PolyGP: Improving GP-Based Analog Optimization through Accurate High-Order Monomials and Semidefinite Relaxation

Ye Wang, Constantine Caramanis, and Michael Orshansky

Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX, USA, 78712
{lhywang, constantine, orshansky}@utexas.edu

ABSTRACT

Geometric programming (GP) is popular for use in equation-based optimization of analog circuits thanks to GP-compatible analog performance functions, and its convexity, hence computational tractability. The main challenge in using GP, and thus a roadblock to wider use and adoption, is the mismatch between what GP can accurately fit, and the behavior of many common device/circuit functions.

In this paper, we leverage recent tools from sums-of-squares, moment optimization, and semidefinite optimization (SDP), in order to present a novel and powerful extension to address the monomial inaccuracy: fitting device models as higher-order monomials, defined as the exponential functions of polynomials in the logarithmic variables.

By the introduction of high-order monomials, the original GP problems become polynomial geometric programming (PolyGP) problems with non-linear and non-convex objective and constraints. Our PolyGP framework allows significant improvements in model accuracy when symbolic performance functions in terms of device models are present. Via SDP-relaxations inspired by polynomial optimization (POP), we can obtain efficient near-optimal global solutions to the resulting PolyGP.

Experimental results through established circuits show that compared to GP, we are able to reduce fitting error of device models to 3.5% from 10.5% on average. Hence, the fitting error of performance functions decrease from 12% of GP and 9% of POP, to 3% accordingly. This translates to the ability of identifying superior solution points and the dramatic decrease of constraint violation in contrast to both GP and POP.

1. INTRODUCTION

Analog design automation is promising as it can increase the productivity while reducing the design time. Although the ultimate goal of automated analog design is simultaneous optimal selection of circuit topology and the optimal sizing under that topology [12], currently most works assume fixed topology and focus on optimal design sizing in terms of area, performance and power. In the area of analog sizing, existing works are divided into two categories: simulation-based methods [12, 9] in which optimization is driven by extensive SPICE simulation on design points, and equation-based methods which build an analytical model for optimization [6, 2, 11]. Simulation-based methods enjoy high accuracy of point evaluations of feasibility, and local algorithms are able to guarantee convergence to local optimal solutions. The cost is that SPICE simulations may be time-consuming, and furthermore, the local nature of the algorithms typically captures no global structure of the problem, which might be used for faster convergence to globally optimal solutions. The promise of equation-driven convex methods is the potential to capture such global structure, and thus converge efficiently to globally optimal solutions. Yet due to the limited nature of the fitting functions (typically GP monomials – see more below) that convex optimization essentially imposes, the promise of this approach has remained largely unfulfilled. Helping to bridge this gap is the main focus of this paper. In this work, we focus on equation-based analog synthesis only.

Perhaps the most widely used and explored equation-based method is geometric programming (GP) [2, 6]. Here, analog circuit performance is approximated by *posynomial* functions of design variables. Optimization problems using posynomial constraints and objective function can be solved efficiently using convex optimization via fast interior point solution methods. More importantly, closed under multiplication and division, monomials are compatible with most analog performance functions, which involves fractional forms of transistor behavior models. As has been widely observed [17, 8], the first-principles models derived based on long-channel transistor theory [2] fail to provide sufficiently accurate fits for nanometer-scale technologies. An alternative is to use automatic monomial/posynomial model-fitting approaches [1, 5] to approximate the SPICE-generated pre-characterization data with monomial/posynomial functions. Yet because of the strong non-linearity of the analog performance and transistor behavioral models, GP monomials appear inadequate, as they are in general incapable of producing a reasonable fit over even small portions of the range.

Another synthesis framework considers using a richer class of functions to fit: higher order polynomials [11]. Polynomial optimization approaches have been explored in diverse areas, including control, combinatorial and continuous optimization, differential equations, and elsewhere [3, 10, 14, 16]. Multivariate polynomials can deliver better model fitting performance than monomials/posynomials as they allow more degrees of freedom and permit model non-linearity and non-convexity. While non-convex, fairly recent advances in algebraic geometry and semidefinite optimization [13, 10] have demonstrated how polynomial optimization can be well-approximated by semidefinite optimization, which is tractable, and moreover guarantees theoretical convergence to global optimal solutions.

While multivariate polynomials are in principle able to approximate arbitrary continuous functions, in practice this may require extremely high degrees. Consequently, the resulting semidefinite relaxations are massive, containing a number of variables *orders of magnitude* larger than the size of the problem. We have observed that this is particularly true in the circuit performance setting, where important analog performance metrics are fractional polynomials of transistor behavioral models. While fractional functions are continuous and in principle can be approximated by polynomial functions, extremely high degree polynomials are required, thus significantly limiting the

978-3-9815370-6-2/DATE16/

practical impact of such a direction. For instance, as we discuss in Section 4, LNA models involve fractional polynomial functions which require high-order relaxation with high computational complexity.

One of our key observation is that, in most cases, symbolic analog performance functions are still able to capture some global structure of the problem when written in terms of transistor behavioral models like g_m, g_{ds}, c_{gs} and so on, rather than monomial functions with design variables W, L and I. Often symbolic expressions of analog performance functions in terms of device models are available from first principles analysis, or from tools for constructing symbolic equations [4].

The key idea and motivation in this paper, is to develop a framework that can capture the best of both worlds of GP and POP: we want a function class as flexible and powerful as polynomial optimization, yet closed under fractional composition, just like GP posynomials. Indeed, polynomial models provide a promising solution to dealing with nonlinear curvature and non-convexity as we can increase the degree and thereby increase the fitting accuracy.

The major innovation and contribution in this work is in the formulation coupling GP with POP. By introducing *high-order monomial fitting* for transistor behavioral functions, we arrive at a *nonlinear and nonconvex* problem formulation with more accurate performance functions. We explain precisely this formulation in the sections that follow. We call this new optimization problem **PolyGP** for Polynomial Geometric Programming problems.

In cases with analog performance function in terms of device models, PolyGP is ideally suited for the problem at hand, since the transistor behavioral parameters can be fitted, and then combined via fractional functions, to match circuit performance. Unlike POP, the framework we introduce here can easily handle such fractional combinations.

As is well known, not all analog performance functions have accurate symbolic equations in the global range. For instance, the frequency of voltage controlled oscillator (VCO) [17] with a chain of inverters involves transient simulation in full-swing signal model, and hence is not appropriate for symbolic equations. In this setting, PolyGP, just like any other approach (GP, POP, etc.) must build the model using some form of fitting, typically regression. Our PolyGP offers a significant generalization over GP, because optimizing over high-order monomials is the same as polynomial optimization, which is more powerful than linear programming resulting from GP with monomials only. We demonstrate this in Section 3.2.

Therefore, if accurate symbolic equations are available, our approach provides a much better fit than POP, at significantly less cost; meanwhile, if no such symbolic equations are available, our approach essentially reduces to POP.

In order to solve this PolyGP problem, we leverage recent work on Reformulation-Linearization/Convexification-Technique (RLT) proposed by [10, 16]. The basic idea of our method is to convert the general nonconvex and nonlinear polynomial constraints into linear matrix inequality (LMI) constraints by introducing a semidefinite (SDP) relaxation to remove the general non-linear cross-product term in the formulation. This allows for converting the *non-convex* PolyGP problem into a sequence of *convex* relaxations, enabling efficient global solution by modern convex optimization solvers like CVX [7].

While our main motivation and focus is the area of analog synthesis, we believe that this is a general and rich class of functions, that has not previously been used, or even recognized as highly tractable. Thus, we believe that the work here might prove important more broadly than in problems in analog circuits. One way to think about our non-linear formulation, is as a direct polynomial lifting of GP – indeed, the first step in our series of relaxations coincides with GP.

Our simulation experiments on established benchmarks show that allowing higher-order monomial is critical for both transistor behavioral functions and analog performance functions. The fitting error decreases from 10.5% to 3.5% compared to the standard monomials in terms of g_m, g_{ds}, c_{gs} and so on, which results in dramatic decrease of fitting errors of analog performance metrics. PolyGP models with high-order monomials are also superior to general polynomials in fitting as structured information cannot be well utilized in the black-box treatment by polynomials. For instance, in the op-amp circuits, errors of PolyGP is bounded by 3% with that of quadratic polynomial exceeding 13%. Specifically, this improvement in accuracy translates to the ability of identifying better feasible solutions by achieving better objective values and increasing the success rate of constraint satisfaction.

2. BACKGROUND AND OVERVIEW: RELATED APPROACHES

In this section, we describe the geometric programming and polynomial optimization frameworks. We focus on the main techniques of how GP and POP are convexified, which inspires the formulation of PolyGP in Section 3 and enables efficient algorithms for PolyGP.

2.1 Geometric programming and its convex form

GP problems are a family of optimization problems where the objective and constraints are *polynomials*. A posynomial $f(x_1, \ldots, x_n)$ has the general form $\Sigma_{k=1}^{K} c_k x_1^{\alpha_{k1}} \cdots x_n^{\alpha_{kn}}$, where the α's are arbitrary real numbers and the c's are positive. Each product term is called a monomial. A key property of monomials is that they are closed under multiplication and, unlike polynomials, they are also closed under division. This is a key compositional property that has made the GP paradigm very useful for circuit optimization problems.

The generic geometric program has the form:

$$\begin{aligned}
\min_x : \quad & f_0(x) = \sum_{k=0}^{K_0} c_{0k} \prod_{j=0}^{n} x_j^{\alpha_{0kj}} \\
\text{s.t.} : \quad & f_i(x) = \sum_{k=0}^{K_i} c_{ik} \prod_{j=0}^{n} x_j^{\alpha_{ikj}} \leq 1, \quad i = 1, \ldots, m, \\
& g_i(x) = d_i \prod_{j=0}^{n} x_j^{\beta_{ij}} = 1, \quad i = 1, \ldots, p, \\
& x_i > 0, \quad i = 1, \ldots, n
\end{aligned}$$

in which $f_0, \ldots, f_m$ are posynomial functions with c_{ik} and α_{ikj} denoting the coefficient and exponents of monomial k of f_i respectively. All $g_1, \ldots, g_p$ are monomials functions with the coefficient d_i and exponents β_{ij}.

Geometric programs are not convex optimization problems in their natural forms as the Hessian matrix of posynomials is not always positive semidefinite (PSD). But after a change of variables and a transformation of the objective and constraints, they can be converted into convex forms.

Denote $y_i = \log(x_i)$, so that $x_i = e^{y_i}$. By taking this log transformation, the monomial of given x:

$$f(x) = c x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$$

reduces to

$$f(y) = e^{\alpha^T y + \log(c)}$$

in which α is a column vector composed of $\alpha_1, \alpha_2, \ldots, \alpha_n$.

Hence the posynomials as a function of x become positive sums of exponentials of linear functions of y.

By taking log transformation on monomials and posynomials of variable y, we arrive at the problem:

$$\begin{aligned}\min_y: \quad & \tilde{f}_0(y) = \log(\Sigma_{k=0}^{K_0} e^{\alpha_{0k}^T y + \log(c_{0k})}) \\ \text{s.t.}: \quad & \tilde{f}_i(y) = \log(\Sigma_{k=0}^{K_0} e^{\alpha_{0k}^T y + \log(c_{0k})}) \le 0, \quad i = 1, \ldots, m, \\ & \tilde{g}_i(y) = \beta_i^T y + \log(d_i) = 0, \quad i = 1, \ldots, p.\end{aligned}$$

Since $\tilde{f}$'s are in the form of log-sum-of-exp and $\tilde{g}$ are linear constraints, this problem is convex, thus is able to be solved efficiently by modern convex solvers. We refer to it as the convex form of GP.

2.2 Polynomial Optimization and Semidefinite Relaxations

Without loss of generality, a generic polynomial optimization problem can be written as follows.

$$\begin{aligned}\text{minimize}_x: \quad & f_0(x) \\ \text{subject to}: \quad & f_i(x) \ge 0, \quad i = 1, \ldots, p.\end{aligned}$$

Here, $x = (x_1, \ldots, x_n)$ is an n-dimensional vector, and the $\{f_i(x)\}$ are degree d_i polynomials, with $d_i \le d$ for all i. These relaxations are parameterized by an integer r, where $r \ge d/2$ (recall $d = \max_i d_i$). By relaxing the non-linear cross terms x^α to corresponding moments, we arrive at the (linear) semidefinite programming problems of the form:

$$\begin{aligned}\text{minimize}_{\{m_\alpha\}}: \quad & \sum_\alpha p_\alpha m_\alpha \\ \text{subject to}: \quad & M_r(m) \succeq 0, \\ & M_{r-r_i}(f_i m) \succeq 0.\end{aligned}$$

In the interest of space, we defer the details to an online appendix, and in the meantime refer the interested reader to [10]. As above, $m = \{m_\alpha\}$ denotes the moments over which we are optimizing. The moment constraints that $\{m_\alpha\}$ is a valid moment sequence are approximated by the semidefinite constraints $M_r(m) \succeq 0$ and $M_{r-r_i}(f_i m) \succeq 0$, where $M_r(m)$ and $M_{r-r_i}(f_i m)$ are *moment matrices*, standard in the literature. To give these definitions, we order multi indices using the graded reverse lexicographic ordering. Thus, for example, if we have 2 variables, then $\beta_1 = [0,0]$, $\beta_2 = [1,0]$, $\beta_3 = [0,1]$, $\beta_4 = [2,0]$, and so on. As described in [10], as the relaxation parameter, r, increases, the optimum of the (convex) SDP above is guaranteed to converge to the optimum of the original polynomial (and non convex) optimization problem.

3. POLYNOMIAL GEOMETRIC PROGRAMMING FRAMEWORK

While polynomials can fit any continuous function, this may require arbitrarily high degree. The degree, however, may not accurately reflect the complexity of the function being fitted. A key example of this for us, is fractional functions. Even simple fractional functions may require a very high polynomial in order to produce a good fit. Since the degree impacts the complexity of solution of the resulting POP, polynomials are not useful for all problems in practice – in particular, they are problematic for applications where fractional functions are common. GP on the other hand, excels in precisely this area, since combining monomials by addition, multiplication or division, again returns a posynomial. This section describes the proposed PolyGP framework of analog synthesis, which aims to combine the two key properties of POP and GP: the ability to use higher order polynomials to produce better fits (like POP) and using a class of functions closed under division, so as to be suitable for fractional problems (like GP).

We describe the formulation here, and then we show how semidefinite relaxations can be employed to obtain globally optimal solutions.

3.1 Device Models with High-Order Monomials

As we mentioned in Section 2, monomial functions can be regarded as exponential of linear functions. So a natural way to extend monomial models for transistor behavioral functions is to write device models as exponential of general linear functions with degree d:

$$\begin{aligned}f_d(y) &= \exp(P_d(y)) \\ &= \exp(\sum_{\sum \alpha_i \le d} c_\alpha x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n})\end{aligned}$$

We define this $f_d(y)$ as the d-degree monomial.

As with fitting standard monomial functions, least square regression can be used to find the optimal coefficient c.

3.2 PolyGP Formulation

Now we formally describe the polynomial geometric programming framework in analog synthesis problems defined as following:

$$\begin{aligned}\min: \quad & APF_0 \\ \text{s.t.}: \quad & APF_i \ge SPEC_i, \quad i = 1, \ldots, p.\end{aligned}$$

In which, each APF denotes one analog performance function, with $SPEC_i$ as the corresponding specification.

As we have mentioned above, a large portion of analog performance functions can be written in terms of transistor behaviors as $APF_i(TBF_1, TBF_2, \ldots, TBF_m)$. For instance, TBF_j's can be g_m, g_{ds}, c_{gs} of some transistors.

With TBF_i in high-order monomial forms, we extend GP to the so-called PolyGP as:

$$\begin{aligned}\min_y: \quad & APF_0(TBF_1, \ldots, TBF_m) \\ \text{s.t.}: \quad & APF_i(TBF_1, \ldots, TBF_m) \ge SPEC_i, \quad i = 1, \ldots, p \\ & TBF_j(y) = \exp(P_d^{(j)}(y)), \quad j = 1, \ldots, m.\end{aligned}$$

Where $P_d(y)$ is the general multivariate polynomial with degree d. So when $d = 1$, all TBF's reduce to monomial functions in design variables.

By eliminating TBF_j's and taking log transformations, we arrive at the general form of PolyGP of degree d as:

$$\begin{aligned}\min_y: \quad & \log(\sum_{k=0}^{K_0} \exp(\sum_{\sum \alpha_i \le d} c_{0\alpha} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n})) \\ \text{s.t.}: \quad & \log(\sum_{k=0}^{K_i} \exp(\sum_{\sum \alpha_i \le d} c_{i\alpha} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n})) \le 0, \quad i = 1, \ldots, m, \\ & \sum_{\sum \alpha_i \le d} d_{i\alpha} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} \le 0, \quad i = 1, \ldots, p.\end{aligned}$$

3.3 SDP Relaxation for Solving PolyGP

Despite the improved power in modeling, these cross-product terms of PolyGP make the problem non-convex. In this section, we show that the semidefinite relaxation scheme can be used for obtaining near-optimal solutions for PolyGP, similarly to what is done in POP.

First we introduce an auxiliary decision matrix M for the

problem linearization:

$$\begin{aligned}
\min_{y,M}: \quad & \log\Big(\sum_{k=0}^{K_0}\exp(\langle A_{0k}, M\rangle)\Big) \\
\text{s.t.}: \quad & \log\Big(\sum_{k=0}^{K_i}\exp(\langle A_{ik}, M\rangle)\Big) \le 0, \quad i=1,\ldots,m, \\
& \langle B_i, M\rangle \le 0, \quad i=1,\ldots,p, \\
& M_{i,j} = \mathbf{y}^{\alpha_i+\alpha_j}.
\end{aligned}$$

In which the α_i is the exponents of monomial i under the predefined multi indices in Section 2 and $\mathbf{y}^\alpha$ is short for $\mathbf{y}^\alpha = y_1^{\alpha_1}y_2^{\alpha_2}\ldots y_n^{\alpha_n}$. Appropriate matrix A's and B's are chosen to be the coefficients of corresponding monomials in M. Where r denotes the relaxation order, which must satisfies the degree constraint: $2r \ge d$.

With the introduction of M, the only nonconvex and nonlinear constraint is the rank one matrix M. Inspired by sums of square polynomials, moment optimization in POP, we can relax this constraint to be positive semidefinite:

$$\begin{aligned}
M &\succeq 0, \\
M_{0,0} &= 1, \\
M_{i,j} &= M_{k,l}, \textbf{if } \alpha_i+\alpha_j = \alpha_k+\alpha_l.
\end{aligned}$$

Then the PolyGP can be approximated and lower-bounded by a sequence of convex problems of relaxation order r with semidefinite constraints, which can be solved by modern convex solvers.

Computational complexity of PolyGP can be characterized by the size of the SDP constraints, as the PSD matrix update dominates the runtime in each iteration of CVX. According to our analysis, moment matrix M is of dimension $O(n^r)\times O(n^r)$, which implies the complexity of PolyGP with relaxation order r as $O(n^{2r})$. The minimum relaxation order for d order monomials is $\lceil d/2\rceil$ for covering all possible polynomials in moment matrix M. By increasing the monomials d and relaxation order r, PolyGP points out a way of exploring the trade-off between model accuracy and computational complexity.

4. EXPERIMENTAL RESULTS

In this section, we test the performance of the proposed PolyGP framework against geometric programming (GP) [2] and polynomial optimization (POP) [11]. In all of these optimization methods, the models are fitted using least-square regression. Transistor characteristics are modeled for the 180nm TSMC high-performance model. All the simulations are done using HSPICE$^{\text{TM}}$. The PolyGP framework has been implemented in MATLAB using the general convex solver CVX [7]. The comparison with GP can be done in CVX as well. To run the comparisons with the POP, we use the sparse polynomial optimization package SparsePOP [18]. All testbenches run on an Intel Xeon 2.93G Linux workstation with 74G memory.

To construct posynomial/polynomial models we used Latin hypercube sampling method [19] with $n = td$ samples for training in an d-dimensional problem, where t represents the number of points for each dimension (determined by cross validation). In addition to training data, we also generate another set of data for model validation through uniform random sampling.

First, we illustrate how high-order monomials can capture the non-linearity and non-convexity of transistor behaviors for a single transistor and reduce the fitting error dramatically compared to monomials. Then, we perform the experiments on the two commonly used benchmarks, a two-stage operational amplifier (OPAMP) and a low-noise amplifier (LNA).

Table 1: Error Reduction for g_{ds} fitting

	Training Error		Test Error	
Method	Max Error	RMS Error	Max Error	RMS Error
Monomial	39.7%	10.51%	33.4%	10.57%
Quad Monomial	9.49%	3.45%	8.91%	3.46%
Cubic Monomial	13.56%	1.94%	13.14%	2.03%

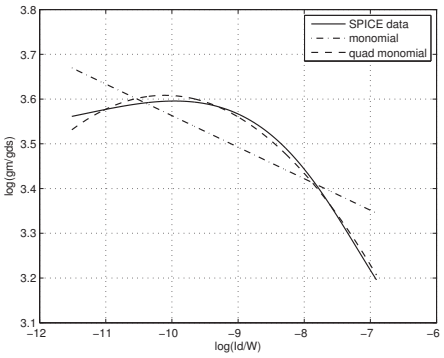


Figure 1: Advantages of quad monomial in capturing the non-linearity of g_m/g_{ds}

The single NMOS transistor is simulated in HSPICE$^{\text{TM}}$ to obtain small-signal functions such as g_m, g_{ds} and c_{gs}, under a 10x range of design variables of (W, L, I). In Table 1, we show the fitting error for the intrinsic output conductance g_{ds}, which is the most difficult metric to fit, using both monomials (first-order) and quadratic monomials. The results indicate that by allowing the non-linearity with more degrees of freedom, high-order monomials are able to reduce the fitting error of g_{ds} of a single transistor from 10% to 2%, on average, compared to monomials. Note that the accuracy increases from the quadratic term to the cubic term is not as significant as that of from the linear to the quadratic one. By taking the $O(n^{2d})$ complexity of subsequent optimization into consideration, we choose to use quadratic monomials for transistor models in OPAMP and LNA implementations.

Figure 1 illustrates why capturing non-linearity and non-convexity is crucial. This example plots g_m/g_{ds} as a function of current density I/W by fixing channel length L for the first-order monomial, the quadratic monomial and the true behavior produced by SPICE simulator. We see that the second-order monomials almost exactly matches the true SPICE curve while the first monomial fit shows significant error over the entire range.

Next we investigate the two-stage operational amplifier. Figure 2a shows the schematic of the two-stage operational amplifier with 8 transistors. We select transistor widths (W), lengths (L), bias current (I_b), and compensation capacitance (C_c) as the input variables. Based on the matching requirements [2], which dictate symmetry between transistors of the differential pair and current mirrors, we are able to reduce the number of independent input variables to 7.

The output circuit performance metrics that we aim to model and optimize are open-loop gain, unity gain bandwidth (UGB), and phase margin (PM). For this circuit, symbolic equations for gain, UGB and PM are available in the analytical forms [2], which we list in Table 2. Here, C_1, C_2 and C_{out} denote total capacitance at the gate of M6 and M3 and at the output node. As we verified with SPICE, the equations for gain and UGB have good accuracy with respect to the small-signal models.

Table 2: Symbolic Analog Performance Functions with Small Signal Models in OPAMP

Gain	$\frac{g_{m2}\times g_{m6}}{(g_{ds2}+g_{ds4})\times(g_{ds6}+g_{ds7})}$
Pole 2	$\frac{g_{m6}\times C_c}{C_1\times C_c+C_1\times C_{out}+Cc\times C_{out}}$
Pole 3	$\frac{g_{m3}}{C_2}$
Pole 4	$\frac{g_{m6}}{C_1}$
UGB	$\frac{g_{m1}}{C_c}$
PM	$\frac{\pi}{2}-\arctan\frac{UGB}{p_2}-\arctan\frac{UGB}{p_3}-\arctan\frac{UGB}{p_4}$

Table 3: Error Reduction for OPAMP Gain

	Training Error		Test Error	
Method	Max Error	RMS Error	Max Error	RMS Error
Quad Monomial	16.9%	2.89%	15.14%	2.96%
Quad Poly	84.0%	9.18%	54.0%	9.6%
Monomial	58.74%	11.85%	52.4%	11.6%

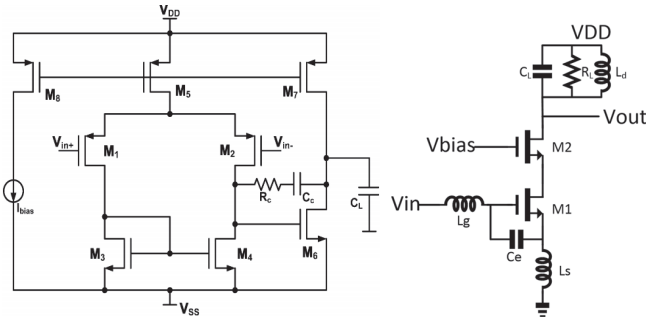
PM constraints can be transformed to the standard PolyGP form by taking arctan as $UGB/p_2 \leq r$. The ratio r is derived by the requirement of phase of pole p_2.

We first show the power of regression of our method in exploring the space of high-degree polynomials for subsequent optimization. The total number of SPICE simulation points used for training is 10,000 and for testing is 1000. We model the function over an input range spanning 4X (from 0.4 to 1.6) of the center value. Table 3 summarizes the fitting results of the OPAMP gain as it is the most non-linear metric. Results indicate that PolyGP regression with quadratic monomials achieves substantial improvement in accuracy with respect to GP: the RMS error is reduced from 12% to 3%. The results show that PolyGP also outperforms the same order of polynomial functions because it utilizes the structural knowledge of symbolic equation. For functions capturing phase margin and unity-gain bandwidth, all three methods deliver sufficiently good accuracy (below 5% RMS error).

Next we demonstrate the effectiveness of PolyGP by evaluating several optimization scenarios. They are summarized in Table 4 in terms of optimal objective, the true value of a function obtained via SPICE simulation and the specification for each metric. Geometric Programming is unable to find good solutions as it could not predict performance within the acceptable accuracy, i.e. 20% error found in the gain spec. Although polynomial optimization performs better than GP with higher degrees, its violation in the gain and PM ratio is still too large (10%) to identify a true feasible solution evaluated by SPICE. In contrast, PolyGP is able to find the good solution for all three cases owing to the accuracy of the model. Notice that in the case 1 when both GP and PolyGP satisfy constraints, PolyGP can deliver better objective function as it captures the trade-off curve between gain and bandwidth more accurately than GP.

Table 4: Optimization Results: OPAMP

			PolyGP		Quad Poly		GP	
Case #	Metric	Spec	Model	SPICE	Model	SPICE	Model	SPICE
Case 1	UGB (MHz)	max	148	148	164	162	137	139
	Gain (10^3)	>2.5	2.5	2.54	2.50	2.57	2.5	2.67
	PM ratio	<0.35	0.35	0.327	0.35	0.37	0.35	0.32
Case 2	UGB (MHz)	max	135.0	135.0	148	148	129	125
	Gain (10^3)	>3.0	3.0	3.03	3.0	2.99	3.0	2.79
	PM ratio	<0.35	0.35	0.327	0.35	0.370	0.35	0.28
Case 3	UGB (MHz)	max	122.7	122.0	139	140	145	149
	Gain (10^3)	>4.5	4.5	4.41	4.5	4.0	4.5	3.7
	PM ratio	<0.5	0.5	0.47	0.5	0.485	0.5	0.486



(a) Two-stage opamp. (b) LNA.

Figure 2: Schematics for experiments.

Table 5: Rate of Constraint Satisfaction

Threshold	PolyGP	Quad Poly	GP
0.1%	57.4%	7.4%	25.9%
1%	75.9%	11.1%	35.2%
2.5%	98.1%	11.1%	35.2%
3%	98.1%	18.5%	35.2%
4%	98.1%	22.2%	44.4%
5%	98.1%	24.1%	44.4%
8%	98.1%	52.9%	57.4%
10%	98.1%	66.6%	59.3%
15%	98.1%	100%	74.1%

Note that even though PolyGP is most accurate among the three options, we cannot guarantee the true feasibility in every case of Table 4 because of the residual model error. However, statistically, higher fitting accuracy translates into better performance in optimization. We run 54 experiments by varying numerical values of constraints and extract the average rate of constraint satisfaction, i.e. the rate of finding true feasible solution given the threshold of violation in Table 5. We find that the PolyGP framework always outperforms the quadratic polynomial model given different violation thresholds. In addition, the average violation error is consistent with the fitting error: the quadratic polynomial has about 7.8% RMS violation, the GP has 5.5%, while the PolyGP can reduce that to 0.5% for the gain constraint.

The runtimes of optimization are summarized in Table 6. Runtime of PolyGP nearly doubles that of GP in order to deal with larger matrix resulting from accurate, yet high-order monomials. In this point of view, our PolyGP framework gives a trade-off between runtime and optimization performance. Runtimes of fitting for all three methods are negligible compared to that of optimization, so we do not list them here.

Figure 2b depicts the schematic of a low noise amplifier (LNA) [15]. Apart from transistor width W and L, we also take the gate inductance L_g, source degenerated inductance L_s and gate-source parallel capacitance C_e into consideration as input variables. By the small signal analysis around the working frequency, the requirement of impedance matching can be written

Table 6: Runtime: OPAMP

Method	Runtime (s)
PolyGP	12.9
Quad Poly	3.57
GP	7.91

Table 7: Error Reduction for LNA Noise Figure

	Training Error		Test Error	
Method	Max Error	RMS Error	Max Error	RMS Error
Quad Monomial	1.90%	0.43%	3.8%	0.52%
Quad Poly	37.0%	12.0%	44.6%	12.9%
Monomial	18.2%	5.55%	33.6%	5.67%

Table 8: Optimization Results: LNA

			PolyGP		GP	
Case #	Metric	Spec	Model	SPICE	Model	SPICE
Case 1	Area	min	21.02	21.02	17.83	17.83
	NF (ratio)	<3.5	3.5	3.6	3.5	4.82
	Gm (S)	>0.005	0.0135	0.0132	0.0137	0.0109
Case 2	Area	min	26.48	26.48	22.2	22.2
	NF (ratio)	<2.5	2.50	2.56	2.5	3.29
	Gm (S)	>0.01	0.0174	0.0170	0.0176	0.0141
Case 3	Area	min	23.22	23.22	19.6	19.6
	NF (ratio)	<3.0	3.0	3.08	3.0	4.05
	Gm (S)	>0.01	0.0151	0.0148	0.0153	0.0122

as $\frac{g_m}{C_{gs}+C_e}L_s = R_s = 50\Omega$. Besides, input matching requires that at the resonance frequency, the impedance of the input stage is purely real, which implies $j\omega_0(L_s+L_g)+\frac{1}{j\omega_0(C_{gs}+C_e)} = 0$, where ω_0 is the resonance frequency. C_e is provided in case that the intrinsic C_{gs} is not large enough.

In the LNA testbench, we aim to minimize the die area with noise figure and amplifier gain constraints. The noise factor F is defined as [15]:

$$F = 1 + \gamma g_m R_s (\frac{\omega_0}{\omega_T})^2 + 4\frac{R_L}{R_s}(\frac{\omega_0}{\omega_T})^2$$

In which, ω_T is the equivalent intrinsic frequency defined by $\omega_T = \frac{g_m}{C_{gs}+C_e}$ and constant γ are picked according to the technology node used [15]. The gain of LNA is characterized by the circuit transconductance G_m defined as $G_m = \omega_T/(\omega_0 R_s(1+\frac{\omega_T L_s}{R_s})) = 1/(2\omega_0 L_s)$. The area of the LNA can be approximated by the sum of transistor area, inductor area and capacitance area denoting using proportionality constants λ_T, λ_L and λ_C as $A = \lambda_T WL + \lambda_L(L_g + L_s) + \lambda_C C_e$.

We follow the same procedure as for the 2-stage OPAMP: pick the range of 7X (0.25 to 1.75) and sample two data sets, 200 points for training and 100 points for test separately.

We first show the ability of regression using high-order monomials to explore the high dimensional space to reduce the fitting error for the noise figure. We choose noise factor because it is the most non-linear among all those metrics. The results are summarized in Table 7.

In this case, the advantage of the PolyGP framework is more pronounced with a 10X improvement in accuracy compared to GP with the first-order monomials and 20X to the quadratic polynomials. Table 8 gives the comparison of optimization performance. In the LNA testbench, formulating optimization problem to be a POP is difficult, as the impedance matching constraints involve fractional polynomial optimization with much higher relaxation order. So we do not include the optimization results of POP here. This result agrees with our theoretical study that improvements in accuracy help reducing the percentage of constraint violation.

5. CONCLUSIONS

In this paper, we present an extension and improvement for GP. We demonstrate that by lifting monomials into high-order monomials, powerful SDP relaxation techniques for (non-convex) polynomial optimization can be leveraged to greatly improve the accuracy of transistor behavioral functions and hence performance functions in equation-based approaches for analog synthesis. Our results demonstrate promise, showing that significant improvements are possible in terms of both model accuracy and reliability in meeting performance constraints.

6. REFERENCES

[1] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming. *Optimization and engineering*, 2007.

[2] S. Boyd, T. Lee, et al. Optimal design of a cmos op-amp via geometric programming. *TCAD*, 2001.

[3] G. Chesi. Lmi techniques for optimization over polynomials in control: a survey. *ITAC*, 2010.

[4] W. Daems, G. Gielen, and W. Sansen. Circuit simplification for the symbolic analysis of analog integrated circuits. *TCAD*, 2002.

[5] W. Daems, G. Gielen, and W. Sansen. Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits. *TCAD*, 2003.

[6] M. del Mar Hershenson. Design of pipeline analog-to-digital converters via geometric programming. In *ICCAD*, 2002.

[7] M. Grant, S. Boyd, and Y. Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[8] J. Kim, J. Lee, and L. Vandenberghe. Techniques for improving the accuracy of geometric-programming based analog circuit design optimization. In *ICCAD*, 2004.

[9] M. Krasnicki, R. Phelps, R. A. Rutenbar, and L. R. Carley. Maelstrom: efficient simulation-based synthesis for custom analog cells. In *DAC*, 1999.

[10] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 2001.

[11] S.-H. Lui, H.-K. Kwan, and N. Wong. Analog circuit design by nonconvex polynomial optimization: Two design examples. *IJCTA*, 2010.

[12] T. McConaghy, P. Palmers, G. Gielen, and M. Steyaert. Simultaneous multi-topology multi-objective sizing across thousands of analog circuit topologies. In *DAC*, 2007.

[13] P. A. Parrilo and B. Sturmfels. Minimizing polynomial functions. *DIMACS*, 2003.

[14] S. Prajna, A. Papachristodoulou, and F. Wu. Nonlinear control synthesis by sum of squares optimization: A lyapunov-based approach. In *ACA*, 2004.

[15] B. Razavi and R. Behzad. *RF microelectronics*. 1998.

[16] H. D. Sherali and C. H. Tuncbilek. A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. *Journal of Global Optimization*, 1992.

[17] A. K. Singh, M. Lok, K. Ragab, C. Caramanis, and M. Orshansky. An algorithm for exploiting modeling error statistics to enable robust analog optimization. In *ICCAD*, 2010.

[18] H. Waki, S. Kim, M. Kojima, M. Muramatsu, and H. Sugimoto. Algorithm 883: Sparsepop—a sparse semidefinite programming relaxation of polynomial optimization problems. *TOMS*, 2008.

[19] H. You, M. Yang, D. Wang, and X. Jia. Kriging model combined with latin hypercube sampling for surrogate modeling of analog integrated circuit performance. In *ISQED*, 2009.