Efficient Multiple Starting Point Optimization for Automated Analog Circuit Optimization via Recycling Simulation Data

Bo Peng[1], Fan Yang[1*], Changhao Yan[1], Xuan Zeng[1] and Dian Zhou[1,2*]
[1]State Key Lab of ASIC & System, School of Microelectronics, Fudan University, Shanghai, P. R. China
[2]Department of Electrical Engineering, University of Texas at Dallas, Richardson, TX, U.S.A.

***Abstract*—Multiple starting point optimization is an efficient approach for automated analog circuit optimization. Starting from a set of starting points, the corresponding local optimums are reached by local optimization method Sequential Quadratic Programming (SQP). The global optimum is then selected from these local optimums. If one starting point is located in a valley, it converges rapidly to the local optimum by the local search. Such a region-hit property makes the multiple starting optimization approach more likely to reach the global optimum. However, the SQP method needs the gradients to drive the optimization. In the traditional method, the gradients are approximated by finite differences. A large number of simulations are needed to obtain the gradients, which becomes the bottleneck of the circuit optimization. We find that for a new point, it is usually surrounded by several neighboring points which have been evaluated in the previous SQP steps. In this paper, we propose an efficient method to calculate the gradient by recycling the previous evaluated points. It is based on the relationship between gradients and the directional derivatives along the directions of the neighbor points. If the neighboring points are not enough for gradient calculation, we will sample adequate neighboring points for gradient calculation. Furthermore, since the performances of the circuits are not sensitive to some design parameters, the gradients are usually sparse. We can thus further employ the idea of sparse recovery to recover the sparse gradients with fewer simulations. Our experimental results demonstrate that with these strategies, the number of simulations can be reduced by up to 63% without significantly surrendering the accuracy of the optimization results.**

I. Introduction

Analog and mix-signal circuit design requires specialized knowledge and circuit design skills acquired through many years of experience [1]. Advances in computer-aided tools, optimization algorithm and parallel computation provide ample power for automated analog circuit design. Compared with manual design, automated circuit design is promising to increase the design productivity by not only reducing the design time but also improving the circuit performance. Automated circuit design involves the selection of circuit topology and device sizing. However, the optimal device sizing attracts many research interests because it can be well formulated as constrained nonlinear optimization problems and thus possibly be solved by the well-developed optimization algorithms.

Current device sizing approaches can be categorized into model-based and simulation-based ones [1]. Optimizations of the model-based methods are driven by simplified equations or polynomial models. These equations or polynomial models approximate the performances of the circuits. One of the well-known model-based method is the geometric programming (GP) method, which employ polynomials to approximate the performances of the circuits [2], [3]. The polynomial approximation guarantees the optimization problem is convex and thus it is possible to find the global optimum. The limitation of the GP method is that the polynomial approximations cannot accurately approximate the performances of circuits especially for the large-scale circuits. Recent advances in polynomial optimization show that the general polynomial optimization problem can be transformed to convex problem by Semidefinite-Programming (SDP) relaxations, which makes it possible to find the global optimum for general polynomial optimization problem [4]. In order to reduce the complexity of models, the sparse-regression-based method was proposed to fitting the performance models by exploiting the sparsity of the problems [4]. The problem of the SDP-based polynomial optimization method is that the global polynomial approximations cannot guarantee the modeling accuracy over the whole design space. Thus, the global optimum found by the SDP-based polynomial optimization is not the optimum of the original problem. On the other hand, in order to improve the accuracy of the polynomial models by high-order models, the number of sampling data points would increase greatly, which prohibits it from practical applications.

Simulation-based optimization is an alternative approach for device sizing. It is driven directly by SPICE simulation. With the data points produced by SPICE simulations, the simulation-based optimization can converge quickly to the local optimums. In order to find the global optimums, general global optimizations such as Simulated Annealing (SA) algorithm [5], Genetic Algorithm (GA) [6] and Particle Swarm Optimization (PSO) algorithm [7] are used to better explore the solution space and avoid falling into the local optimums. Due to the stochastic characteristics of these general global optimization algorithms, the converge rates of these algorithms are relatively low. Multiple starting point optimization algorithm has been proposed for device sizing recently [8]. Multiple starting point optimization is an efficient approach for automated analog circuit optimization. Starting from a set of starting points, the corresponding local optimums are reached by local optimization method Sequential Quadratic Programming (SQP). The global optimum is then selected from these local optimums. If one starting point is located in a valley, it converges rapidly to the local optimum by the local search. As the number of starting points increases, the multiple starting point optimization has high probability to find the global optimum. Such a region-hit property makes the multiple starting optimization approach more likely to reach the global optimum. It has been shown in [9] that the convergence rate of the multiple starting point algorithm is remarkably higher than that of general optimization algorithms such as GA, PSO

*Corresponding authors: {yangfan, zhoud}@fudan.edu.cn.

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

and SA.

However, in the multiple starting point optimization method, the SQP method needs the gradients to drive the optimization. In the traditional method, gradients are approximated by finite differences. A large number of simulations are needed to obtain the gradients, which becomes the bottleneck of the circuit optimization. We find that for a new point, it is usually surrounded by several neighboring points which have been evaluated in the previous SQP steps. In this paper, we propose an efficient method to calculate the gradients by recycling the previous evaluated points. It is based on the relationship between gradient and the directional derivatives along the directions of the neighbor points. The gradient and the directional derivatives satisfy a set of linear equations. If the neighboring points are not enough for gradient calculation, we will sample adequate neighboring points for gradient calculation. Furthermore, since the performances of the circuits are not sensitive to some design parameters, the gradients are usually sparse. We can thus further employ the idea of sparse recovery to recover the sparse gradients with fewer simulations. Note that the time of solving the linear equations of gradient and directional derivatives is negligible compared with the time of SPICE simulations. Our experimental results demonstrate that with these strategies, the number of simulations can be reduced by up to 63% without significantly surrendering the accuracy of optimization results.

The rest of paper is organized as follows. In section II, the background of our method is introduced. Details of the algorithm are presented in section III. Section IV shows the experimental simulation results. Finally conclusions are drawn in section V.

II. Background Review

Given the circuit topology, the automated analog circuit optimization problem can be formulated as a constrained nonlinear optimization problem

$$\begin{aligned} &\min \ f(\vec{x}) \\ &\text{s.t. } g_i(\vec{x}) \leq 0, i = 1, \cdots, m_c, \end{aligned} \tag{1}$$

where $f(\vec{x})$ is the objective function, $\{g_i(\vec{x}), i = 1, \cdots, m_c\}$ are the m_c constraints, $\vec{x} \in R^d$ denotes the d design variables. Generally, for each design variable x_i, it is limited in a range $[p_i^-, p_i^+]$ due to the fabrication process or design requirements.

The objective function $f(\vec{x})$ represents the performance target of the circuit. In general, the objective function can be formulated as a sum of weighted performances

$$f(\vec{x}) = \sum_{j=1}^{n_1} w_j * p_j(\vec{x}), \tag{2}$$

where $\{p_j(\vec{x}), j = 1, \cdots, n_1\}$ denotes a set of target performances, the coefficients $\{w_j, j = 1, \cdots, n_1\}$ are the weights of the corresponding performances. The constraints $\{g_i(\vec{x}), i = 1, \cdots, m_c\}$ represent the performance constraints which should be satisfied during optimization.

A. Multiple Starting Point Algorithm

The analog circuit optimization problem as described in (1) is a general constrained nonlinear optimization problem. It would be difficult to find the global optimum of this problem. Multiple Starting Point (MSP) algorithm is an efficient strategy to search the whole design space and find the global optimum.

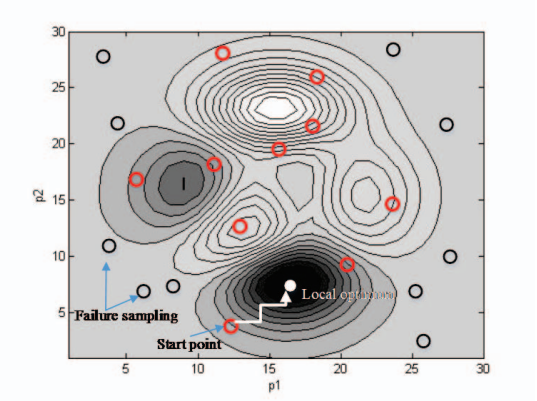

Fig. 1. Contour of two-dimensional parameter space

Firstly, a set of starting points are uniformly sampled using Sobol quasi-random sequence in different regions of the space

$$sp = \{\vec{p} | p_i \in [p_i^-, p_i^+], i = 1, 2, \cdots, d\}. \tag{3}$$

We first evaluate the performances of these starting points in parallel by simulations. The quality of the starting points are obtained by considering both the effects of target performances and constraints. The starting points with better quality are selected from these initial starting points. The number of start points is heuristically selected, considering a trade-off between speed and quality. As shown in Fig. 1, these circles represent the initial starting points. After the filtering, only the red ones are kept as good starting points. The red starting points would be taken as the initial points for parallel local searches.

From Fig. 1, we can see that if one starting point is located in a valley, it converges rapidly to the local optimum during the local search. As the number of starting points increases, the multiple starting point optimization has high probability to find the global optimum. Such a region-hit property makes the multiple starting optimization approach more likely to reach the global optimum.

B. Local Search Using SQP

After the good starting points are selected from the initial starting points, a serial of local searches are invoked to find the local optimums from these good starting points. Note that local searches from different starting points can be invoked in parallel because their search routines are independent generally. Sequential Quadratic Programming (SQP) is the most successful method for constrained nonlinear optimization problems [10]. Since SQP is a quasi-Newton method, it converges rapidly to the local optimum. The description of SQP algorithm can be found in [11]–[14]. SQP is used as the local search algorithm in our multiple starting point method. We give a brief introduction to the SQP algorithm here.

The basic idea of SQP is to model a constrained nonlinear optimization problem in (1) by a Quadratic Programming (QP) subproblem at each iteration, since such problems are relatively easier to solve. Given the formulation (1), a Lagrangian function is constructed firstly

$$L(\vec{x}, \lambda) = f(\vec{x}) + \sum_{i=1}^{m_c} \lambda_i \cdot g_i(\vec{x}), \tag{4}$$

where $\{\lambda_i, i = 1, ..., m_c\}$ are the Lagrange multipliers used for balancing the deviations in magnitude of the objective function and constraints. A QP subproblem can be expressed as

$$\begin{aligned} \min\ & \nabla f(\vec{x}_k)^T \vec{d}_k + \frac{1}{2}(d_k)^T B_k \vec{d}_k \\ \text{s.t.}\ & \nabla g_i(\vec{x}_k)^T \vec{d}_k + g_i(\vec{x}_k) \leq 0, i = 1, ..., m_c, \end{aligned} \tag{5}$$

where B_k denotes the quasi-Newton approximation to Hessian matrix of the Lagrangian function (4). At each major iteration, B_k is calculated by the BFGS method [15]

$$B_{k+1} = B_k + \frac{\vec{q}_k \vec{q}_k^T}{\vec{q}_k^T \vec{s}_k} - \frac{B_k \vec{s}_k \vec{s}_k^T B_k^T}{\vec{s}_k^T B_k \vec{s}_k}, \tag{6}$$

where

$$\vec{s}_k = \vec{x}_{k+1} - \vec{x}_k, \tag{7}$$

$$\vec{q}_k = \left(\nabla f(\vec{x}_{k+1} + \sum_{i=1}^{m_c} \lambda_i \cdot \nabla g_i(\vec{x}_{k+1}))\right) - \left(\nabla f(\vec{x}_k + \sum_{i=1}^{m_c} \lambda_i \cdot \nabla g_i(\vec{x}_k))\right). \tag{8}$$

Here, $\{\lambda_i, i = 1, ..., m_c\}$ is an estimate of the Lagrange multipliers. The solution $\vec{d}_k$ of (5) is the search direction used to find a new iteration point $\vec{x}_{k+1}$ from $\vec{x}_k$. In each iteration, the new iteration point is generated as

$$\vec{x}_{k+1} = \vec{x}_k + t_k \vec{d}_k, k \geq 0, \tag{9}$$

where t_k is the step length determined by an appropriate line search procedure. The gradients are crucial to the SQP algorithm, since search directions are obtained based on the gradients of both objective function and constraint functions. For automated analog circuit optimization, the gradients are approximated by finite difference methods. For each gradient evaluation with d design variables, the finite difference needs d times SPICE simulations of the circuits. It is crucial to reduce the complexity of gradient evaluation.

III. Efficient Gradient Calculation via Recycling Simulation Data

In this paper, we follow the basic idea of multiple starting point algorithm but propose a novel gradient evaluation method to drive the local SQP search. We propose to recycle the existing simulation data to reduce the complexity of gradient evaluation. We find that for a new point, it is usually surrounded by several neighboring points which have been evaluated in the previous SQP steps. These neighboring points are reused to accelerate the calculation of the gradient at the new point. This idea is based on the relationship between gradient and the directional derivatives along the directions of the neighbor points. We will present the relationship between gradient and the directional derivatives, the recycling scheme and the sparse regression method for the sparse gradients.

A. Relationship between Gradient and Directional Derivatives

We consider a general nonlinear function $q(\vec{x})$, where $\vec{x} = \{x_1, x_2, \cdots, x_d\}$. Its directional derivative along direction $\vec{l}$ can be expressed as

$$\frac{\partial q(\vec{x})}{\partial \vec{l}} = \lim_{||\vec{l}|| \to 0} \frac{q(\vec{x} + \vec{l}) - q(\vec{x})}{||\vec{l}||}.$$

The relationship between gradient and the directional derivative is expressed as

$$\frac{\partial q(\vec{x})}{\partial \vec{l}} = \frac{\vec{l}^T \cdot grad(\vec{x})}{||\vec{l}||}, \tag{10}$$

where $grad(\vec{x})$ is the gradient of $q(\vec{x})$, and

$$grad(\vec{x}) = \left[\frac{\partial q(\vec{x})}{\partial x_1}, \cdots, \frac{\partial q(\vec{x})}{\partial x_d}\right]^T.$$

Traditionally, the gradient $grad(\vec{x})$ is approximated by finite difference. For the i-th component of $grad(\vec{x})$, it can be approximated by

$$\frac{\partial q(\vec{x})}{\partial x_i} \approx \frac{q(\vec{x} + h\vec{e}_i) - q(\vec{x})}{h},$$

where h can be viewed as the step of the finite difference, $\vec{e}_i$ is the i-th row of a $d \times d$ identity matrix. With such a finite difference strategy, no data points simulated before can be reused. For a new point $\vec{x}_0$, it is usually surrounded by several neighboring points which have been evaluated in the previous steps. We assume that these neighboring points can be expressed as

$$\{\vec{x}_1, \cdots, \vec{x}_t\},$$

where t is the number of neighboring points. Distances between these points and x_0 satisfy

$$||\vec{x}_i - \vec{x}_0|| \leq C,$$

where C is a constant in the same order as the step length of the finite difference. We use C to control the accuracy of the approximation.

We consider the directional derivatives $\{\frac{\partial q(\vec{x})}{\partial \vec{l}_i}, i = 1, \cdots, t\}$ where $\{\vec{l}_i = \vec{x}_i - \vec{x}_0, i = 1, \cdots, t\}$. These directional derivatives can be approximated by finite difference as

$$\frac{\partial q(\vec{x})}{\partial \vec{l}_i} \approx \frac{q(\vec{x}_i) - q(\vec{x}_0)}{||\vec{x}_i - \vec{x}_0||}.$$

According to the relationship between directional derivatives and the gradients (10), we have the following equations

$$A \cdot grad(\vec{x}_0) = \vec{b}. \tag{11}$$

where

$$A = \begin{bmatrix} \vec{l}_1^T \\ \vec{l}_2^T \\ \vdots \\ \vec{l}_t^T \end{bmatrix} = \begin{bmatrix} (\vec{x}_1 - \vec{x}_0)^T \\ (\vec{x}_2 - \vec{x}_0)^T \\ \vdots \\ (\vec{x}_t - \vec{x}_0)^T \end{bmatrix} \quad \vec{b} = \begin{bmatrix} q(\vec{x}_1) - q(\vec{x}_0) \\ q(\vec{x}_2) - q(\vec{x}_0) \\ \vdots \\ q(\vec{x}_t) - q(\vec{x}_0) \end{bmatrix}. \tag{12}$$

Obviously, if the number of neighboring points t equals to the number of design variables d, the gradient $grad(\vec{x}_0)$ can be solved directly from (11).

B. Filter and Generate Neighboring Points

If the number of neighboring points t is larger than the number of design variables d, we must select d data points from t. In order to guarantee the accuracy of the gradients obtained by (11), those selected data points should satisfy the following conditions

- the selected data points should be those closest data points to $\vec{x}_0$,
- the condition number of the matrix A in (11) should be as small as possible.

We propose an algorithm based on Gram-Schmidt process as shown in Algorithm 1 to filter the data points. Firstly, we sort the neighboring data points $\{\vec{x}_1, \cdots, \vec{x}_t\}$ in ascending

order according to their distances to x_0. Afterward, we use the Gram-Schmidt process to filter the data points so that the data points are not linearly dependent. The first d points are then selected from these filtered data points for gradient calculation.

Algorithm 1 Algorithm of Filtering Data Points

Input: Data point $\vec{x}_0$ and its neighboring data points $\{\vec{x}_1, \cdots, \vec{x}_t\}$, the dimension d of $\vec{x}_0$
Output: The data points indexes $inds$ used for gradient calculation

Sort the neighboring data points $\{\vec{x}_1, \cdots, \vec{x}_t\}$ according to their distances to x_0 in ascending order
$\vec{q}_1 = (\vec{x}_1 - \vec{x}_0)$, $\vec{q}_1 = \vec{q}_1/\text{norm}(\vec{q}1)$
$Q = [\vec{q}_1]$, $inds = 1$
for $i = 2 : t$ **do**
  $\vec{q}_i = \vec{x}_i - \vec{x}_0$
  $\vec{q}_i = \vec{q}_i - Q[1 : i-1] * (Q[1 : i-1]^T * \vec{q}_i)$
  $e = norm(\vec{q}_i)$
  if $e < \epsilon$ **then**
    continue
  end if
  $\vec{q}_i = \vec{q}_i/e$, $Q = [Q, \vec{q}_i]$, $inds = [inds, i]$
end for

On the other hand, if the number of neighboring points t is smaller than the number of design variables d, we must generate new data points near the data point $\vec{x}_0$. In order to efficiently explore the new state space, we firstly randomly generate a group of candidate data points. Then, a Gram-Schmidt process similar to that in Algorithm 1 is used to filter these candidate data points so that the generated data points are not linearly dependent. The algorithm is summarized in Algorithm 2.

Algorithm 2 Algorithm of Generating New Points

Input: Data point $\vec{x}_0$ and its neighboring data points $\{\vec{x}_1, \cdots, \vec{x}_t\}$, the dimension d of $\vec{x}_0$
Output: The new points $\{\vec{x}_{t+1}, \cdots, \vec{x}_d\}$ generated for gradient calculation

for $i = 1 : t$ **do**
  $\vec{p}_i = \vec{x}_i - \vec{x}_0$
end for
$P = [p_1, \cdots, p_t]$
$[Q, R] = qr(P)$
for $i = t+1 : d$ **do**
  Randomly generate $\vec{x}_i$ such that $\|\vec{x}_i - \vec{x}_0\| < C$
  $\vec{q}_i = \vec{x}_i - \vec{x}_0$
  $\vec{r} = \vec{q}_i - Q[1 : i-1] * (Q[1 : i-1]^T * \vec{q}_i)$
  $e = norm(\vec{r})$
  if $e > \epsilon$ **then**
    $i = i + 1$
    $\vec{r} = \vec{r}/e, Q = [Q, \vec{r}]$
  end if
end for

The proposed approach was used to calculate the gradient of the power consumption of an operational amplifier, which has 24 design variables. As shown in Fig. 2, the approximated gradient obtained by the proposed approach matches that by the finite difference well.

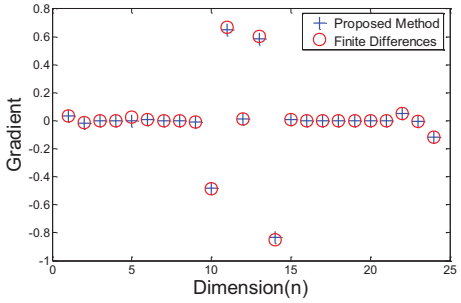

Fig. 2. Gradients obtained by traditional finite difference and our proposed approach based on random samplings.

C. Calculation of Sparse Gradient by Sparse Recovery

Since the performances of the circuits are not sensitive to some design parameters, the gradient is usually sparse. As shown in Fig. 2, most components of the gradient are zeros or nearly zeros. We can thus further employ the idea of sparse recovery to recover the sparse gradients even if the number of neighboring points t is smaller than the number of design variables d.

If $t < d$, equation (11) is underdetermined and the gradient cannot be solved directly from (11). However, if the gradient is sparse, the theory of sparse recovery can be employed to estimate the gradients by solving the following optimization problem [16], [17]

$$\begin{aligned} \min \ & \| grad \|_1 \\ \text{s.t.} \ & A \cdot grad = \vec{b}, \end{aligned} \tag{13}$$

where $\| grad \|_1$ means L_1-norm of $grad$. It aims to find the sparse solution of $A \cdot grad = \vec{b}$. The optimization problem (13) can be transformed to a linear programming problem and solved easily [18]. Note that compared with SPICE simulations, the time of solving (13) is negligible because the number of design variables, i.e., the dimension of $grad$ is 10-1000 in most designs.

We propose a heuristics to guarantee the accuracy of gradients obtained by sparse recovery. If t neighboring data points are found, a sparse gradient $grad_1$ is firstly recovered using these t data points by solving (13). Then, we generate a new neighboring point using the approach proposed in subsection III-B. This new data point and the corresponding simulation result is added to the set of the existing neighboring data points, and a new gradient $grad_2$ is obtained by solving (13) by using $t+1$ data points. If the difference of $grad_1$ and $grad_2$ is less than a predefined threshold η, the iterations stop. On the other hand, if the number of neighboring points equals to the number of design variables d finally, we solve (11) directly to get the gradient.

The proposed sparse recovery approach was used to calculate the gradient of the power consumption of an operational amplifier, which has 24 design variables. As shown in Fig. 3, the approximated gradient obtained by the proposed sparse recovery approach matches the gradient calculated by the finite difference well with 20 neighboring data points.

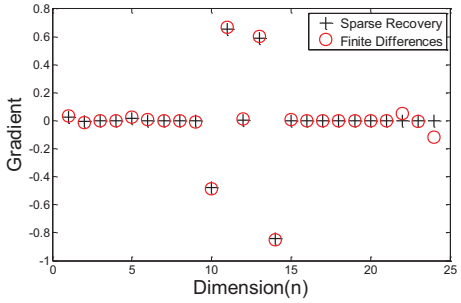

Fig. 3. Gradients obtained by traditional finite difference and our proposed approach based on sparse recovery.

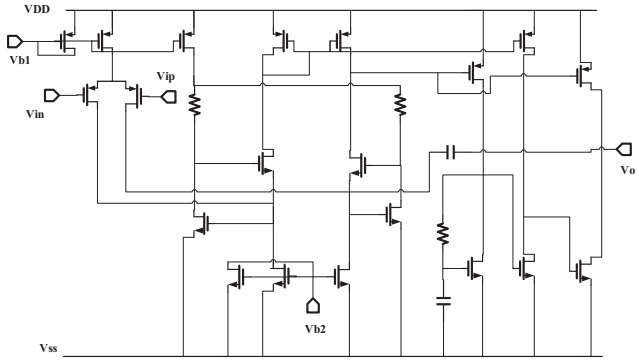

Fig. 4. 3-stage amplifier in [19]

IV. Experimental Results

In this section, we demonstrate the efficiency of our proposed method using several analog circuits. The optimization method is implemented in Matlab R2013a. HSPICE 2012.06 is used to simulate the circuits. The SQP algorithm implemented in Matlab is used as our baseline. All numerical experiments are performed on a workstation with 4 2GHz Intel Xeon CPU and 24GB memory.

A. Operational Amplifier

A 0.016-mm^2 144-μW three-stage amplifier which is capable of driving 1-to-15-nF capacitive load is described in [19]. This circuit is implemented in a standard 350nm CMOS technology. The circuit schematic is shown in Fig.4. In this experiment, we select the widths and lengths of transistors, resistance of resistors, bias current, and compensation capacitors as design variables. The number of the design variables is 24. The design objective and constraints of the circuit are listed in Table I. The multiple starting point methods using the traditional SQP algorithm and the SQP algorithm with our efficient gradient calculation method are used to optimize the circuit. The optimization results are then compared with manual design.

For this circuit, 5 starting points for subsequent local search are generated by heuristic methods. Since the time cost of SPICE simulations is significant compared with that of the optimization algorithm, the number of circuit simulations is utilized as the metric of computational cost. Table II summarizes the optimization results of the amplifier by different optimization methods. It can be observed that the optimization results meet the specifications [19]. The performance target I_q has a significant improvement over the manual design.

Compared with the traditional SQP algorithm, by recycling the previous simulation data, our proposed approach can reduce the number of simulations by 63%. On the other hand, the optimized target performance I_q is even better than that optimized by the traditional SQP algorithm. Although the traditional SQP using finite difference to derive the gradients, the gradients are not the exact gradients. The optimization result obtained by our proposed approach is better than the traditional SQP algorithm, which indicates that our proposed approach can achieve even higher accuracy in calculating gradients in this case.

B. Analog to Digital Converter

The second testcase is shown in Fig. 5. It is a 200M 10.5Bit analog-to-digital converter(ADC) with a ring amplifier in the pipeline front-end stage as depicted in Fig.6. The ADC is designed in a standard 65nm CMOS technology. To handle the complexity of ADC design and reduce the simulation time,

TABLE I
Design objective and constraints for the operational amplifier.

	Performance	optimization
Design Objective	I_q	minimize
Constraints	GBW	>0.92 MHz
	PM	>52.5 degree
	GM	>19.5 dB
	SR+	>0.18 V/μs
	SR-	>0.20 V/μs
	1% Ts+	<5.17 μs
	1% Ts-	<5.71 μs

TABLE II
Comparison of the performance and constraints of manual design and the optimized designs for the operational amplifier.

	Manual Design	MSP +SQP	MSP +Proposed
GBW(MHz)	0.92	0.95	0.93
PM(degree)	52.5	52.5	54.8
GM(dB)	19.5	24.2	20.0
SR+(V/μs)	0.18	0.21	0.23
SR-(V/μs)	0.20	0.35	0.37
1% Ts+(μs)	5.17	4.22	3.93
1% Ts-(μs)	5.71	3.67	3.45
Iq(μA)	**69.2**	**57.1**	**56.0**
# sim. runs	/	**2.45e+3(2.7x)**	**9.00e+2(1x)**

TABLE III
Design objective and constraints for the ADC circuit.

	Performance	optimization
Design Objective	ENOB	maximize
Constraints	SFDR	>66 dB

TABLE IV
Comparison of the performance and constraints of manual design and the optimized designs for the ADC circuit.

	Manual Design	MSP +SQP	MSP +Proposed
ENOB(bit)	**9.40**	**9.73**	**9.71**
SFDR(dB)	66.0	73.5	73.4
# sim. runs	/	**1.42e+3(1.5x)**	**9.53e+2(1x)**

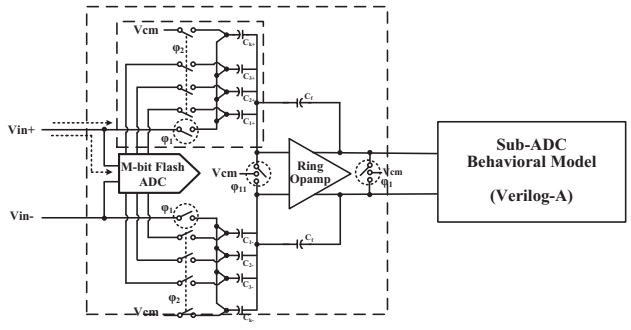

Fig. 5. Schematic of the 200M 10.5Bit Ring-amplifier-based ADC.

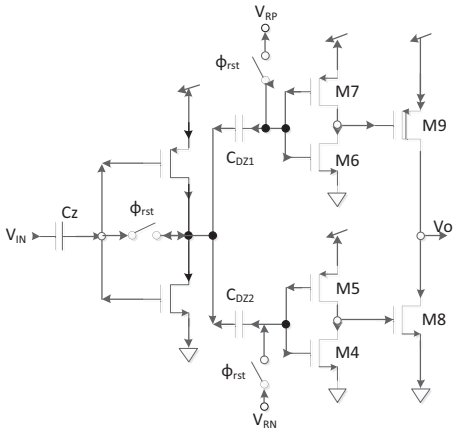

Fig. 6. Schematic of the ring operational amplifier of the ADC circuit.

the overall design is broken up into smaller functional blocks, and the sub-ADC is replaced with a behavior model. There are 42 circuit parameters in this design, including the widths of transistors and the resistance of some important resistors. As listed in Table III, two important performances of ADCs, including the efficient number of bit (ENOB) and spurious free dynamic range (SFDR), are selected as our target performance and constraint. In this experiment, 3 starting points are selected for local optimization by heuristic methods. The optimization results are showed in Table IV.

Compared with the traditional SQP algorithm, by recycling the previous simulation data, our proposed approach can reduce the number of simulations by 33%. The optimization result obtained by our proposed approach is close to that of the traditional SQP algorithm, which indicates that our proposed approach can achieve high accuracy in calculating gradients.

V. Conclusion

In this paper, we propose an efficient multiple starting point optimization method for automated analog circuit optimization. The gradients of the SQP-based local search are obtained by reusing the existing simulation data. The sparse recovery technique is further employed to reduce the number of simulations by exploiting the sparsity of the gradients. Experimental results have demonstrated that the proposed method can reduce the number of SPICE simulations up to 63% while achieving even better optimization, compared with the traditional multiple starting point method with standard SQP algorithm.

VI. Acknowledgement

This research is supported partly by National Natural Science Foundation of China (NSFC) research project 61474026, 61274032, 61125401, 61376040, 91330201, 61574046 and 61574044, partly by the National Basic Research Program of China under the grant 2011CB309701, partly by the Recruitment Program of Global Experts (the Thousand Talents Plan), partly supported by Chen Guang project supported by Shanghai Municipal Education Commission and Shanghai Education Development Foundation. This work is also partially supported by NSF grant 1115556.

References

[1] G. G. Gielen, R. Rutenbar *et al.*, "Computer-aided design of analog and mixed-signal integrated circuits," *Proc. IEEE*, vol. 88, no. 12, pp. 1825–1854, 2000.

[2] S. P. Boyd, T. H. Lee *et al.*, "Optimal design of a cmos op-amp via geometric programming," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 1–21, 2001.

[3] M. del Mar Hershenson, "Design of pipeline analog-to-digital converters via geometric programming," in *Proc. ACM/IEEE Int. Conf. Computer-Aided Design(ICCAD)*, 2002, pp. 317–324.

[4] Y. Wang, M. Orshansky, and C. Caramanis, "Enabling efficient analog synthesis by coupling sparse regression and polynomial optimization," in *Proc. ACM/EDAC/IEEE Design Automation Conf. (DAC)*, 2014, pp. 1–6.

[5] J. Yuan, N. Farhat, and J. Van der Spiegel, "Gbopcad: A synthesis tool for high-performance gain-boosted opamp design," *IEEE Trans. Circuits and Systems I*, vol. 52, no. 8, pp. 1535–1544, 2005.

[6] J. B. Grimbleby, "Automatic analogue circuit synthesis using genetic algorithms," in *IEEE Proc. Circuits, Devices and Systems*, vol. 147, no. 6, 2000, pp. 319–323.

[7] M. Kotti, B. Benhala, M. Fakhfakh, A. Ahaitouf, B. Benlahbib, M. Loulou, and A. Mecheqrane, "Comparison between pso and aco techniques for analog circuit performance optimization," in *Proc. ICM*, 2011.

[8] G. Huang, L. Qian, S. Saibua, D. Zhou, and X. Zeng, "An efficient optimization based method to evaluate the drv of sram cells," *IEEE Trans. Circuits and Systems I*, vol. 60, no. 6, pp. 1511–1520, 2013.

[9] M. Li, D. Zhou, S.-G. Wang, and X. Zeng, "Fmssqp: An efficient global optimization tool for the robust design of rail-to-rail op-amp," in *Proc. IEEE Int. Conf. ASIC (ASICON)*, 2013, pp. 1–4.

[10] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta numerica*, vol. 4, pp. 1–51, 1995.

[11] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[12] P. E. Gill, W. Murray, and M. H. Wright, "Practical optimization," 1981.

[13] M. Powell, "Variable metric methods for constrained optimization," in *Mathematical Programming The State of the Art*. Springer, 1983, pp. 288–311.

[14] W. Hock and K. Schittkowski, "A comparative performance evaluation of 27 nonlinear programming codes," *Computing*, vol. 30, no. 4, pp. 335–358, 1983.

[15] M. J. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in *Numerical analysis*. Springer, 1978, pp. 144–157.

[16] E. J. Candès *et al.*, "Compressive sampling," in *Proc. International Congress of Mathematicians*, vol. 3, 2006, pp. 1433–1452.

[17] D. L. Donoho, "Compressed sensing," *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[18] X. Li, R. R. Rutenbar, and R. D. Blanton, "Virtual probe: a statistically optimal framework for minimum-cost silicon characterization of nanoscale integrated circuits," in *Proc. ACM/IEEE Int. Conf. Computer-Aided Design(ICCAD)*, 2009, pp. 433–440.

[19] Z. Yan, P.-I. Mak, M. Law, and R. P. Martins, "A 0.016-mm^2 144-μw three-stage amplifier capable of driving 1-to-15 nf capacitive load with 0.95-mhz gbw," *IEEE J. Solid-State Circuits*, vol. 48, no. 2, pp. 527–540, 2013.