A Fitting Approach to Generate Symbolic Expressions for Linear and Nonlinear Analog Circuit Performance Characteristics

Walter Daems, Georges Gielen, Willy Sansen
Katholieke Universiteit Leuven, Department of Electrical Engineering, ESAT-MICAS
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium

Abstract

This paper presents a novel method to automatically generate symbolic expressions for both linear and nonlinear circuit characteristics using a template-based fitting of numerical, simulated data. The aim of the method is to generate convex, interpretable expressions. The posynomiality of the generated expressions enables the use of efficient geometric programming techniques when using these expressions for circuit sizing and optimization. Attention is paid to estimating the relative 'goodness-of-fit' of the generated expressions. Experimental results illustrate the capabilities of the approach.

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 business.
The sizing process can be decomposed into two subtasks [1]:
• one "investigates" the circuit's behavior (knowledge acquisition), and
• one adjusts the design parameters according to the obtained knowledge (knowledge use)
Symbolic analysis tries to automate the first subtask. Therefore, it is an important research topic in the analog electronic design area. While symbolic analysis research started much earlier, the first research boom is located in the 1980s [2], [3], [4], [5], [6], [7] mainly due to the wider availability of computing equipment. Soon it became obvious that the size of the circuits that could be analyzed was severely restricted by the exponential relationship between the the length of the generated expressions and the size of the circuit. Therefore, the research focus restricted itself more and more towards enabling the analysis of larger linear circuits by employing all sorts of approximation techniques [8], [9], [10], [11], [12], [13], [14]. The developed approximation techniques are based on exploiting the numerical differences in order of magnitude of the small-signal parameters. In most techniques reported in literature these differences are obtained by numerically analyzing the circuit in a single design point. Another line of research investigated hierarchical techniques, i.e. splitting the symbolic analysis task up in several smaller tasks [15], [16], [17]. Linear hierarchical techniques typically generate a sequence of expressions relating a network function with the small-signal parameters involved and the Laplace variable s. Flat techniques generate network functions like

$$H(s) = \frac{\sum_{i=0}^{p} a_i s^i}{\sum_{j=0}^{p} b_j s^j}. \tag{1}$$

More recently, some promising extensions towards pole-zero analysis [18], [19], [20], [21], [22] and nonlinear analysis [23], [24] also emerged.

The traditional approach used in symbolic analysis for linear (or linearized) analog circuits has been depicted on the left-hand side of Fig. 1. The approach starts with an AC expansion of the netlist,

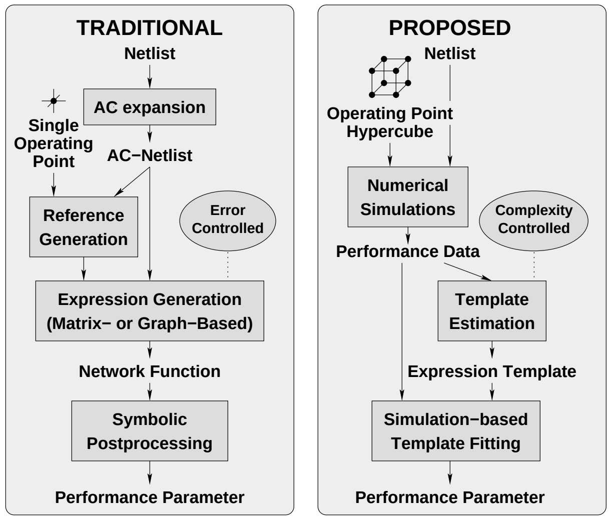

Figure 1. Comparison of the traditional symbolic analysis technique with the proposed analysis technique

followed by an expression generation step. This step first tries to simplify the netlist, then generates a simplified network function containing only the dominant terms. Finally symbolic postprocessing techniques are used to extract the performance characteristics from the network function.
However, so far, a number of problems remain unsolved:

1. The mainstream symbolic analysis techniques are limited to linear (or linearized) analog circuits with some extensions to weakly nonlinear circuits.
2. The resulting expressions generated by the current linear and nonlinear techniques are lengthy and their complexity is difficult to control, sometimes making their usefulness for interactive use questionable.
3. The linear expressions that are generated are formulated in terms of (correlated) small-signal parameters (g_m, c_{gs}, $r_o, \ldots$) instead of in terms of the true uncorrelated design parameters (V_{bias}, I_{bias}, w, l, $\ldots$). This makes an extra substitution necessary to reveal the true dependence on the design parameters. In addition, symbolic postprocessing steps are needed to extract the performance characteristics (e.g. phase margin, pass-band ripple) from the generated network functions. E.g., for the gain-bandwidth (GBW) of a circuit, this leads to the following substitution sequence:

$$\begin{aligned} g_m, c_{gs}, r_o, \ldots &= f(V_{bias}, I_{bias}, w, l, R, L, C) && (2)\\ H(s) &= g\left(s, g_m, c_{gs}, r_o, \ldots\right) && (3)\\ GBW &= h\left(H(s)\right) && (4)\\ &= h\left(g\left(f\left(V_{bias}, I_{bias}, w, l, R, L, C\right)\right)\right) && (5) \end{aligned}$$

4. Simplified transistor models are needed, in order to limit the complexity of the results.
5. The simplification (or simplified generation) of the expressions is only verified in a single design point, and therefore uncertainty arises concerning the reliability of the generated expressions.
6. The form of the expressions is not tuned towards the use in an optimization engine used for automatic sizing.

This paper describes an approach that attempts to overcome these problems. The idea behind the proposed method can be seen on the right-hand side of Fig. 1. The core of the method consists of directly fitting well-chosen expression templates to the numerically simulated performance characteristics of the circuit. Its strong points are:

1. Expressions for linear *and* nonlinear characteristics can be generated.
2. The complexity of the resulting expression is limited and imposed by the complexity of the fitting template. The designer is informed about the accuracy of the fit.
3. Performance parameters are expressed directly in terms of the design parameters, avoiding additional postprocessing and substitutions. E.g., for the Gain-Bandwidth (GBW) of a circuit, this corresponds to the following single expression:
$$GBW = u(V_{bias}, I_{bias}, w, l, R, L, C) \tag{6}$$
4. Well established and accepted device models, such as BSIM-3v3 or MM9 (used and trusted by designers today), can be used without impairing the compactness of the resulting expression.
5. The fitting samples are chosen so that the part of the design space in which the designer is interested is fully covered. Therefore the expressions are suited to be used over a larger part of the design space during circuit design.
6. The technique generates posynomial expressions [25]. This allows to formulate the sizing problem as a geometric program [26], [27].

The paper is organized as follows. Section 2 will provide some theoretical background. In section 3, we will discuss our approach in detail. The proposed approach has been implemented in a software prototype. The experimental results obtained with this prototype are described in section 4. Finally, section 5 summarizes both strong and the weak points of the presented approach. Some ideas for future research also have been indicated.

2 Posynomial performance parameter expressions

2.1 Performance parameters

Consider a system S transforming an input signal E into an output signal Y (Fig. 2-(a)). The mathematical modeling of this input-output relationship is called *behavioral modeling*. Fig. 2-(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*. Because our goal is to compose symbolical expressions that can be used for designing a circuit, we will concentrate on the latter.

Figure 2. Electronic system seen (a) as a system that relates an input signal $\mathbf{E}$ to an output signal $\mathbf{Y}$ and (b) as a system that relates a set of design parameters $\mathbf{X}$ the system's performance $\mathbf{P}$

2.2 Posynomial expressions

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
$$f(X) = \sum_{i=1}^{m} \left(c_i \prod_{j=1}^{n} \left(x_j^{\alpha_{ij}}\right) \right) \tag{7}$$
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*. Whereas the former has better fitting properties, the latter allows to formulate analog circuit sizing as a geometric program [26], [27].
A (primal) geometric program is the constrained optimization problem:
$$\begin{array}{rrcll} \text{minimize } & f_0(X) & & & \\ \text{with the constraints: } & f_i(X) & \leq & 1, & i = 1, \ldots, p \\ & g_j(X) & = & 1, & j = 1, \ldots, q \\ & x_k & \geq & 0, & k = 1, \ldots, n \end{array} \tag{8}$$
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 using interior point methods [28], even for large problems. Notice that strictly speaking posynomial expressions are not convex: the above mentioned transformation makes them convex. However, in view of that property, we conveniently denote them as convex expressions.

3 Symbolic Performance expression generation

Fig. 3 shows the outline of the proposed method to generate symbolic performance expressions. First we will determine a set of experiments to perform, in order to obtain an optimal dataset for the expression template estimator. In a second step we will perform these experiments using a numerical circuit-level simulator such as SPICE. The obtained samples are then used to estimate a good posynomial template and afterwards reused to generate the posynomial expression. In a final step, the model quality is verified. In the next subsections we will treat the different steps of our method in detail.

3.1 Posynomial expression generation

Whereas traditional symbolic analysis methods compose their expressions using network analysis intermixed with numerical approximation techniques, our method is based on straight mathematical fitting of a pre-estimated expression template using simulated data. Standard mathematical fitting techniques like interpolation or least-squares regression are well kown, but are rarely used in symbolic analysis methods.
However, the advantages of using simulation-based fitting techniques over traditional symbolic analysis techniques are multiple:

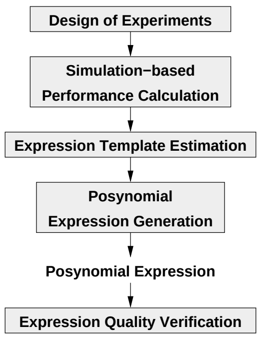
Design of Experiments
Simulation−based Performance Calculation
Expression Template Estimation
Posynomial Expression Generation
Posynomial Expression
Expression Quality Verification

Figure 3. Outline of the expression-generation method

- any performance parameter (linear as well as nonlinear) can be fitted,
- full accuracy SPICE transistor models can be used,
- performance parameters (such as GBW or slew rate) are directly fitted as a function of the design parameters, and
- the complexity of the resulting expression is directly imposed by the fitting template that is chosen.

Of course, the selection of the expression template is crucial, in order to obtain a good fit. A trade-off between the number of fit parameters allowed and the resulting fit accuracy has to be made. The designer's experience and insight in the system may clearly help in selecting (or refining) a model template. However, to assist the designer in this process, we propose the use of an *expression template estimator* that allows to compose a template without relying on human input.

Consider the generic n-dimensional posynomial template

$$f(X) = \sum_{k=-1,0,1} \sum_{l=-1,0,1} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \left(c_{i,k,j,l} x_i^k x_j^l \right) \right) \tag{9}$$

with n the number of design parameters and all $c_{i,k,j,l}$ positive. The template of (9) allows to obtain good fitting results. However, the number of independent parameters to fit equals

$$\#c_{i,j,indep} = \frac{3}{2}n^2 + \frac{5}{2}n + 1 \tag{10}$$

This number of fit parameters is too large for realistic sizing problems (e.g., $n > 10$ leads to $\#c_{i,j,indep} > 176$). The *expression template estimator* described in the next subsection allows to reduce this number.

3.2 Expression template estimation

To estimate which coefficients $c_{i,k,j,l}$ need to be retained in the posynomial fitting, we first fit the signomial function

$$g(X) = d_0 + \sum_{i=1}^{n} (d_i x_i) + \sum_{i=1}^{n} (d_{i,i} x_i^2) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} (d_{i,j} x_i x_j) \tag{11}$$

to the experimental data. The number of independent parameters to fit amounts to

$$\#d_{i,j,indep} = \frac{1}{2}n^2 + \frac{3}{2}n + 1, \tag{12}$$

which is about a third lower than (10). Because the fitting process itself is $O(d_{i,j,indep}^3)$, this is a considerable gain.

Then, a template estimation algorithm is applied to relate the obtained coefficients $d_{i,j}$ to the $c_{i,k,j,l}$ that need to be present in (9).

```
1.  start with an empty template
2.  add $c_{i,0,j,0}$
3.  if $d_i > 0$, then add $c_{i,1,j,0} x_i^1$
    else if $d_i < 0$, then add $c_{i,-1,j,0} x_i^{-1}$
4.  if $d_{i,i} > 0$, then add $c_{i,1,i,1} x_i^2$
    else if $d_i < 0$, then add $c_{i,-1,j,0} x_i^{-1}$
5.  if $d_{i,j}|_{i \neq j} > 0$, then add $c_{i,1,j,1} x_i x_j$
    else if $d_{i,j}|_{i \neq j} < 0$, then,
      if ($a_0$), then
        if ($\Delta x_i / x_i > \Delta x_j / x_j$), then
          add $c_{i,-1,j,1} x_i^{-1} x_j$ and $c_{i,0,j,-1} x_j^{-1}$
        else
          add $c_{i,1,j,-1} x_i^1 x_j^{-1}$ and $c_{i,-1,j,0} x_i^{-1}$
      else
        if ($\Delta x_i / x_i < \Delta x_j / x_j$), then
          add $c_{i,-1,j,1} x_i^{-1} x_j$ and $c_{i,0,j,-1} x_j^{-1}$
        else
          add $c_{i,1,j,-1} x_i^1 x_j^{-1}$ and $c_{i,-1,j,0} x_i^{-1}$
6.  if ($a_1$), then add $c_{i,-1,j,0}$, $i = 1 \ldots n$
7.  if ($a_2$), then add $c_{i,1,j,0}$, $i = 1 \ldots n$
8.  if ($a_3$), then add $c_{i,2,j,0}$, $i = 1 \ldots n$
```

Figure 4. The template estimation algorithm

A set of template estimators is proposed based on a single algorithm that contains 4 binary choices. This leads to 16 alternatives. For convenience, we represent the 4 binary choices by a 4-bit word $(a_3a_2a_1a_0)$. The algorithm can be found in Fig. 4.

This algorithm keeps the terms of (11) that have positive coefficients and replaces the negative terms by appropriate approximations (see [29]). The number of terms of the estimated templates is at most $1/2n^2 + 5/2n + 1$. In order to keep the template as sparse as possible, the estimation correlation between the coefficients of (11) should be as close as possible to zero. Techniques from design of experiments can assist us to achieve this goal.

3.3 Design of experiments

The theory of *Design Of Experiments (DOE)* provides a mathematical basis to select an optimal sample set that allows an uncorrelated estimation of the fit parameters, given a fit template [30]. The number of sampling schemes described in literature is vast: starting from full- and fractional factorial design, over Placket-Burman and Taguchi schemes, to Latin hypercube and even random design. A sampling scheme that allows an uncorrelated estimation of the coefficients of (11) are level-3 orthogonal arrays of strength 3 [31].

This scheme places the sampling points on the edges of a fitting hypercube of size ΔX around a center point X^*. Usually the center point is provided as well.

This sampling scheme has been chosen, in order to keep the coefficient leakage in (11) as small as possible.

The same samples are reused to fit the estimated posynomial template. This avoids additional (costly) numerical simulations.

3.4 Fit quality verification

In order to assess the quality of fit, we use the quality-of-fit parameter q, defined in [29]. The starting point for this parameter is the root mean square of the deviation in the a sampling points. This parameter then is normalized by division with the performance range of the sampling points:

$$q = \frac{\sqrt{\sum_{j=1}^{a} (f(X_j) - p_j)}}{a \left[c + \left(\max_{j=1}^{a} p_j - \min_{j=1}^{a} p_j \right) \right]}. \tag{13}$$

In ([13]) c is a constant 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:
1. computationally cheap (no extra simulations are needed), and
2. easy to assess: a quality larger than 1 suggests a bad fit.
We will further denote the fit-quality parameter that reuses only the set of sampling points by q_0. In order to have a more realistic view on the fit quality, additional samples located in the interior of the fitting hypercube may be selected. We will label this fit-quality parameter q_1. A drawback of the latter is the need for extra circuit simulations.

4 Experimental results

4.1 S-PR$\overset{\blacktriangle}{\imath}$SM

The flow of Fig. [3] has been implemented in S-PR$\overset{\blacktriangle}{\imath}$SM, our Symbolic Posynomial Response Surface Modeling prototype. Fig. [5] shows the overall concept of S-PR$\overset{\blacktriangle}{\imath}$SM. Sixteen alternative template estimation schemes have been incorporated in the template estimation engine allowing to compare and to select the most effective ones. The core engines have been coded using C++, while the analysis clients and servers have been coded in Perl. The total amounts to about 42 000 lines of code. S-PR$\overset{\blacktriangle}{\imath}$SM's TCP-based client-server simulation system schedules simulation and extraction jobs on remote workstations over the intra-net or the Internet. 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 are provided.[1]

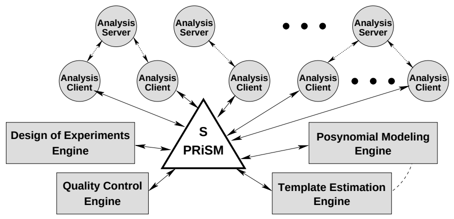

Figure 5. Overall concept of S-PR$\overset{\blacktriangle}{\imath}$SM

4.2 Experiments

As test case we use the circuit of Fig. [6], a high-speed CMOS OTA in a 0.7μm CMOS technology. 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.
Our goal is to derive expressions for the low frequency gain ($A_{v,LF}$), the unity frequency (f_u), the phase margin (PM), 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), we will fit the inverse of the characteristics that need to be maximized (i.e. $-A_{v,LF}$, $-f_u$, $-PM$ and $-SR_p$). Thirteen independent design variables can be chosen for the high-speed OTA of Fig. [6]. In Table [I] an overview of the chosen variables and their nominal values around wich expressions are to be generated is given. The variables are normalized by division with a reference value, indicated in the rightmost column of the overview. As a consequence all scaled variables are positive (as required for

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

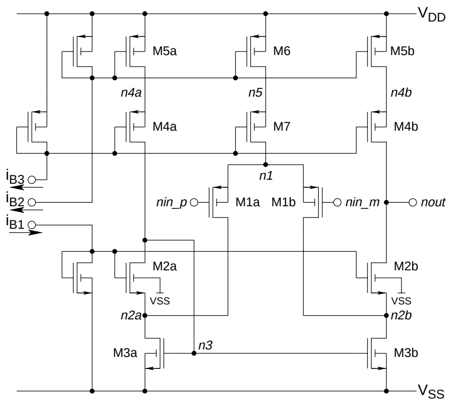

Figure 6. Schematic of a high-speed CMOS OTA

the posynomial formulation). Note that transistor currents and voltages are chosen as variables, rather than transistor widths, since we use an operating-point driven formulation for analog circuit sizing [32].

i	x_i	nominal	$\Delta x_{i,1}$	$\Delta x_{i,2}$	$\Delta x_{i,3}$	$x_{i,ref}$
1	$v_{GS,M1}$	-0.9V	± 0.2V	± 0.1V	±40mV	-4.0V
2	$v_{GS,M2}$	0.8V	± 0.2V	± 0.1V	±40mV	4.0V
3	$v_{DS,M2}$	0.5V	± 0.2V	± 0.1V	±40mV	4.0V
4	$v_{GS,M3}$	2.5V	± 0.2V	± 0.1V	±40mV	4.0V
5	$v_{GS,M4}$	-1.2V	± 0.2V	± 0.1V	±40mV	-4.0V
6	$v_{GS,M5}$	-1.2V	± 0.2V	± 0.1V	±40mV	-4.0V
7	$v_{DS,M5}$	-1.4V	± 0.2V	± 0.1V	±40mV	-4.0V
8	$v_{DS,M6}$	-0.85V	± 0.2V	± 0.1V	±40mV	-4.0V
9	$i_{D,M1}$	1mA	± 0.5mA	± 0.25mA	±10mA	10mA
10	$i_{D,M2}$	0.8mA	± 0.5mA	± 0.25mA	±10mA	10mA
11	i_{B1}	11uA	± 5uA	± 2.5uA	±10mA	100uA
12	i_{B2}	13uA	± 5uA	± 2.5uA	± 1uA	100uA
13	i_{B3}	12uA	± 5uA	± 2.5uA	± 1uA	100uA

Table I. Design variables chosen as model inputs

For each of the characteristics to model, we will derive posynomial expressions using three different sampling hybercube widths (ΔX_1, ΔX_2, ΔX_3). Their respective values can also be found in Table [I].
S-PR$\overset{\blacktriangle}{\imath}$SM was run on an Intel Celeron 466MHz running GNU/Linux. The analysis servers ran on 16 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. 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, -SR_p, SR_n) can be fitted.
Comparison of the template estimators
We gathered the fit-quality parameters that can be obtained by each of the template estimation variants in Table [II]. Hereto, we divided the quality range (difference between best and worst fit quality) for each parameter into five subranges (each representing 20% of the range). The subranges have been labeled from 'best' (++), over good (+), average (o), and poor (-), to bad (--). Table [II] clearly suggests the parallel use of multiple template estimators in order to obtain an acceptable fit in all cases. The template variants that we have chosen are 0xE and 0xF (grey rows in table [II]).
Fig. [7] shows the trade-off between the generation time (i.e. a direct function of the template size) and the obtained accuracy (fit-quality) for the posymial fitting of an expression for SR_n. In order not to overload the picture, we omitted the fits based on template estimators 0x2 to 0xD. It can readily be seen that the speed advan-

nr.	$A_{v,LF}$			f_u			PM			SR_p			SR_n		
	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III
0x0	++	o	++	++	++	+	-	o	++	--	+	++	--	--	++
0x1	--	--	--	--	-	o	--	++	--	++	--	++	--	++	++
0x2	++	o	++	++	++	+	-	o	++	--	+	++	--	--	++
0x3	--	--	--	--	o	--	--	++	--	--	++	--	++	++	--
0x4	o	+	++	++	+	++	-	--	++	+	+	++	++	-	++
0x5	--	o	++	--	-	++	--	o	--	++	--	++	--	++	++
0x6	o	+	++	++	+	++	-	--	++	+	+	++	++	-	++
0x7	--	o	++	++	-	--	--	o	--	++	++	++	++	++	--
0x8	++	++	++	++	++	--	++	-	++	--	++	++	--	--	++
0x9	--	--	++	--	--	--	o	+	--	++	++	++	++	++	--
0xA	++	++	++	++	++	--	++	-	++	--	++	++	--	--	++
0xB	--	--	++	--	++	--	o	+	--	++	--	++	++	++	--
0xC	++	++	++	++	++	+	++	--	++	--	+	--	--	-	++
0xD	--	o	++	++	--	++	--	o	--	++	--	++	--	++	--
0xE	++	++	++	++	+	+	++	--	++	--	+	--	--	-	++
0xF	--	o	++	--	o	--	--	o	--	++	--	++	++	++	--

Table II. Template estimator performances (++ = best, + = good, o = average, - = poor, -- = bad) for ΔX_1 (I), ΔX_2 (II), ΔX_3 (III)

tage of using a template estimator over using the full expression is paid by a small loss of accuracy. Also notice the low fitting time of the second-order polynomial template, which has a template complexity that is almost equal to the estimated templates. However, it is outperformed by the estimated templates when the fit quality is taken into concern.

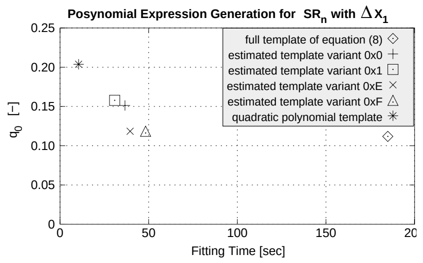

Figure 7. Trade-off between fitting time and obtained accuracy when generating a posynomial expression for SR_n using ΔX_1

Resulting models

The obtained number of terms in the resulting expressions, the fitting time and the fit-quality parameters can be found in Table III. We can clearly observe that the fit quality improves with smaller hypercube widths ($\Delta X_1 > \Delta X_2 > \Delta X_3$) and that the expressions based on the estimated templates are a good trade-off between generation time and accuracy. Also take note of the small number of terms needed for the expressions. The fact that the fit quality of the expression based on the full template is sometimes outperformed by the expressions based on the estimated template indicates that the fit routine itself can be improved.

	ΔX_1				ΔX_2				ΔX_3			
param	terms [#]	time [sec]	q_0 [-]	q_1 [-]	terms [#]	time [sec]	q_0 [-]	q_1 [-]	terms [#]	time [sec]	q_0 [-]	q_1 [-]
	Full template of eqn. (9)											
-$A_{v,LF}$	21	227.4	0.026	0.039	24	257.3	0.021	0.030	10	226.9	0.017	0.017
-f_u	25	210,7	0.115	0.177	23	250.4	0.096	0.126	14	242.9	0.054	0.062
-PM	21	194.1	0.081	0.080	19	244.1	0.039	0.049	18	261.6	0.024	0.023
-SR_p	19	202.4	0.125	0.164	25	202.9	0.095	0.127	14	193.7	0.059	0.076
SR_n	24	183.4	0.116	0.146	22	214.0	0.115	0.149	9	202.9	0.076	0.078
	Estimated template (0xE & 0xF)											
-$A_{v,LF}$	19	68.3	0.024	0.035	18	77.0	0.020	0.028	15	87.0	0.011	0.013
-f_u	18	68.5	0.127	0.176	11	70.0	0.118	0.152	13	79.4	0.091	0.087
-PM	17	66.7	0.088	0.087	21	78.4	0.046	0.058	23	94.0	0.019	0.019
-SR_p	19	70.6	0.123	0.137	14	73.0	0.144	0.186	15	81.0	0.050	0.058
-SR_n	28	68.3	0.121	0.148	13	69.3	0.116	0.131	15	78.6	0.068	0.084

Table III. Complexity, generation time and fit-quality parameters for the generated expressions

Consider the graphical representations (generated by S-PRISM) of the expressions for $-A_{v,LF}$, $-SR_p$ and SR_n in Fig. 8. Every column in the figure corresponds to one term. The bottom part of the figure identifies all x_i, x_j input parameter combinations involved, using a color code. The color code indicating the value of the exponents can be found in the input parameter legend. The colored bars appearing in the top part each represent the value of a coefficient. Coefficient values that normally would *not* be visible on the scale are indicated by bars below the axis. The relative dynamic range bargraphs for each term in the middle allow to easily locate the dominant design parameters occurring in a performance variable.

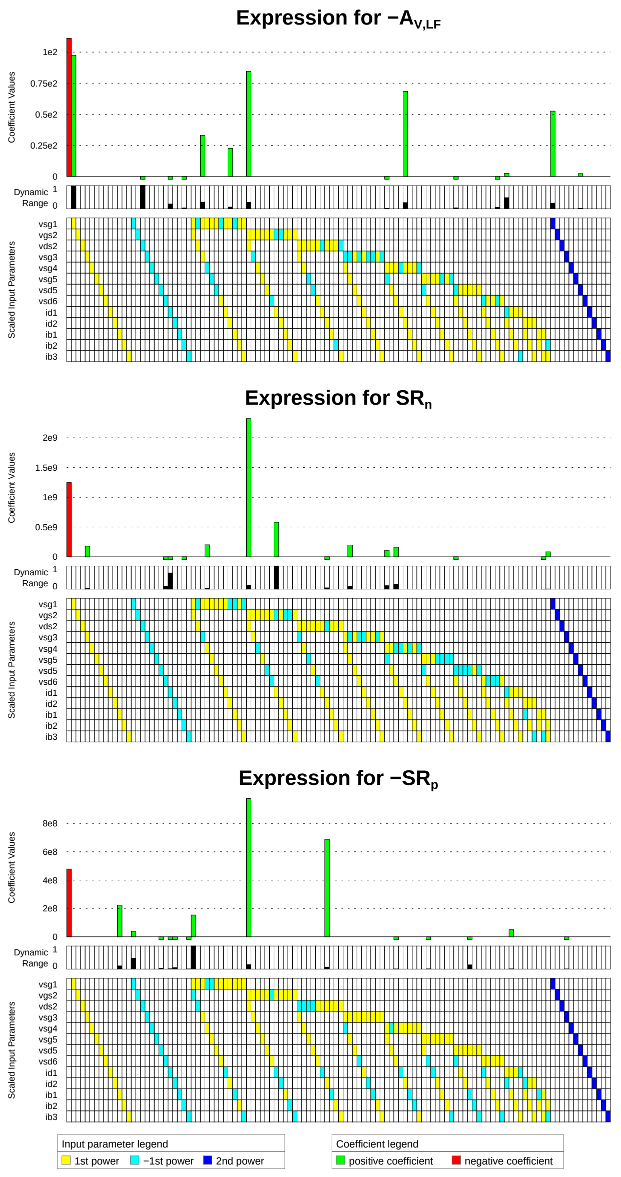

Figure 8. Graphical representation of the expression for $A_{v,LF}$, SR_p and SR_n fitted for ΔX_3

The dynamic ranges of the terms of $-A_{v,LF}$ show two dominant terms: one linear term in $-v_{GS,M1}$ and one term inversely proportional to $v_{DS,M2}$. Indeed, if $v_{GS,M1}$ drops, than g_{mM1} increases (for constant I_{D1}, and hence $A_{v,LF}$ increases. Likewise if

$v_{DS,M2}$ increases, then both g_{mM2} and r_{oM2} increase, thus $A_{v,LF}$ increases.

Examining its dominant terms, the interpretation of the expression for $-SR_n$ is also straightforward. The term inversely proportional to $i_{D,M1}$ reflects the fact that in negative slewing conditions all input stage bias current is mirrored via M3a to M3b, forming the driving force to discharge the load capacitor. The interaction term containing $1/v_{GS,M2}$ indicates the beneficial effect of M2b's low channel resistance. This effect is more important for higher values of $i_{D,M1}$. This explains the interaction effect.

Since $i_{D,M1} > i_{D,M2}$, we expect the expression for SR_p to show no dependence on $i_{D,M1}$ (as M2 goes in cut-off). The corresponding terms are indeed negligible, However the large dependence on $i_{D,M2}$ that one expects, cannot be found in the expression. Likewise, the dependence on $v_{GS,M1}$ and $v_{GS,M2}$ is totally unexpected. We did not (yet) find an explanation for this unobvious result.

Finally, it must also be mentioned that for larger fitting hypercubes — though their numerical accuracy is reasonable — it is much harder to recognize the expected effects in the expressions.

5 Conclusions

We presented a fitting-based posynomial expression generation technique based on numerical simulations. The first experimental results clearly show that the presented technique is able to generate short and reasonably accurate convex expressions for both linear and nonlinear performance characteristics. However, additional research is needed to further mature the technique. More general and more powerful template estimators are needed to accomodate for larger fitting hypercubes and the nature of the paradoxical expressions needs to be further investigated.

Besides presenting a new approach to symbolic analysis, we also hope that this paper may be an impulse towards a shift in focus of the research on symbolic analysis. Generating short and usable expressions should – in our view – be of more concern than extending the size limits of the transistor-level circuits that can be analyzed.

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. DAC*, pp. 298–303, June 1996.

[2] K. Singhal and J. Vlach, "Symbolic circuit analysis," *Proc. IEE*, vol. 128, pp. 81–86, Apr. 1981.

[3] J. Starzyk and A. Konczykowska, "Flowgraph analysis of large electronic networks," *IEEE Trans. on CAS*, vol. 33, pp. 302–315, Mar. 1986.

[4] A. Liberatore and S. Manetti, "SAPEC: A personal computer program for the symbolic analysis of electric circuits," in *Proc. IEEE ISCAS*, pp. 897–900, 1988.

[5] S. J. Seda, G. R. Degrauwe, and W. Fichtner, "A symbolic analysis tool for analog circuit design automation," in *Proc. IEEE/ACM ICCAD*, pp. 488–491, 1988.

[6] G. Gielen, H. Walscharts, and W. Sansen, "ISAAC: a symbolic simulator for analog integrated circuits," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 1587–1597, Dec. 1989.

[7] F. V. Fernández-Fernandéz and A. Rodríguez-Vázquez, "Interactive AC modeling and characterization of analog circuits via symbolic analysis," *Journal of Analog Integrated Circuits and Signal Processing*, vol. 1, pp. 183–208, Nov. 1991.

[8] R. Sommer, E. Hennig, G. Dröge, and E.-H. Horneber, "Equation-based symbolic approximation by matrix reduction with quantitative error prediction," *Alta Frequenza – Rivista Di Elettronica*, vol. 5, pp. 317–325, Dec. 1993.

[9] Q. Yu and C. Sechen, "Approximate symbolic analysis of large analog integrated circuits," in *Proc. IEEE/ACM ICCAD*, pp. 664–671, 1994.

[10] P. Wambacq, F. V. Fernández-Fernandéz, G. Gielen, W. Sansen, and A. Rodríguez-Vázquez, "Efficient symbolic computation of approximated small-signal characteristics," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 327–330, Mar. 1995.

[11] C.-J. R. Shi and X.-D. Tan, "Symbolic analysis of large analog circuits with determinant decision diagrams," in *Proc. IEEE/ACM ICCAD*, Nov. 1997.

[12] O. Guerra, J. D. Rodríguez-García, E. Roca, F. V. Fernández-Fernandéz, and A. Rodríguez-Vázquez, "A simplification before and during generation methodology for symbolic large-circuit analysis," in *Proc. ICECS*, vol. 3, pp. 81–84, Sept. 1998.

[13] W. Daems, G. Gielen, and W. Sansen, "Circuit complexity reduction for symbolic analysis of analog integrated circuits," in *Proc. DAC*, pp. 958–963, June 1999.

[14] W. Verhaegen and G. Gielen, "Efficient DDD–based symbolic analysis of large linear analog circuits," in *Proc. DAC*, pp. 139–144, June 2001.

[15] M. M. Hassoun and K. S. McCarville, "Symbolic analysis of large-scale networks using a hierarchical signal flowgraph approach," *Journal of Analog Integrated Circuits and Signal Processing*, no. 3, pp. 31–42, 1993.

[16] M. Pierzchała and B. Rodanski, "Symbolic analysis of large-scale networks by circuit reduction to a two-port," in *Proc. SMACD Workshop*, Oct. 1996.

[17] O. Guerra, J. D. Rodríguez-García, E. Roca, F. V. Fernández-Fernandéz, and A. Rodríguez-Vázquez, "True hierarchical symbolic analysis of large-scale analog integrated circuits.," in *Proc. SMACD Workshop*, pp. 164–168, Oct. 1998.

[18] G. Dröge and E.-H. Horneber, "Symbolic calculation of poles and zeros," in *Proc. SMACD Workshop*, 1996.

[19] G. Nebel, U. Kleine, and H. Pfleiderer, "Symbolic pole/zero calculation using SANTAFE," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 752–761, July 1995.

[20] J. J. Hsu and C. Sechen, "Accurate extraction of simplified symbolic pole/zero expressions for large analog ICs," in *Proc. IEEE ISCAS*, pp. 2083–2087, 1995.

[21] F. Constantinescu and M. Nitescu, "A new approach to symbolic pole computation," in *Proc. ECCTD*, pp. 665–658, 1995.

[22] J. D. Rodríguez-García, O. Guerra, F. V. Fernández-Fernandéz, and A. Rodríguez-Vázquez, "A symbolic pole/zero extraction methodoly based on analysis of circuit time-constants," in *Proc. DCIS*, pp. 327–332, 1999.

[23] P. Wambacq, G. Gielen, P. R. Kinget, and W. Sansen, "High-frequency distortion analysis of analog integrated circuits," *IEEE Trans. on CAS-II: Analog and Digital Signal Processing*, vol. 46, pp. 335–345, Mar. 1999.

[24] W. Verhaegen and G. Gielen, "Symbolic distortion analysis of analog integrated circuits," in *Proc. ECCTD*, Aug. 2001.

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

[26] 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, pp. 1–21, Jan. 2001.

[27] 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, pp. 22–38, Jan. 2001.

[28] 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.

[29] W. Daems, G. Gielen, and W. Sansen, "Simulation-based automatic generation of signomial and posynomial performance models for analog integrated circuits," in *Proc. IEEE/ACM ICCAD*, pp. xx–xx, Nov. 2001.

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

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

[32] 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 ICCAD*, pp. 304–307, Nov. 1998.