An Efficient Bayesian Optimization Approach for Analog Circuit Synthesis via Sparse Gaussian Process Modeling

Biao He[1], Shuhan Zhang[1], Fan Yang[1*], Changhao Yan[1], Dian Zhou[2] and Xuan Zeng[1*]
[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*—Bayesian optimization with Gaussian Process (GP) models has been proposed for analog synthesis since it is efficient for the optimizations of expensive black-box functions. However, the computational cost for training and prediction of Gaussian process models are $O(N^3)$ and $O(N^2)$, respectively, where N is the number of data points. The overhead of the Gaussian process modeling would not be negligible as N is relatively large. Recently, a Bayesian optimization approach using neural network has been proposed to address this problem. It reduces the computational cost of training and prediction of Gaussian process models to $O(N)$ and $O(1)$, respectively. However, reducing the infinite-dimensional kernel to finite-dimensional kernel using neural network mapping would weaken the characterization ability of Gaussian process. In this paper, we propose a novel Bayesian optimization approach using Sparse Pseudo-input Gaussian Process (SPGP). The idea is to use $M < N$ so-called inducing points to build a sparse Gaussian process model to approximate the conventional exact Gaussian process model. Without the need to sacrifice the modeling ability of the surrogate model, it also reduces the computational cost of both training and prediction to $O(N)$ and $O(1)$, respectively. Several experiments were provided to demonstrate the efficiency of the proposed approach.**

***Index Terms*—Bayesian optimization, Sparse Gaussian Process, Analog Circuit Synthesis**

I. Introduction

Analog circuit design automation attracts more and more research interests over the past decades. Generally, the analog circuit design automation consists of two major components, i.e., topology selection and device sizing. We focus on the device sizing problem in this paper. Due to the serve process variations and more and more complicated device structures, the device sizing problem becomes more and more complicated.

The conventional device sizing approaches can be classified into two categories: model-based and simulation-based approaches. Model-based approaches build mathematical models of circuits firstly. The optimization is conducted based on the built mathematical models. A typical model-based method is geometric programming based method [1], which uses posynomials to model the circuits. The circuit optimization problem can then be transformed to convex optimize problem, and thus the global optimum could be easily found. General polynomials can also be used to model the circuits, and Semidefinite-Programming (SDP) relaxations could be used to transform the polynomial programming problem to a convex programming problem [2]. The main advantage of the model-based approach is that once the models were built, the computational cost would be much lower than the simulation-based approach. However, the global optima of the constructed model may deviate from the real global optima of the circuit performance. And a globally accurate model is not computationally affordable.

On the contrary to model-based approaches, simulation-based approaches view the performances of circuits as black-box functions. The function values are obtained by directly simulating the circuits. Many well-known heuristic optimization methods such as simulated annealing (SA) [3], particle swarm intelligence (PSO) algorithm [4], evolutionary algorithm [5], and multi-start optimization algorithm [6] [7] were employed to efficiently explore the design space and avoid falling into the local optima. Although the simulations were invoked on-the-fly, the convergence rate of the heuristic methods were relatively low.

Surrogate-model based optimization approaches have been proposed recently [8] [9] [10], which combine the model-based and simulation-based approaches. In contrast to the models built beforehand in model-based approaches, the surrogate models are built on-the-fly during the optimization procedure. The surrogate models are updated once new data points are obtained. The number of circuit simulations can thus be greatly reduced. The Gaussian Process Regression (GPR) is the most widely used surrogate model, because it provides both predictive means and uncertainty estimations. In GASPAD [11], the GPR surrogate-model combined with evolutionary algorithm is proposed for microwave circuit synthesis. In [12], GPR based Bayesian optimization approach has been applied to analog circuit synthesis. In [12], the weighted Expected Improvement (wEI) is leveraged as the acquisition function to facilitate the optimization procedure.

Although the surrogate model is able to greatly reduce the number of circuit simulations during the optimization procedure, the overhead of construction and evaluation of the surrogate models would not be negligible when the dataset becomes relatively large. The computational cost for training and prediction of Gaussian process models are $O(N^3)$ and $O(N^2)$, respectively, where N is the number of data points. Recently, a Bayesian optimization approach using neural network has been proposed to address this problem [13]. It reduces the computational cost of training and prediction of Gaussian process models to $O(N)$ and $O(1)$, respectively. In [13], the GPR model is derived from a weight space view. The original data is mapped to a finite-dimensional feature space using neural network, and the GPR model is then derived through Bayesian linear regression. It can be regard as a

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

degenerate Gaussian process with finite-dimensional kernel function, which would weaken the modeling ability of the surrogate model.

To chart a way out of this dilemma, we propose a novel Bayesian optimization approach using Sparse Pseudo-input Gaussian process (SPGP) [14] in this paper. The idea is to generate M so-called inducing points and define the covariance matrix of the M inducing points to approximate the covariance matrix of the N data points. The proposed approach can also reduce the computational cost of training and prediction to $O(N)$ and $O(1)$, respectively. The sparse Gaussian process regression models are widely used for building regression models with large datasets. However, they are rarely used in Bayesian optimization as surrogate model. An adaptive strategy for selecting inducing points is designed to improve the modeling ability and accelerate the optimization procedure. Without sacrificing the modeling ability of the surrogate model, the kernel of the proposed approach remains infinite-dimensional and retain most of data information. It would be efficient whilst still preserving the desirable properties of the exact GP. Several experiments were provided to demonstrate the efficiency of the proposed approach.

The rest of the paper is organized as follows. In section II, we will review the background of analog circuit sizing and Bayesian optimization based on the GPR model. In section III, we will present our proposed Bayesian optimization using Sparse Pseudo-input Gaussian process. The efficiency of our proposed approach will be demonstrated with two real-world circuits in section IV. And we conclude the paper in section V.

II. Background

In this section, we will review the background of analog circuit sizing problem and the GPR-based Bayesian optimization approach.

A. Problem Formulation of Analog Circuit Sizing

The analog circuit sizing problem can be formulated as a constrained optimization problem:

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{s.t.} & c_i(\boldsymbol{x}) < 0 \\ & \forall i \in 1 \dots N_c, \end{array} \tag{1}$$

where $\boldsymbol{x} \in R^d$ represents d design variables, $f(\boldsymbol{x})$ denotes the Figure of Merit (FOM) of the analog circuit, and $c_i(\boldsymbol{x})$ is the i-th constraints of the N_c constraint functions that need to be satisfied.

B. Bayesian Optimization

Bayesian Optimization framework has emerged as a powerful solution for black-box function that is costly and noisy [15]. Bayesian optimization incorporated prior belief about the objective function and updated the prior with samples drawn from the simulations to get a posterior that better approximates the black-box function. The probabilistic surrogate model, which provides prediction and uncertainty, is used to approximate the function. An acquisition function, which is used to balance the exploration and exploitation, is utilized in the Bayesian optimization to guide the optimization [15]. By maximizing the acquisition function value, a sequence of query point can be selected to facilitate the optimization procedure. The value of the black-box function at the new location can be obtained by simulation and added to the training set. The surrogate model will then be updated and the optimization procedure continues. Gaussian process regression model is a widely used surrogate model in Bayesian optimization.

C. Gaussian Process Regression

Consider a scalar black-box function $f(\boldsymbol{x})$, where $\boldsymbol{x} \in R^d$ is the input vector. Given a set of N training data $(\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ denotes the input and $\boldsymbol{y} = \{y_1, \cdots, y_N\}$ represents the observations drawn from the black-box function $f(\boldsymbol{x})$. $\boldsymbol{y}$ would be different from $f(\boldsymbol{x})$ if observation noise is considered.

Gaussian process regression model assumes that $f(\boldsymbol{x})$ can be regarded as a Gaussian process, which means any finite samples (e.g., N samples) drawn from the black-box function $f(\boldsymbol{x})$ follow a joint Gaussian distribution $\boldsymbol{f} \sim N(\mu, \boldsymbol{K})$, where $\boldsymbol{\mu} = \{\mu(\boldsymbol{x}_1), \cdots, \mu(\boldsymbol{x}_N)\}$ is the mean vector and $\boldsymbol{K} \in R^{N \times N}$ is the covariance matrix. The mean function $\mu(\boldsymbol{x})$ could be any function, and the element $K_{i,j}$ of the matrix $\boldsymbol{K}$ is drawn from a kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. In this paper, we follow the suggestion of [12] and set the mean function as $m(\boldsymbol{x}) = \mu_0$ and the kernel function as:

$$k_{SE}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\Big(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_j)\Big), \tag{2}$$

where μ_0, σ_f^2 and $\Lambda = \text{diag}(l_1^2, l_2^2, \dots, l_d^2)$ are hyper-parameters to optimize. By using the assumption that N training data points follow joint Gaussian distribution, these hyper-parameters could be obtained by maximum likelihood estimation [12].

Gaussian process regression aims to build a regression model to predict the value y^* of the black-box function at a new point $\boldsymbol{x}^*$. With the assumption of Gaussian process, the vector $\boldsymbol{y}$ and y_* follow a known joint Gaussian distribution

$$\begin{bmatrix} y_* \\ \boldsymbol{y} \end{bmatrix} \sim N\left(\begin{bmatrix} m(\boldsymbol{x}_*) \\ \boldsymbol{m} \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}_*, \boldsymbol{x}_*) & k(\boldsymbol{x}_*, \boldsymbol{X}) \\ k(\boldsymbol{X}, \boldsymbol{x}_*) & \boldsymbol{K}_N \end{bmatrix}\right). \tag{3}$$

A Gaussian noise $\epsilon \sim N(0, \sigma_n^2)$ can be introduced to model the observation noise of $f(x)$ [12]. The prediction of the Gaussian process regression model is a Gaussian distribution $y^* \sim N(\mu(\boldsymbol{x}^*), \sigma^2(\boldsymbol{x}^*))$ [12], where

$$\begin{cases} \mu_{y_*|\boldsymbol{y}} &= m(\boldsymbol{x}_*) + k(\boldsymbol{x}_*, \boldsymbol{X})(\boldsymbol{K}_N + \sigma_n^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{m}) \\ \sigma^2_{y_*|\boldsymbol{y}} &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{x}_*, \boldsymbol{X})(\boldsymbol{K}_N + \sigma_n^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*). \end{cases} \tag{4}$$

D. Acquisition Function

Acquisition function is used to balance exploration and exploitation in Bayesian optimization. A popular function is the Expected improvement (EI) function [12]

$$\text{EI}(\boldsymbol{x}) \quad = \quad \mathbb{E}[\max(0, y - \tau)], \tag{5}$$

where $\mathbb{E}$ means the mathematical expectation, and τ means the current minimal objective value. $\text{EI}(\boldsymbol{x})$ thus denotes the expected improvement with the Gaussian process regression model.

To handle the constraints, the expected improvement can be weighted by the probability of the constraints are met [12]

$$\text{wEI}(\boldsymbol{x}) \quad = \quad \text{EI}(\boldsymbol{x}) \prod_{i=1}^{N_c} \text{PF}(c_i(\boldsymbol{x}) < 0), \tag{6}$$

where the $\mathrm{PF}_i(\boldsymbol{x})$ indicates the probability of the i-th constraint satisfied.

III. Bayesian Optimization with Sparse Pseudo-input Gaussian Process (SPGP)

The computational cost for training and prediction of Gaussian process models are $O(N^3)$ and $O(N^2)$, respectively [13]. The overhead of the Gaussian process modeling would not be negligible as the number of data points N becomes relatively large. This is mainly because an inverse of a $N \times N$ dense matrix $\boldsymbol{K}_N + \sigma_n^2 \boldsymbol{I}$ is involved in the training process of the Gaussian process regression, as shown in (4).

An intuition to reduce the computational cost is to find a low-rank approximation of the full-rank matrix. Such an intuition leads to a series of so-called sparse Gaussian process models. In [16], a subset of M inducing points are selected directly from the training points, and the full-rank matrix $\boldsymbol{K}_N$ is approximated by a M-rank matrix derived from the selected M data points. However, the selection of the inducing points are complicated and the rest of the training data are not fully utilized. In [17], instead of selecting inducing points directly from the raw training data, a set of pseudo inducing points were defined, and the pseudo inducing points are generated by maximum likelihood estimation with all the training data points. These sparse Gaussian process regression models are widely used for building regression models with large datasets. However, they are rarely used in Bayesian optimization as surrogate model.

In this section, we take the sparse pseudo-input Gaussian process model in [17] as our surrogate model for Bayesian optimization. It reduces the computational cost of training and prediction of Gaussian process models to $O(N)$ and $O(1)$, respectively. We propose an efficient strategy to determine the inducing points, which could effectively accelerate the optimization procedure.

A. Sparse Pseudo-input Gaussian Process

In order to find a sparse approximation of the original covariance matrix, a set of M latent variables $\boldsymbol{u} = [u_1, \cdots, u_m]^T$ is introduced, where M is much smaller than the number of training data points N. These latent variables are also referred as inducing inputs, inducing variables or pseudo inputs [17]. These latent variables are samples of a Gaussian process at a set of input locations $\boldsymbol{X}_u = \{\boldsymbol{x}_1^u, \cdots, \boldsymbol{x}_M^u\}$, which means that $[u_1, \cdots, u_m]$ also follows a joint Gaussian distribution. Without loss of generality, we assume that

$$\boldsymbol{u} \sim N(\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}), \tag{7}$$

where $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}} \in R^{M \times M}$ denotes the covariance matrix of the latent variables. The location of the pseudo input points $\boldsymbol{X}_u$ and the parameters of this joint Gaussian distribution are taken as hyper-parameters, which would be learned from the training data as discussed later.

We aim to predict the output y^* at x^* with the know training data $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ and $\boldsymbol{f} = \{f_1, \cdots, f_N\}$. Note that we use $\boldsymbol{y} = \{y_1, \cdots, y_N\}$ to represent the observations drawn from the black-box function. $\boldsymbol{y}$ would be different from $\boldsymbol{f}$ if noises is introduced.

We follow the idea in [17], where the outputs of the black-box functions as well as the observations $\boldsymbol{y}$ are assumed to be conditionally independent given the latent variables $\boldsymbol{u}$, as shown in Fig. 1. The name inducing comes from the fact that f_i and f_j communicate through $\boldsymbol{u}$. With this assumption, we have

$$p(f_i, f_j|\boldsymbol{u}) = p(f_i|\boldsymbol{u})p(f_j|\boldsymbol{u}), \forall i \neq j,$$

and

$$p(y_i, y_j|\boldsymbol{u}) = p(y_i|\boldsymbol{u})p(y_j|\boldsymbol{u}), \forall i \neq j.$$

By taking the latent variables or pseudo inputs as training data points, we can get the distributions of y_i (or y^*) according to (4) by setting the mean function and covariance matrix to those in (7), i.e.,

$$p(y_i|\boldsymbol{u}) = N(\boldsymbol{K}_{\boldsymbol{x_i},\boldsymbol{u}}\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}^{-1}\boldsymbol{u}, \mathrm{diag}(\boldsymbol{K}_{\boldsymbol{x}_i,\boldsymbol{x}_i} - \boldsymbol{K}_{\boldsymbol{x}_i,\boldsymbol{u}}\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{u}}^{-1}\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{x}_i}) + \sigma_n^2), \tag{8}$$

where $\boldsymbol{K}_{\boldsymbol{x_i},\boldsymbol{u}} \in R^{N \times M}$ denotes the covariance between f_i and the inducing variables $\boldsymbol{u}$, and $\boldsymbol{K}_{\boldsymbol{u},\boldsymbol{x_i}}$ is the transpose of $\boldsymbol{K}_{\boldsymbol{x_i},\boldsymbol{u}}$. σ_n is introduced to model the observation noise for training data. The distribution of y^* can be obtained similarly by replacing $\boldsymbol{x}_i$ with $\boldsymbol{x}^*$.

According to the conditionally independent assumptions, we have

$$p(\boldsymbol{y}|\boldsymbol{u}) = \prod_{\mathrm{i}=1}^{N} p(y_i|\boldsymbol{u}). \tag{9}$$

By integrating (9) over the inducing variable $\boldsymbol{u}$, we could get the marginal likelihood of $\boldsymbol{y}$

$$p(\boldsymbol{y}) \quad = \quad \int p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{u})\mathrm{d}\boldsymbol{u}. \tag{10}$$

The hyper-parameters and the locations of inducing variables $\boldsymbol{X}_{\boldsymbol{u}}$ could be obtained by maximizing the marginal likelihood $p(\boldsymbol{y})$.

The prediction of y^* is also a distribution, which could be expressed as conditional distribution $p(y^*|\boldsymbol{y})$. It can be obtained through Bayes' theorem

$$p(y_*|\boldsymbol{y}) = \frac{p(y_*, \boldsymbol{y})}{p(\boldsymbol{y})} = \mathrm{N}(\mu_*, \sigma_*^2), \tag{11}$$

where $p(y_*, \boldsymbol{y})$ and $p(\boldsymbol{y})$ could be obtained similarly from (10). For simplicity, we show results here

$$\begin{aligned} \mu_* &= \boldsymbol{Q}_{\boldsymbol{x}_*,\boldsymbol{X}}[\boldsymbol{Q}_N + \boldsymbol{\Gamma}]^{-1}\boldsymbol{y} \\ \sigma_* &= \boldsymbol{K}_{\boldsymbol{x}_*,\boldsymbol{x}_*} - \boldsymbol{Q}_{\boldsymbol{x}_*,\boldsymbol{X}}[\boldsymbol{Q}_N + \boldsymbol{\Gamma}]^{-1}\boldsymbol{Q}_{\boldsymbol{X},\boldsymbol{x}_*} + \sigma_n^2, \end{aligned} \tag{12}$$

where $\boldsymbol{Q}_N \equiv \boldsymbol{K}_{\boldsymbol{X},\boldsymbol{X}_u}\boldsymbol{K}_{\boldsymbol{X}_u,\boldsymbol{X}_u}^{-1}\boldsymbol{K}_{\boldsymbol{X}_u,\boldsymbol{X}}$ and $\boldsymbol{K}_{\boldsymbol{x}_*,\boldsymbol{X}_u}$ denotes the covariance between predict point $\boldsymbol{x}_*$ and inducing point $\boldsymbol{X}_u$, and $\boldsymbol{\Gamma} = \mathrm{diag}(\boldsymbol{K}_{\boldsymbol{X},\boldsymbol{X}} - \boldsymbol{Q}_{\boldsymbol{X},\boldsymbol{X}}) + \sigma_n^2\boldsymbol{I}$.

Equation (12) is similar to the prediction of the exact Gaussian process as shown in (4). By simply substituting the full-rank covariance matrix with a low-rank one $\boldsymbol{Q}' = \boldsymbol{Q}_N + \boldsymbol{\Gamma}$ in (4), we can get prediction of the sparse Gaussian process model.

B. Inducing-Point Selection Strategy

In [16], the inducing points are selected explicitly from the real training data points, which is a complicated combinatorial optimization problem. In Sparse Pseudo-input Gaussian Process [17], the inducing points are selected by maximizing the marginal likelihood (10). For numerical considerations, we

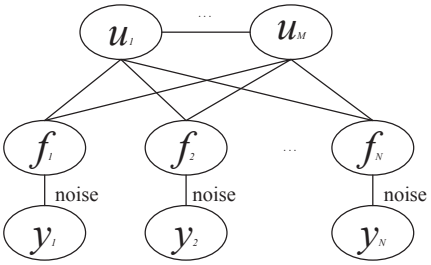

Fig. 1. Conditionally Independent Assumption.

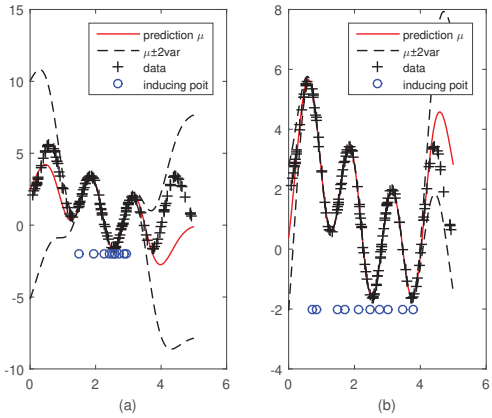

Fig. 2. (a) Prediction with randomly initialized inducing points, and (b) Prediction with inducing points after optimization.

minimize a negative log-likelihood function instead, which can be expressed as

$$\mathrm{L}(\boldsymbol{\theta}, \boldsymbol{X_u}) = log|\boldsymbol{Q}_N + \boldsymbol{\Gamma}| + \boldsymbol{y}^T(\boldsymbol{\Gamma} + \boldsymbol{\Gamma})^{-1}\boldsymbol{y} + Nlog2\pi, \quad (13)$$

which could be efficiently solved by gradient-based algorithms, e.g., L-BFGS algorithm [18].

Note that $\mathrm{L}(\boldsymbol{\theta}, \boldsymbol{X_u})$ is generally a non-convex function. It may have many local minima, which leads to different fitting results. In our proposed approach, the regression model is taken as surrogate model for optimization. We should carefully select the inducing points to accelerate the optimization procedure.

We use a one-dimensional example as shown in Fig. 2 to study the affects of the selection of inducing points. In Fig. 2(a), we show the prediction result with randomly initialized inducing points before minimizing (13). In Fig. 2(b), we shown the prediction result with the inducing points which minimizing (13). From Fig. 2, we have the following observations. Firstly, the prediction accuracy in a region is higher if more inducing points are located in this region. Secondly, the inducing points would spread after optimization. Nevertheless, we found in our experiments that the distribution of the inducing points after optimization are mainly determined by the initial values, especially for high-dimensional problems.

Inspired by these observations, we propose an efficient strategy to select the initial values for minimizing (13), which could significantly accelerate the optimization procedure. In the beginning stage of the optimization, we need to explore more design space. Thus, during the optimization of (13), we initialize the inducing points randomly. As the Bayesion optimization continues, a set of specific regions would be explored. Thus, we will initialize the inducing points as M latest query points during Bayesian optimization, which would located in the specific regions to be explored. After minimizing (13), the optimized inducing points would spread to guarantee the exploration. Nevertheless, it would also ensure the prediction accuracy over the specific regions, which could improve the prediction accuracy in the regions to be explored and thus efficiently accelerate the optimization procedure. As shown in Fig. 3, our proposed strategy could achieve much higher convergence speed compared with random initialization for the optimization of a three-stage amplifier.

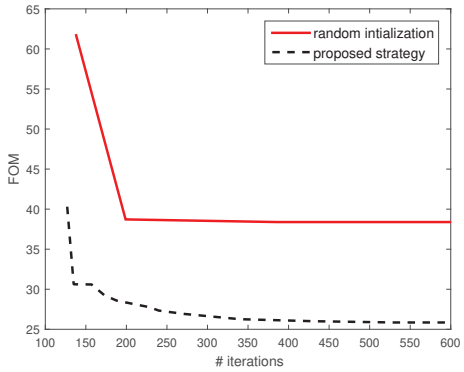

Fig. 3. Comparison of convergence speed for random initialization and our proposed strategy.

C. Computational Complexity Analysis

We will analyze the computational cost of training and prediction of SPGP model in this subsection. In this subsection, N denotes the number of training points, and M is the number of inducing points, which can be viewed as a constant.

For both training (13) and prediction (12) of the SPGP model, the computational cost is dominated by the inverse of the $N \times N$ matrix $\boldsymbol{Q}' = \boldsymbol{Q}_N + \boldsymbol{\Gamma}$. Different from the exact GP, $\boldsymbol{Q}_N + \boldsymbol{\Gamma}$ is a low-rank matrix, the computational cost can thus be reduced. According to Woodbury formula [19], the inverse of the low-rank matrix can be written as

$$[\boldsymbol{Q}_N + \boldsymbol{\Gamma}]^{-1} = \boldsymbol{\Gamma}^{-1} - \boldsymbol{\Gamma}^{-1}\boldsymbol{K}_{\boldsymbol{X_u},\boldsymbol{X}}\boldsymbol{B}^{-1}\boldsymbol{K}_{\boldsymbol{X},\boldsymbol{X_u}}\boldsymbol{\Gamma}^{-1}, \quad (14)$$

where $\boldsymbol{B} = \boldsymbol{K_u} + \boldsymbol{K}_{\boldsymbol{X_u},\boldsymbol{X}}\boldsymbol{\Gamma}^{-1}\boldsymbol{K}_{\boldsymbol{X},\boldsymbol{X_u}}$. The computational cost for the calculation of $\boldsymbol{B}$ is $O(NM^2)$. The cost for $\boldsymbol{\Gamma}^{-1}$ is $O(N)$, since it is a diagonal matrix. So the computational cost of calculating $[\boldsymbol{Q}_N + \boldsymbol{\Gamma}]^{-1}$ is approximately $O(NM^2)$.

Since the parameters of the sparse kernel varies during the training stage, the calculation of $\boldsymbol{Q}'$ is required for each iteration, which leads the complexity for training to $O(NM^2)$. Once the training is done, $[\boldsymbol{Q}_N + \boldsymbol{\Gamma}]^{-1}$ is fixed. We could precompute it before predictions. Then, the computational cost for prediction is $O(M^2)$, which can be viewed as constant computational cost.

IV. Experimental Results

In this section, we will demonstrate the efficiency of our proposed approach by two real circuits. Our proposed ap-

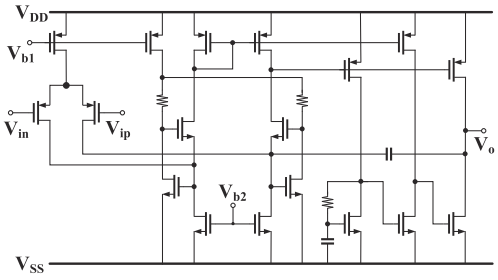

Fig. 4. Schematic of the three-stage amplifier [20].

proach is implemented in C++. We compared our proposed approach with the existing methods including Bayeisan optimization using Neural Netowrk (BONN for short) [13], WEIBO [12], GASPAD [11], and DE [5]. The two test-cases were provided by the authors of [12]. The results of WEIBO [12], GASPAD [11], and DE [5] were taken directly from [12]. We tested BONN [13] and our proposed approach on a Linux workstation with 2 Intel Xeon CPUs and 128G memory.

A. A Three-stage Amplifier

The first case is a three-stage amplifier [20] which is implemented in a TSMC 0.35um technology. Its schematic is shown in Figure 4. There are 24 design variables in this circuit. The design variables include the sizes of transistors, the resistance, capacitance, and the bias current.

We aim to minimize the static current Iq with an acceptable DC gain, unit gain frequency UGF, gain margin GM, phase margin, rising slew rate SRR, and falling slew rate. The specification of design is listed in (15).

$$\begin{array}{lllll} \text{minimize} & Iq & & & \\ \text{s.t.} & GAIN & > & 100\,\text{dB} & \\ & UGF & > & 0.92\,\text{MHz} & \\ & PM & > & 52.5\,^\circ & \\ & GM & > & 19.5\,\text{dB} & \\ & SRR & > & 0.18\,\text{V}/\mu\text{s} & \\ & SRF & > & 0.2\,\text{V}/\mu\text{s}. & \end{array} \tag{15}$$

To eliminate the effects of random fluctuations, we ran all the optimization algorithms 12 times. We have set the maximum allowed number of simulations as 800 for our proposed algorithms and 900 for BONN [13]. The number of inducing points is set to 50 for our proposed approach. $\boldsymbol{X}_u$ is initialized here as the latest collected data points to accelerate the optimization process. For the other algorithms, the maximum number of simulations is limited to 3000 as described in [12].

The optimization results are shown in Table I. The results of WEIBO [12], GASPAD [11] and DE [5] were taken from [12] directly. Here '#success' means that we have found a feasible solution. As we can see, all algorithms can find feasible solution, but our proposed methods give the best result on average. Compare with WEIBO [12] and BONN [13], we have reduced the number of simulations by 35.9% and 35.6%, respectively. Moreover, compared with the best results of WEIBO [12] and BONN [13], the best Iq of our proposed approach is reduced by 0.84 and 1.7, respectively.

TABLE I
THE OPTIMIZATION RESULTS OF THE THREE-STAGE AMPLIFIER. THE RESULTS OF WEIBO, GASPAD AND DE COME FROM [12]

Algo	Ours	BONN	WEIBO	GASPAD	DE
GAIN/dB	100.85	101.57	100.57	101.6	102.73
UGF/MHz	0.922	0.936	0.922	0.924	0.956
PM/°	52.63	53.38	52.52	52.60	54.62
GM/dB	19.69	19.92	19.76	20.05	20.62
SR+(V/μs)	0.25	0.21	0.20	0.24	0.21
SR-(V/μs)	0.46	0.51	0.46	0.55	0.54
Iq(mean)/μA	**27.08**	29.34	27.87	31.55	41.26
Iq(median)/μA	**27.00**	28.80	27.65	31.47	40.70
Iq(best)/μA	**25.86**	27.56	26.70	28.36	37.32
Iq(worst)/μA	**29.26**	32.44	29.30	34.47	46.09
Avg. # Sim	**515**	805	798	2744	2400
# Success	12/12	12/12	12/12	12/12	12/12

Our Bayesian optimization approach with sparse Gaussian process model could obtain better results than the WEIBO [12] with exact Gaussian process. This is mainly because the sparse Gaussian process model could efficiently prevent the surrogate model from overfitting.

Our proposed approach could also obtain better results than BONN [13]. This is mainly because the kernel of the sparse Gaussian process model is still infinite-dimensional. The kernel used in BONN [13] is finite-dimensional, which would weaken the modeling abilities of Gaussian process models.

B. Voltage-Controlled Oscillator

The second test case is a Voltage Controlled Oscillator (VCO), which is implemented in a SMIC 40nm process. It consists of 131 transistors, and the circuit is shown in Figure 5. There are 20 design variables in this design. It considers a total of 10 PVT corners. The simulation of VCO itself is time-consuming.

For this circuit, we aim to minimize the variation of the output frequency, with the constraints of the linearity and the slope of the frequency-voltage curve. The specifications are listed as (16).

$$\begin{array}{lllll} \text{minimize} & FOM & & & \\ \text{s.t.} & k_{min} & > & 300\,\text{MHz/V}, & \\ & R_{min} & > & 0.95. & \end{array} \tag{16}$$

where

$$\begin{cases} k_{min} &= \min_{\forall PVT}(k), \\ R_{min} &= \min_{\forall PVT}(R), \\ FOM &= 10 \times (\dfrac{\sigma(f_l)}{\mu(f_l)} + \dfrac{\sigma(f_h)}{\mu(f_h)}). \end{cases} \tag{17}$$

In (17), k_{min} is the minimum of measured slopes k in all corners; R_{min} is the minimum of the measured correlations. $\mu(\cdot)$ and $\sigma(\cdot)$ in the Figure of Merit (FOM) expression represent the standard deviation and mean function, respectively. The FOM consists of the standard deviation and the mean of the f_l and f_h, which are used to model the variation of the output frequency.

The optimization results are shown in Table II. In this experiment, we also ran BONN and our proposed approach 12 times. The results of WEIBO [12], GASPAD [11] and DE [5] were taken from [12] directly. The maximum number of simulations is limited to 300 for our proposed approach and

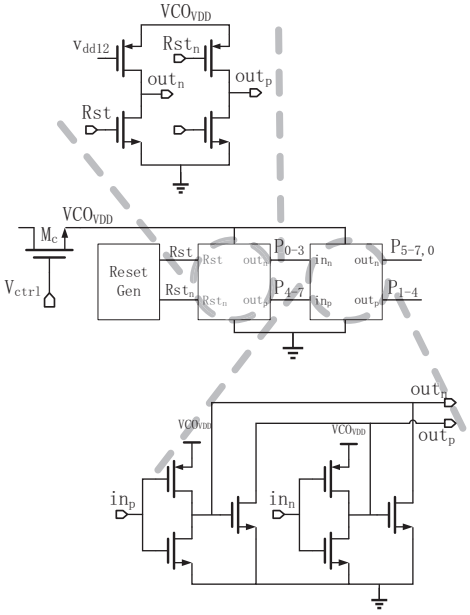

Fig. 5. Schematic of the voltage controlled oscillator [12].

TABLE II
THE OPTIMIZATION RESULTS OF THE VCO. THE RESULTS OF WEIBO, GASPAD AND DE COME FROM [12]

Algo	Ours	BONN	WEIBO	GASPAD	DE
k(MHz/V)	336	336	355	368	368
r	0.996	0.996	0.997	0.998	0.998
FOM(mean)	**3.78**	3.88	3.81	3.90	4.08
FOM(median)	**3.78**	3.89	3.82	3.88	4.08
FOM(best)	**3.75**	3.84	3.78	3.82	3.96
FOM(worst)	**3.82**	3.90	3.82	4.01	4.21
Avg. # Sim	**136**	278	181	917	577
# Success	12/12	12/12	12/12	12/12	12/12

WEIBO [12]. For BONN [13], it is set to 350. For all the other approaches, the maximum number of simulations is set to 1000 as described in [12]. As shown in II, our proposed algorithm can find the best FOM with the lest number of simulations. Compare with BONN [13] and WEIBO [12], we reduced the number of simulations by 51.4% and 25.2%, respectively.

V. CONCLUSION

In this paper, we propose a sparse pseudo-input Gaussian process based Bayesian optimization approach for analog circuit synthesis. The proposed approach can reduce the computational cost of training and prediction to $O(N)$ and $O(1)$, respectively, while retaining the infinite-dimensional kernel characteristics as Bayesion optimization approach with exact Gaussian process model. The efficiency of the proposed approach has been verified with two real-world circuits.

ACKNOWLEDGMENTS

This research is supported partly by the National Major Science and Technology Special Project of China (2017ZX01028101-003), partly by National Natural Science Foundation of China (NSFC) 61822402, 61774045, 61574046, 61974032, 61574044, 61674042 and 61929102.

REFERENCES

[1] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and engineering*, 8(1):67, 2007.
[2] Ye Wang, Michael Orshansky, and Constantine Caramanis. Enabling efficient analog synthesis by coupling sparse regression and polynomial optimization. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2014.
[3] Rodney Phelps, Michael Krasnicki, Rob A Rutenbar, L Richard Carley, and James R Hellums. Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(6):703–717, 2000.
[4] RA Vural and T Yildirim. Analog circuit sizing via swarm intelligence. *AEU-International journal of electronics and communications*, 66(9):732–740, 2012.
[5] Bo Liu, Yan Wang, Zhiping Yu, Leibo Liu, Miao Li, Zheng Wang, Jing Lu, and Francisco V Fernández. Analog circuit optimization system based on hybrid evolutionary algorithms. *INTEGRATION, the VLSI journal*, 42(2):137–148, 2009.
[6] Guanming Huang, Liuxi Qian, Siwat Saibua, Dian Zhou, and Xuan Zeng. An efficient optimization based method to evaluate the drv of sram cells. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(6):1511–1520, 2013.
[7] Bo Peng, Fan Yang, Changhao Yan, Xuan Zeng, and Dian Zhou. Efficient multiple starting point optimization for automated analog circuit optimization via recycling simulation data. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pages 1417–1422. EDA Consortium, 2016.
[8] Bo Liu, Ying He, Patrick Reynaert, and Georges Gielen. Global optimization of integrated transformers for high frequency microwave circuits using a gaussian process based surrogate model. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011.
[9] Shuhan Zhang, Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, Xuan Zeng, and Xiangdong Hu. An efficient multi-fidelity bayesian optimization approach for analog circuit synthesis. 2019.
[10] Bo Liu, Noël Deferm, Dixian Zhao, Patrick Reynaert, and Georges GE Gielen. An efficient high-frequency linear rf amplifier synthesis method based on evolutionary computation and machine learning techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):981–993, 2012.
[11] Bo Liu, Dixian Zhao, Patrick Reynaert, and Georges GE Gielen. Gaspad: A general and efficient mm-wave integrated circuit synthesis method based on surrogate model assisted evolutionary algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(2):169–182, 2014.
[12] Wenlong Lyu, Pan Xue, Fan Yang, Changhao Yan, Zhiliang Hong, Xuan Zeng, and Dian Zhou. An efficient bayesian optimization approach for automated optimization of analog circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(6):1954–1967, 2017.
[13] Shuhan Zhang, Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. Bayesian optimization approach for analog circuit synthesis using neural network. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1463–1468. IEEE, 2019.
[14] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.
[15] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.
[16] Ralf Herbrich, Neil D Lawrence, and Matthias Seeger. Fast sparse gaussian process methods: The informative vector machine. In *Advances in neural information processing systems*, pages 625–632, 2003.
[17] Edward Lloyd Snelson. *Flexible and efficient Gaussian process models for machine learning*. PhD thesis, UCL (University College London), 2007.
[18] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.
[19] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition.
[20] Zushu Yan, Pui-In Mak, Man-Kay Law, and Rui P Martins. A 0.016-mm ^{2} 144-mu w three-stage amplifier capable of driving 1-to-15 nf capacitive load with >0.95-mhz gbw. *IEEE journal of solid-state circuits*, 48(2):527–540, 2013.