents, not only integers.
- the models it generates are sparse, i.e. a lot of the coefficients of the composing monomials are zero. This is advantageous for the needed CPU time when using these models in sizing applications afterwards.

3.3 Model quality assessment

In order to assess the fit–quality of the generated models, we use the quality–of–fit parameter q, defined in [5]. This allows a fair comparison between the indirect–fitting method and the direct–fitting method.

The starting point for this parameter is the root mean square of the deviation in the a sampling points. This parameter is then normalized by division with the performance range of the sampling points:

$$q = \frac{\sqrt{\sum_{k=1}^{a} \left(f(X_k) - p_k \right)}}{a \left[c + \left(\max_{k=1}^{a} p_k - \min_{k=1}^{a} p_k \right) \right]}. \tag{12}$$

In (12), c is a constant parameter to avoid error overestimation when the performance range approaches zero. If we reuse the sampling points used during the fitting process, then this figure is:

- computationally cheap (no extra simulations are needed), and
- easy to assess: a quality larger than 1 suggests a bad fit.

However, using verification points located within the fitting hypercube (not coinciding with the sampling points) yields a more reliable verification yardstick. In addition, the use of orthogonal arrays [8] (the sampling scheme that is used in [5]) locates the samples at the fitting hypercube's boundaries. This inherently disfavors the indirect–fitting approach.

We will therefore use the same three relative quality figures as proposed in [5]:

- q_{oc}: the relative model deviation in the center point,
- q_{tc}: the quality figure of (12) evaluated in sampling points located in the interior of the fitting hypercube, and
- q_{wc}: the quality figure of (12) evaluated in the original sampling points used for the model generation

The obvious drawback of using q_{tc} is the need for extra analyses (i.e. circuit simulations). Notice that for the generated posynomial models q_{wc} can be considered as the worst–case value (therefore the subscript wc), q_{tc} as the typical–case value and (especially for the indirect–fitting approach) q_{oc} as the optimal–case value.

4. EXPERIMENTAL RESULTS

As test case we use the same test circuit as used in [5], a high–speed CMOS OTA in a $0.7\mu m$ CMOS technology (see Fig. 6). 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 can be chosen for the high–speed OTA of Fig. 6. The chosen design variables are gathered in Table 1. 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 [9]. The bounded range of variables $v_i \in [lb_i, ub]$ is logarithmically scaled onto $x_i \in [0, 1]$ using

$$x_i = \log\left(\frac{v_i}{lb}\right) \Big/ \log\left(\frac{ub}{lb}\right) \tag{13}$$

As a consequence all scaled variables are positive (as required for the posynomial formulation). The bounds have also been indicated in Table 1.

Our goal is to derive expressions for the low–frequency gain ($A_{v,LF}$), the unity–gain frequency (f_u), the phase margin (PM), the input–referred offset voltage (v_{offset}) and the positive and negative slew rate (SR_p, SR_n). In order to comply with the geometric programming formulation (which in its direct form only supports minimization and $\leq$ constraints), we will fit the inverse of the characteristics that need to be maximized or have a $\geq$ constraint (i.e. $-A_{v,LF}$, $-f_u$, $-PM$ and $-SR_p$). All characteristics are scaled lin-

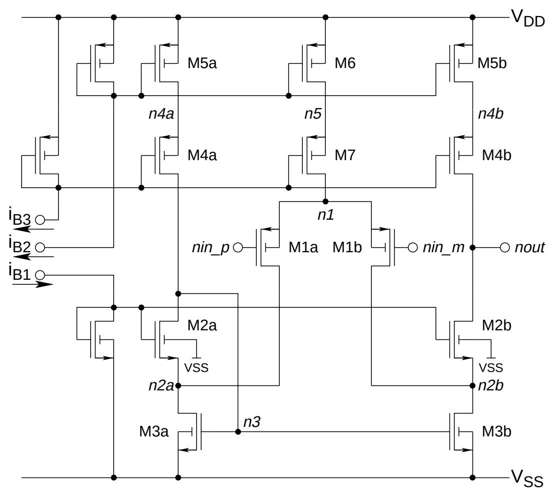

Figure 6: Schematic of a high–speed CMOS OTA

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 1: Design variables chosen as model inputs

early according to (8), except for f_u which is scaled logarithmically according to (9).

For each of the characteristics to model, we will derive posynomial expressions using two different sampling hypercube widths ($dx = 0.1, dx = 0.01$).

The direct–fitting algorithm has been implemented in our posynomial model generation prototype called PRÎSM. Its TCP–based client-server simulation system schedules simulation and extraction jobs on remote workstations over the intra–net or the Internet (*resource parallelization*) and then fits the posynomial models on a single workstation. The core modules have been coded using C++, while the TCP–based client–server system has been coded in Perl. The total amounts to about 40 000 lines of code.

PRÎSM was run on an Intel Celeron 466MHz running GNU/Linux. The analysis servers ran on 16 UNIX workstations (attached to a standard 100Mbit/s TCP–IP network and under normal load circumstances), 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. The simulations needed to obtain a full orthogonal hypercube of sampling points took approximately 3 minutes. Using these data the whole set of performance characteristics ($-A_{v,LF}$, $-f_u$, $-PM$, v_{offset}, $-SR_p$, SR_n) can be fitted. The time needed to fit each of the models using the algorithm of Fig. 5 is indicated in Table 2.

	generation time [s]		no. of coefficients		sparseness	
model	*dx*		*dx*		*dx*	
	0.1	0.01	0.1	0.01	0.1	0.01
$A_{v,LF}$	61	115	14	12	95.0%	95.8%
f_u	63	85	14	9	95.0%	96.9%
PM	125	130	22	20	92.3%	93.0%
v_{offset}	230	149	31	20	89.2%	93.0%
SR_p	116	116	32	12	88.9%	95.8%
SR_n	54	131	12	13	95.8%	95.5%

Table 2: Properties of the models generated using the direct–fitting technique

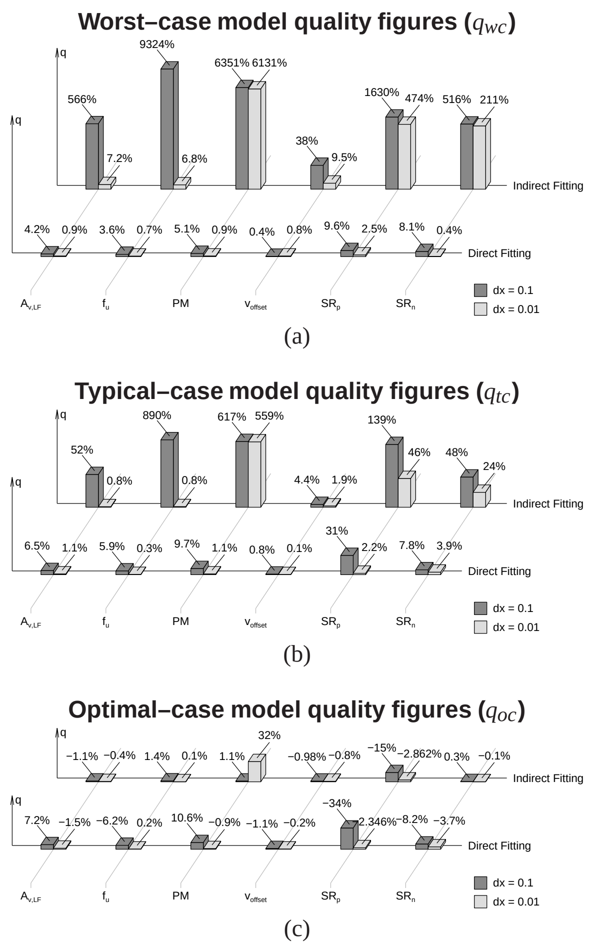

Figure 7: Model quality figures of the posynomial models: (a) worst case, (b) typical case, (c) optimal case

Table 2 demonstrates also clearly the sparseness (number of coefficients that are zero divided by the total number of template coefficients) of the models that are generated. Compare this to an average number of nonzero coefficients of 105 (corresponding to a model sparseness of 63.7%) for the models generated with the indirect–fitting technique. It can be concluded that concerning sparseness the indirect–fitting technique is outperformed by the new direct approach.

The quality of the generated models can be inspected in Fig. 7. The three subfigures, corresponding to the different quality measures q_{wc}, q_{tc} and q_{oc}, each contain two rows of data: the front row for the results of the direct–fitting technique and the back row for the results of the indirect–fitting technique. The improvement of the model qualities introduced by the direct–fitting technique is most considerable. On average an improvement with a factor of over 100 is observed, both for large as for small hypercubes.

5. CONCLUSIONS

In this paper, we presented a new direct–fitting method to fit posynomial model templates with arbitrary real exponents to numerical simulation data. The fit qualities obtained are very good, both for large as well as for small hypercube widths. In addition, the resulting models are very sparse. This method therefore effectively reduces the time and effort needed to setup and solve an analog circuit sizing problem in the form of a geometric program.

6. REFERENCES

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

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

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

[4] 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.

[5] W. Daems, G. Gielen, and W. Sansen, "Simulation–based automatic generation of signomial and posynomial performance models for analog integrated circuits," in *Proceedings IEEE/ACM International Conference on Computer Aided Design*, Nov. 2001, pp. 70–74.

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

[7] 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.

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

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