# Enabling Efficient Analog Synthesis by Coupling Sparse Regression and Polynomial Optimization

Ye Wang
University of Texas At Austin
Austin, TX, 78712
wangye805@hotmail.com

Michael Orshansky
University of Texas At Austin
Austin, TX, 78712
orshansky@utexas.edu

Constantine Caramanis
University of Texas At Austin
Austin, TX, 78712
constantine@utexas.edu

## ABSTRACT

The challenge of equation-based analog synthesis comes from its dual nature: functions producing good least-square fits to SPICE-generated data are non-convex, hence not amenable to efficient optimization. In this paper, we leverage recent progress on Semidefinite Programming (SDP) relaxations of polynomial (non-convex) optimization. Using a general polynomial allows for much more accurate fitting of SPICE data compared to the more restricted functional forms. Recent SDP techniques for convex relaxations of polynomial optimizations are powerful but alone still insufficient: even for small problems, the resulting relaxations are prohibitively high dimensional.

We harness these new polynomial tools and realize their promise by introducing a novel regression technique that fits non-convex polynomials with a special sparsity structure. We show that the coupled sparse fitting and optimization (C-SFO) flow that we introduce allows us to find accurate high-order polynomials while keeping the resulting optimization tractable.

Using established circuits for optimization experiments, we demonstrate that by handling higher-order polynomials we reduce fitting error to 3.6% from 10%, on average. This translates into a dramatic increase in the rate of constraint satisfaction: for a 1% violation threshold, the success rate is increased from 0% to 78%.

## 1. INTRODUCTION

Automated analog optimization promises to increase productivity by reducing design time. While automated synthesis that involves selection of circuit topology is the ultimate goal [12], most approaches currently focus on strategies that assume that topology is fixed and focus on optimal device sizing. In this paper, we focus on the class of optimization approaches that construct an analytical equation (model) of circuit behavior based on specific functional forms, or templates and use the model to drive the optimization [4, 2, 11]. These methods are known as equation-based or model-based in contrast to optimization methods that rely directly on SPICE simulations to drive optimization [12, 8]. Simulation-based methods allow higher accuracy of point evaluations of feasibility and ease of use for convergence to local optimal solutions. However, SPICE simulations are time-consuming, and yield little structure that can be exploited for efficient global optimization. Equation-driven methods are able to overcome the sub-optimality by capturing the global structure of the problem and thus delivering a higher-quality solution with guarantees of global optimality.

The most widely explored approach based on convex optimization uses geometric programming (GP) operating on posynomial functions [2, 4]. A problem that a user of GP-based automated flow faces is the need to generate the models of circuit performance in terms of posynomial functions. Early work [2] derived models using first-principles symbolic equations. Not only do these involve significant designer involvement and expertise, but perhaps worse, they are not accurate for nanometer scale technologies. An alternative, is to use automatic model-fitting approaches [3] that fit SPICE-generated pre-characterization data to posynomial functions. This approach too, has met with challenges, primarily because extensive studies have demonstrated that posynomials fail to produce accurate models for large circuits (see, e.g., [6, 15]). In a real sense, the promise in analog synthesis of global optimization techniques that are automated and global, and can hence explore over a large range of design variables, is unmet.

Recent advances in polynomial optimization provide a sequence of convex relaxations via semidefinite optimization, for non-convex polynomial optimization. These methods suggest a promising way forward. Polynomial optimization approaches have been used in diverse areas, including control, combinatorial and continuous optimization, differential equations, and elsewhere. Yet their use in analog synthesis has not been properly explored, in part due to some important challenges which this paper addresses. The key advantages of polynomial optimization are: 1) by permitting model non-linearity and non-convexity, multivariate polynomials are superior to posynomials in finding regression based models of higher accuracy; 2) importantly, although non-convex in both objective and constraints, polynomial optimization problems can be convexified through Semidefinite-Programming (SDP) relaxations [9, 7] guaranteeing theoretical convergence to global optimal solution.

From a practical point of view, there are major limitations of polynomial-based approaches in both modeling and optimization. Primary among these, are the dual questions of how to fit SPICE data, and subsequently how to optimize the potentially large problems that emerge. Where as work in [11] suggests the potential use of polynomial optimization, both the above questions remain unaddressed. The essential difficulty is that high degree polynomials seem unavoidable, if one wants a good fit to the data. Yet out-of-the-box SDP relaxations for high degree polynomial optimization have exponential, and hence prohibitive, size. Thus, while potentially powerful, these polynomial optimization tools are not practical in a generic or naive implementation.

The key innovation in this work is in *coupling the fitting process with the optimization process.* By designing a tailored regression procedure, we fit *non-convex yet structured polynomials*, which we then show are amenable to efficient SDP optimization, despite their potentially high degree. We call this coupled fitting and optimization procedure **C-SFO** for coupled Sparse Fitting and Optimization.

Our work introduces and leverages recent work on polynomial optimization with *structured sparsity*, proposed by [7, 10]. This allows the large variable set resulting from high degree polynomials, to be decomposed into several small sets. This eliminates terms introduced in a naive out-of-the-box application of SDP tools, yet unnecessary in the structured sparsity setting. As we show, this can improve the computational efficiency dramatically, with little to no decrease in quality.

This paper pursues this agenda. We develop an effective framework of analog synthesis by coupling sparse regression and sparse polynomial optimization by adapting ideas from sparse reconstruction and harmonic analysis. The central contribution of this paper is the novel formulation of polynomial regression through structured sparsity into overlapping group-lasso problems [5]. To the best of our knowledge, while much work in polynomial optimization has focused on exploiting *existing sparsity*, or has considered quadratic polynomials ([1]), the idea of coupling fitting and optimization for potentially high order polynomials in the proposed way is novel in analog synthesis, and more generally.

As we explain in detail, the key sparsity notion that can be exploited in the optimization phase, is a group sparsity among overlapping groups of variables. By imposing this sparsity in the fitting phase, we effectively allow the use of high-degree polynomials in the fitting phase, without sacrificing tractability in the optimization phase.

We accomplish this by introducing a novel greedy sparse polynomial group orthogonal-matching pursuit algorithm to effectively find a compact fitted function for use in optimization. This algorithm is in several respects superior to the canonical sparse method through $\ell^1/\ell^2$-norm penalty based on a convex relaxation of non-convex problems [5]. In particular, it is more effective at producing group sparsity, and it is parameter free, which essentially allows for a much more efficient fitting procedure, as we explain below.

Our simulation experiments on established benchmarks show that allowing higher degree polynomials is critical. The fitting error decreases from 10% to 3.6% compared to the polynomial model with degree two, which is the maximum degree that is computationally tractable without sparsity. This improvement in accuracy translates to the ability to identify better feasible solutions and, specifically, it dramatically increases the rate of constraint satisfaction. For instance, for the op-amp circuit, the success rate is increased from 0% to 78%, assuming that a constraint is considered satisfied if the SPICE-validated performance value is no more than 1% away from the specification.

## 2. SDP RELAXATION FOR SPARSE POLYNOMIAL OPTIMIZATION

A generic polynomial optimization problem, defined as:

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

can be solved through a sequence of (linear) semidefinite programming (SDP) relaxations 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 skip the details about this standard framework of moment SDP relaxation, which are contained in [9, 14]. In this section, we introduce the technique of variable decomposition [10, 17] to significantly reduce computational complexity of polynomial SDP relaxation, and the *structured sparsity* [17] which translates to such efficient decomposition.

### 2.1 Variable Decomposition for Polynomial Optimization

Consider a decomposition of $I = \{x_1, \ldots, x_n\}$ into subsets $I_k \subseteq I$ of variables of the original polynomial problem. We would like to have moment conditions only over these (ideally small) subsets $\{I_k\}$ of variables instead of the complete variable set in the general formulation. Because the size of the moment matrices grows much faster than linearly, having many moment matrices over fewer variables, versus having one large moment matrix involving all the variables, is a great advantage in terms of overall size of the resulting optimization problem.

Generically, however, this is not possible. That is, imposing moment conditions for subsets will not in general imply moment conditions for the entire set. In order for this to happen, the subsets $\{I_k\}$ must satisfy the following three properties: every variable must be in their union, i.e., $\{x_1, \ldots, x_p\} = \bigcup_k I_k$, every moment derived from corresponding monomial must be in at least one moment matrices associated with $I_k$, and also, for every $k \leq p-1$, there must exist an $s < k$ such that

$$I_{k+1} \cap \bigcup_{j=1}^{k} I_j \subseteq I_s.$$

This condition is called the *running intersection property.*

Once we have such variable decomposition, we can reformulate the SDP-relaxations as follows:

$$\begin{aligned} \text{minimize}_{\{m_\alpha\}} : \quad & \sum_\alpha p_\alpha m_\alpha \\ \text{subject to} : \quad & M_r(m, I_k) \succeq 0, \quad k = 1, \ldots, p \\ & M_{r-r_i}(f_i m, I_k) \succeq 0, \quad k = 1, \ldots, p, \end{aligned}$$

where the moment matrices are now constrained to the variables in each subset $I_k$.

By variable decomposition, we can bound the size of the SDP relaxation by $O(p \times \max(\text{card}(I_k))^r)$. This is a significant decrease compared to $O(n^r)$, for dense polynomial optimization [9, 14].

### 2.2 Structured Sparsity and Chordal Extension

Although the great reduction through variable decomposition looks promising, finding an optimal decomposition, i.e. one that minimizes the induced SDP size for a given polynomial optimization problem, is NP-hard. The work in [17] proposed the efficient heuristic featuring structured sparsity and the technique of translating structured sparsity into variable decomposition. We show that this specific sparsity concept is the interface we need, and our coupled fitting and optimization procedure, C-SFO, builds upon this idea. We explain the details in Section 3.

The structured sparsity of a polynomial optimization problem is evaluated by the number of correlated variables in both objective and constraints as captured in the following $n \times n$ symmetric correlative-sparsity-pattern (csp) matrix:

$$R_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 1, & \text{if } \alpha_i, \alpha_j \geq 1, \exists \alpha \in \text{supp}(f_0), \\ 1, & \text{if } x_i \in x_{f_k} \text{ and } x_j \in x_{f_k}, k \geq 1, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $x_{f_k}$ represents the set of variables that appear in the expression of polynomial $f_k(x)$. This csp-matrix can also be viewed as the adjacency matrix for the undirected correlation graph $G(N, E)$ with the node set $N = \{1, 2, \ldots, n\}$ and edge set $E = \{(i, j) : R_{ij} = 1\}$. Finding the optimal variable decomposition satisfying the three conditions discussed above is the same as finding maximal cliques of the correlation graph. Well-studied heuristics described in, e.g., [17] are available to generate maximal cliques from chordal extensions of the csp-matrix. The solver sparsePOP ([17]) efficiently detects structured sparsity by generating maximal cliques of a chordal extension. Thus, once the sparse model is properly built, the application of sparsePOP is a routine. Yet fitting SPICE data in order to produce polynomials whose csp-matrix has small maximal cliques, is a difficult problem. As a proxy to this, we have found that it is enough to produce polynomials whose csp-matrix itself is sparse. Henceforth, *we say that a polynomial optimization problem is sparse, if the corresponding csp-matrix is sparse.*

The key remaining challenge, then, becomes how to properly build the polynomial model with structured sparsity. We now move to this problem.

## 3. SPARSE POLYNOMIAL FITTING

We now formally describe the proposed sparse polynomial fitting algorithm, that seeks to produce an accurate fitting using polynomials with structured sparsity as described above. Then we show how we couple sparse polynomial regression and optimization.

We denote each $n$-dimensional point by row vector $X_i = (X_{i1}, X_{i2}, \ldots, X_{in})$ and the SPICE-generated output, i.e., the value of the function we seek to fit, as $y_i$, $i = 1, 2, \ldots, m$ with $m$ as the number of SPICE-generated data points. We seek to find a polynomial function with degree $d$:

$$F(x) = \sum_{p=1}^{l} c_p \prod_{k=1}^{n} x_k^{\alpha_{pk}},$$

$$\sum_{k=1}^{n} \alpha_{pk} \leq d, \forall p \leq l,$$

with all exponents $\alpha_{pk}$ nonnegative, to fit the SPICE data as well as possible.

As discussed above, our true measure of sparsity is the size of the maximal cliques. Yet fitting with small maximal cliques is hard; thus as a proxy, we consider directly imposing sparsity of the csp-matrix. We expand the design sampling matrix $X$ to $\tilde{X}$ of dimension $m \times l$ with each column corresponding to one monomial, where $l$ is the target number of monomials. Thus, we arrive at our initial sparse polynomial regression:

$$\begin{aligned} \text{min}: \quad & \|Y - \tilde{X}C\|_2^2 \\ \text{s.t.}: \quad & \text{card}(\text{supp}(R)) \leq T_{\text{csp}}, \forall k \\ & f(x) = \sum_{p=1}^{l} c_p \prod_{k=1}^{n} x_k^{\alpha_{pk}}, \end{aligned}$$

Here supp$(R)$ is defined as the support for csp-matrix $R$ generated by polynomial model $f$, and $T_{\text{cap}}$ is the target sparsity of $R$.

This formulation is still not convex, due to the cardinality constraint. However, as we discuss in the next section, the cardinality constraint presents fewer challenges than the maximal clique constraint; hence this problem is significantly easier to solve. Moreover, as we show in our results, this formulation coupled with our algorithm for its solution which we give in Section 3.1, indeed results in sparse csp-matrices $R$, and resulting sparse polynomial optimization problems with small maximal cliques, that are quickly solvable by SDP, even for high degrees.

### 3.1 Regression as Overlapping Group Lasso

Sparsity of the csp-matrix does not correspond directly to sparsity of the vector of monomials. In particular, the presence of some monomials may reduce the sparsity of the csp-matrix more than others. For example, if the coefficient of $x_i x_j x_k$ is non-zero, we have three non-zero elements in the csp-matrix, corresponding to $(i, j)$, $(j, k)$, and $(k, i)$. On the other hand, if the coefficient for the term $x_i x_j$ is non-zero, this results in adding a single '1' in the $(i, j)$ position in the csp-matrix.

Thus, in our sparsity formulation, nonzero coefficients of the former kind (i.e., of higher order terms such as $x_i x_j x_k$) are more costly than for terms such as $x_i x_j$. Thus, from the point of view of sparse regression for optimization, we should penalize more monomials that may contribute more non-zero terms to the csp-matrix.

This intuition is also consistent with the physical nature of performance metric of analog circuits: we expect strong coupling only with a subset of the transistors. This intuition is verified in the Section 4. As emphasized above, a key observation that enables the results in this paper, is that the structure required for efficient optimization is also well-matched to the analog synthesis problem. Indeed, this is the case in this discussion as well.

By inspection, we can group monomial terms by their exponents according to their contributions to the csp-matrix by the following $G$:

$$\begin{aligned} G_0 =& \{\alpha\}, \text{card}(\text{supp}(a)) \geq 1 \\ G_{ij} =& \{\alpha\}, \alpha_i \alpha_j \geq 1, i < j. \end{aligned}$$

Monomials with exponents in $G_0$ contribute merely to the diagonal of the csp-matrix and monomials belonging to $G_{ij}$ at least contribute to the $(i, j)$ position of the csp-matrix.

Note that those groups are generally overlapping if we explore polynomials with degree more than 2: $x_i x_j x_k$ is in $G_{ij}$, $G_{jk}$ and $G_{ik}$. Thus our problem is one of minimizing *group sparsity* in a pre-specified set of overlapping groups. In order to solve this problem, we borrow techniques from signal processing and statistics, formulated for group sparsity, e.g., [5]. A central computational technique developed for this problem is the so-called group lasso.

For non-overlapping groups, the standard *regularizer* or *penalty function* typically used, is the so-called $\ell_1/\ell_2$ norm [18] that penalizes the $\ell_1$ norm of the $\ell_2$ norm inside each group. We consider the natural extension of this to overlapping groups. The penalty for coefficient $\omega$ in overlapping-group lasso with the set of groups $G$ is defined as:

$$\Omega_{overlap}^{G}(\omega) = \inf_{\mathbf{v} \in V_G, \sum_{g \in G} v_g = \omega} \sum_{g \in G} \|v_g\|,$$

in which $\|\omega\|$ denotes the Euclidean norm of $\omega$. Here, $V_G \subseteq R^{n \times |G|}$ is the set of $|G|$-tuples of vector

$$\begin{aligned} & \mathbf{v} = (v_g)_{g \in G} \\ & \text{supp}(v_g) \subseteq g, \forall g \in G, \end{aligned}$$

where $g$ is an element defined above in the set of groups $G$.

Regularized by this penalty, we can consider the following convex optimization problem for polynomial regression with structured sparsity:

$$\min_{C \in \mathbb{R}^l} \frac{1}{2} \|Y - \tilde{X}C\|^2 + \lambda \Omega_{\text{overlapping}}^{G}(C).$$

As with standard Lasso optimization, the effect of $\lambda$ is to control the sparsity level. A larger $\lambda$ gives us a sparser solution.

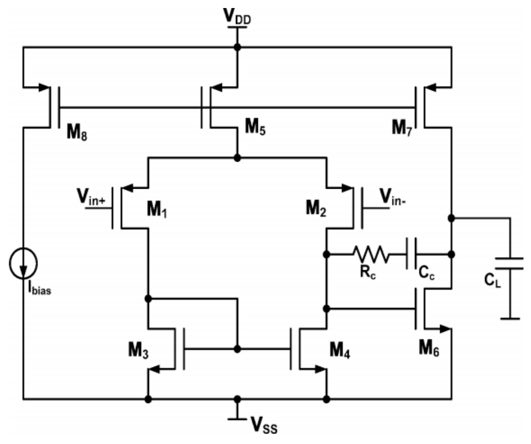

**Figure 1: OPAMP Used for Experiments.**

As $\lambda$ tends to zero, we recover the (unregularized) regression solution. The value of $\lambda$ is typically chosen by cross validation with testing data. Group-coordinate descent [5] or other convex algorithms can be deployed to solve this group lasso efficiently.

Although the approach described in the section above is convex and hence the global optimal solution can be found in polynomial-time, for problems of significant size, as we have in analog optimization, significant improvements are required in order to make the problem practically implementable on a reasonable system. One of the problems with the convex approach described above is that we need to tune parameter $\lambda$. This is typically done by cross validation; moreover, we need to adjust the threshold for identifying groups of nonzero coefficients because there is no direct translation from coefficients to csp-matrix. For large-scale problems like analog performance modeling, iterating through $\lambda$ is time-consuming as for a fixed $\lambda$ we need about 20 mins to converge to an optimal solution. We discuss computational requirements in more detail in Section 4.

In order to overcome the difficulties above, we propose a greedy version of the standard algorithms for solving overlapping group lasso. We call this the Sparse Polynomial Group-Orthogonal-Matching Pursuit Algorithm, and it is the first part of our coupled sparse fitting and optimization framework, or what we have called C-SFO. The approach we propose is simple, highly efficient and scalable. It is motivated by the *orthogonal matching pursuit* algorithm used in statistics and sparse reconstruction (see, e.g., [16]). To the best of our knowledge, its behavior has not been previously explored in the current context, namely, polynomial regression through structured sparsity. By conducting experiments on small-scale test polynomials where the convex optimization approach is feasible, we observe similar performance with respect to the convex optimization method.

Orthogonal Matching Pursuit is a technique that seeks to find the "best matching" projection of multidimensional data onto an over-complete dictionary. In our case, we require to find the next "best matching" group rather than single template in each iteration. Thus, we are effectively seeking the projection not onto the best sparse subspace, but the *best sparse cone*, which hence calls for some modification of the original algorithm. Let us denote the set of all possible polynomial terms up to degree $d$ that do not violate a given csp-matrix $R$ as

$$M(R,d) = \{\alpha, \sum_i \alpha_i \le d, \alpha_i\alpha_j = 0 \text{ if } R(i,j) = 0\}.$$

This set M can be considered as a reverse grouping from csp-matrix to polynomial templates. This reverse grouping has a nice nesting property: if $\mathrm{supp}(R_1) \subseteq \mathrm{supp}(R_2)$, then $M(R_1) \subseteq M(R_2)$. This nesting behavior is precisely what is required in order to allow us to pursue a greedy framework.

While *parameter free* (i.e., we do not need to tune $\lambda$) and generally significantly more efficient, our algorithm behaves at least as well as the convex optimization formulation. Moreover, the algorithm we produce enjoys the same run-time guarantees as standard OMP, and in particular, its running time is at worst *quadratic in the dimension* of the problem, and in the sparsity of the problem.

We describe the algorithm in full detail in Algorithm 1. Here, $R_s$ denotes the current csp-matrix, $P_s$ as polynomial templates selected (i.e., the set of monomials with non-zero coefficient in the polynomial produced by csp-matrix $R_s$), and $r = Y - \tilde{X}_{P_s}C$ the residual at each iteration. $MaxIter$ represents the max number of iterations, and Tol the accuracy tolerance. Before applying this algorithm, we need to normalize all columns of the design matrix $\tilde{X}$ to eliminate any bias effect.

**Algorithm 1** Sparse Polynomial Group-Orthogonal Matching Pursuit Algorithm

```
 1: procedure GREEDY POSYNOMIAL FITTING($\tilde{X}$,$Y$,$d$)
    initialization
 2:     $R_s = I(n \times n)$
 3:     $P_s = M(R_s, d), i = 0$
 4:     $C = \arg\min(\|Y - \tilde{X}_{P_s}C\|_2)$
 5:     $r \leftarrow Y - \tilde{X}_{P_s}C$        ▷ Least-square regression for $G_0$
    Main Loop
 6:     while 1 do
 7:         $(i,j)^* = \arg\max_i(\|\langle \tilde{X}_{M(R_s|E_{ij},d)\setminus M(R_s,d)}, r\rangle\|)$    ▷ Find the most correlated group $(i^*, j^*)$
 8:         if $R_s(i,j) \neq 0$ then        ▷ Update the csp-matrix
 9:             $R_s(i,j) = 1$;
10:             $P_s = M(R_s, d)$
11:         else
12:             Break;                    ▷ Already in csp-matrix
13:         end if
14:         $C = \arg\min(\|Y - \tilde{X}_{P_s}C\|_2)$
15:         $i \leftarrow i+1$, $r \leftarrow Y - \tilde{X}_{P_s}C$        ▷ Update r and i
16:         if $\|r\| <$ Tol or $i > MaxIter$ then
17:             Break;
18:         end if
19:     end while
20: end procedure
```

## 3.2 Dealing with Constrained Optimization

Fitting structured sparse polynomials is not the end of the story. There is a significant complication introduced by adding constraints to the optimization problem. This is because for constraints, sparsity is defined in a somewhat different manner.

In particular, the structured sparsity corresponding to constraints is more stringent than that corresponding to the objective. For a given constraint, the csp-matrix $R$ has $R(i,j) = 1$ if $x_i$ and $x_j$ appear simultaneously – that is, they do not need to appear as a product. The reason for this is that there is a cross product $x_ix_j$ in the dual Lagrangian polynomial in the framework of polynomial optimization.

In order to overcome this difficulty, we essentially form the Lagrangian formally, before fitting polynomials to the functions. We then fit a single sparse polynomial to the resulting problem. Binary search method can be employed to find optimal Lagrangian duals by increasing the multiplicative weights whose constraints are not satisfied. We illustrate the results of this procedure in Section 4.

**Table 1: Error Reduction for OPAMP Gain**

| | Training Error | | Test Error | |
|---|---|---|---|---|
| Method | Max Error | RMS Error | Max Error | RMS Error |
| Monomial | 581% | 36.5% | 458% | 36.4% |
| Quad Poly | 145% | 9.17% | 58.1% | 9.24% |
| C-SFO | 61.3% | 3.73% | 34.2% | 3.69% |

**Table 2: Rate of Constraint Satisfaction**

| Threshold | C-SFO | Quad Poly |
|---|---|---|
| 0.1% | 42.5% | 0% |
| 1% | 77.9% | 0% |
| 2.5% | 80.3% | 1.6% |
| 3% | 81.1% | 22.0% |
| 4% | 85.8% | 73.2% |
| 5% | 88.9% | 79.5% |

## 4. EXPERIMENTAL RESULTS

We test the performance of the proposed C-SFO framework for coupling sparse regression and polynomial optimization against geometric programming (GP) [2] and polynomial optimization (POP) without sparsity [11]. In both of these optimization methods, the models are fitted using least-square regression. Transistor I-V characteristics are modeled for the 180nm TSMC high-performance model. All the simulations are done using HSPICE$^{TM}$. The C-SFO framework has been implemented in MATLAB using the sparse polynomial optimization package SparsePOP [17]. To run the comparisons with the GP, we use GGPLAB [13]. All testbenches run on an Intel Xeon 2.93G Linux workstation with 74G memory.

We perform the experiments on the two commonly used benchmarks, a two-stage operational amplifier (OPAMP) and a voltage-controlled oscillator (VCO).

Figure 1 shows the schematic of the two-stage operational amplifier. The output circuit performance metrics that we aim to model are open-loop gain, unity gain bandwidth (UGB), and phase margin (PM). 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 11.

First, we 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 is 100,000 for training and 5000 for testing. We model the function over an input range spanning 40% (from 0.8 to 1.2) of the center value. Table 1 summarizes the fitting results of the OPAMP gain. Without sparsity, the maximum degree of a polynomial we can afford in optimization is quadratic. Results indicate that sparse polynomial regression achieves substantial improvement in accuracy: the RMS error is reduced from 10% to 3.6% compared to the polynomial model. We use the amplifier gain for comparison because it is the most non-linear and difficult function to fit (10% rms error for quadratic polynomial). For functions capturing phase margin and unity-gain bandwidth, all three methods deliver sufficiently good accuracy (below 1.5% rms error).

**Table 3: Optimization Results: OPAMP**

| | | | C-SFO | | Quad Poly | | GP | |
|---|---|---|---|---|---|---|---|---|
| Case # | Metric | Spec | Model | SPICE | Model | SPICE | Model | SPICE |
| Case 1 | Gain ($10^4$) | max | 1.69 | 1.68 | 1.74 | 1.68 | 1.95 | 0.78 |
| | UGB ($MHz$) | >10 | 10.2 | 10.5 | 9.98 | 9.96 | 10 | 9.76 |
| | PM (°) | >60 | 60.6 | 60.5 | 60 | 60 | 60 | 58.2 |
| Case 2 | Gain ($10^4$) | >1.5 | 1.55 | 1.53 | 1.5 | 1.36 | 1.5 | 0.48 |
| | UGB ($MHz$) | max | 12.6 | 12.6 | 14.8 | 14.8 | 18.4 | 17.8 |
| | PM (°) | >60 | 61.3 | 60.0 | 60 | 60.7 | 60 | 59.71 |
| Case 3 | Gain ($10^4$) | >1.50 | 1.54 | 1.52 | 1.49 | 1.37 | 1.5 | 0.57 |
| | UGB ($MHz$) | max | 18.9 | 19.0 | 14.9 | 14.8 | 25.6 | 25.6 |
| | PM (°) | >48 | 47.8 | 46 | 48 | 47.95 | 48 | 48.1 |

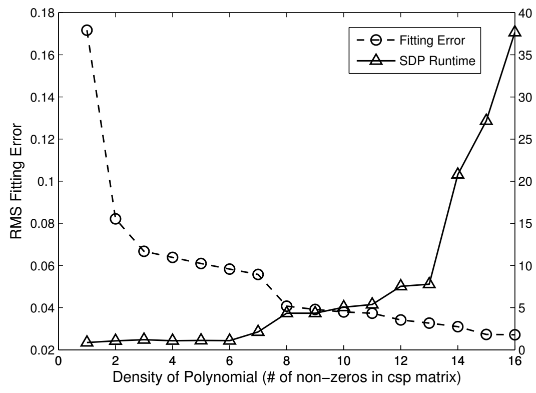

**Figure 2: Trade-off between Sparsity and Optimization Complexity Enabled by C-SFO.**

**Table 4: Runtime: OPAMP**

| Method | Runtime (s) |
|---|---|
| GP | <1 |
| Quad Poly | 20 |
| C-SFO | 100 |

We also give the fitting error and model complexity plot of C-SFO as we increase the polynomial density to demonstrate why our approach gives a trade-off when exploring the high-degree space. From Figure 2, we see that as we increase the density (number of non-zero elements in csp matrix), the RMS error saturates while computational complexity explodes in the mean time.

Next we demonstrate the effectiveness of C-SFO by evaluating several optimization scenarios summarized in Table 3, 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 acceptable accuracy, i.e. 50% error found in the gain spec. Although quadratic polynomial performs better than GP, its violation in the gain spec is still too large to identify a true feasible solution evaluated by SPICE. In contrast, C-SFO is able to find the good solution for all three cases owing to the accuracy of the model.

Note that even though C-SFO is most accurate among all three options, we cannot guarantee the true feasibility in every case, e.g. case 3 of Table 3, because of the residual model error. However, statistically, higher fitting accuracy translates into better performance in optimization. We run 127 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 threshold of violation in Table 2. We find that C-SFO framework always outperforms the quadratic polynomial model given different violation thresholds. In addition, average violation error is consistent with fitting error, that quadratic polynomial have about 4% average violation while the sparse polynomial model can reduce that to 2.5%.

In terms of runtime, the results are summarized in Table

**Table 5: Error Reduction for VCO Max Frequency**

| | Training Error | | Test Error | |
|---|---|---|---|---|
| Method | Max Error | RMS Error | Max Error | RMS Error |
| Monomial | 15.9% | 2.6% | 9% | 2.7% |
| Quad Poly | 4.1% | 1.03% | 5.8% | 1.5% |
| C-SFO | 1.5% | 0.17% | 0.68% | 0.15% |

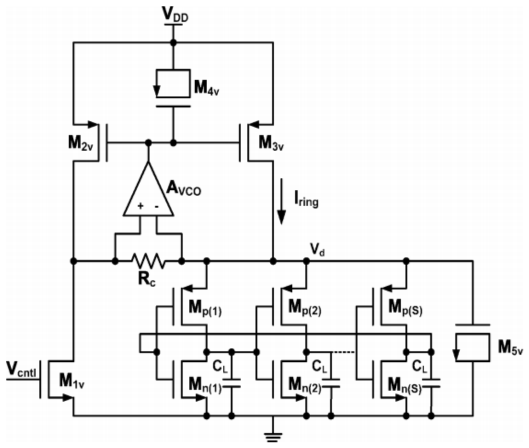

**Figure 3: VCO Used for Experiments.**

**Table 6: Optimization Results: VCO**

| | | | C-SFO | | Quad Poly | | GP | |
|---|---|---|---|---|---|---|---|---|
| Case # | Metric | Spec | Model | SPICE | Model | SPICE | Model | SPICE |
| Case 1 | Power | min | 4.17 | 4.17 | 3.30 | 4.07 | N/A | N/A |
| | FMax ($MHz$) | >2.5 | 2.507 | 2.507 | 2.5 | 2.43 | N/A | N/A |
| | FMin ($MHz$) | <0.5 | 0.479 | 0.479 | 0.423 | 0.46 | N/A | N/A |
| Case 2 | Power | min | 3.054 | 3.054 | 2.9 | 3.45 | 3.11 | 3.02 |
| | FMax ($MHz$) | >2 | 2.00 | 2.00 | 2.00 | 1.98 | 2.00 | 1.97 |
| | FMin ($MHz$) | <0.5 | 0.369 | 0.36 | 0.36 | 0.40 | 0.39 | 0.36 |
| Case 1 | Power | min | 5.44 | 5.43 | 4.74 | 5.41 | N/A | N/A |
| | FMax ($MHz$) | >3 | 3.01 | 3.021 | 3.00 | 2.97 | N/A | N/A |
| | FMin ($MHz$) | <0.6 | 0.598 | 0.599 | 0.59 | 0.595 | N/A | N/A |

4. One iteration of C-SFO takes 5 seconds even with relaxation order 5. In order to converge to acceptable accuracy for Lagrangian multipliers, we need 20 iterations on average for binary search, mentioned in Section 3. In this point of view, our framework also gives a trade-off between runtime and optimization performance.

Figure 3 depicts the schematic of a voltage-controlled oscillator. Apart from transistor width $W$ and $L$, we also take the control voltage $V_{cntl}$ into consideration as an input variable. By symmetry between transistors between current mirrors and equal strength between transistors in inverters, we are able to reduce number of input variable to 6. The output performance considered here are maximum, minimum frequency and power. We follow the same procedure as 2-stage OPAMP cases: pick the range of 80% (0.6 to 1.4) and sample two data sets, 1000 points for training and 100 points for test separately.

We first show the ability of regression in C-SFO to explore the high-degree polynomial space to reduce fitting error for maximum frequency. Again we choose maximum frequency because it is the most difficult to fit among all three metrics. Results are summarized in Table 5. In this case, the advantage of the C-SFO framework is more pronounced with a 20X improvement in accuracy comparison to posynomial and 10X to quadratic polynomial. Table 6 gives the comparison of optimization performance. In cases 1 and 3, GP finds no feasible solution while the sparse polynomial model is able to identify feasible solutions. This result agrees with our theoretical study that improvements in accuracy help reducing the rate of constraint violation. Particularly in the VCO case, our C-SFO framework is superior to others in all three cases. That can be explained in terms of a better model that we fit to SPICE data.

## 5. CONCLUSIONS

In this paper, we present a coupled approach, combining fitting and polynomial optimization, C-SFO. We demonstrate that by tailoring regression to the optimization procedure, powerful SDP techniques for (non convex) polynomial optimization can be leveraged to greatly improve the accuracy of 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] A. S. Bandeira, K. Scheinberg, and L. N. Vicente. Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization. *Mathematical programming*, 2012.

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

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

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

[5] L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *ICML*. ACM, 2009.

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

[7] S. Kim, M. Kojima, and H. Waki. Generalized lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems. *SIAM Journal on Optimization*, 2005.

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

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

[10] J. B. Lasserre. Convergent sdp-relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, 2006.

[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] A. Mutapcic, K. Koh, S. Kim, and S. Boyd. Ggplab version 1.00 a matlab toolbox for geometric programming, 2006.

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

[15] A. K. Singh, K. Ragab, M. Lok, C. Caramanis, and M. Orshansky. Predictable equation-based analog optimization based on explicit capture of modeling error statistics. *TCAD*, 2012.

[16] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans Inf Theory*, 2007.

[17] 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.

[18] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *JRSS*, 2006.