Received: 13 May 2026, Revised: 08 July 2026, Accepted: 10 July 2026, Online: 21 July 2026

DOI: https://dx.doi.org/10.55708/js0507001

Metamodel based Optimization for Analog Integrated Circuits

Vasile Grosu[*,1], **Nicolae Patache**[1], **Emilian David**[2]

[1]Gheorghe Asachi Technical University of Iasi, 700360, Romania

[2]Institute of Computer Science, Romanian Academy, Iasi Branch, Iasi, 700481, Romania

Email(s): vgrosu@etti.tuiasi.ro (V. Grosu), patachen@etti.tuiasi.ro (N. Patache), emilian.david@iit.academiaromana-is.ro (E. David)

[*]Corresponding author: Vasile Grosu, Carol 11A, Iasi, Romania, Email: vgrosu@etti.tuiasi.ro

ABSTRACT: In recent times, machine learning applications have become an important component of the analog integrated circuit development cycle. Several phases such as circuit sizing, optimization or pre-silicon verification have benefited from machine learning automation. For developing such machine learning models, a certain amount of training and testing samples needs to be acquired through circuit simulations. Depending on the circuit complexity this process can become very costly time-wise which can lead to a bottleneck in the development cycle. This article proposes a metamodel based optimization of circuit performances in which we develop a prior machine learning model that will be used as a placeholder for the simulation environment. The machine learning model is designed using adaptive sampling techniques which aim to reduce the number of sample points in the model while still having an overall good accuracy. When compared to traditional methods of sampling such as random and Latin hypercube sampling this method offers better models for either a given error or a set number of sample points. Afterwards, the proposed method for circuit optimization is tested on two use cases, a single-stage and a two-stage operational amplifier. In both cases we compared the metamodel based optimization with the classical approach and no significant differences were found.

KEYWORDS: analog integrated circuits; Bayesian optimization; Gaussian process regression; machine learning models; metamodels; operational amplifiers

1. Introduction

Machine learning (ML) based applications on the analog integrated circuit development cycle have become an attractive area of research over the past decade. Several phases have benefited from automation and improved performance. Such areas include circuit modeling [1, 2, 3, 4], sizing and optimization [5, 6], pre-silicon verification [7, 8] and lastly automated layout generation [9].

Regardless of the application scope (modeling, optimization or verification) a set of simulations needs to be performed in order to develop a ML model. Frequently, the training data is acquired using random or uniform approaches and reiterated until the developed model meets the desired accuracy. This can introduce systematic errors in the ML models by not taking into account output swings or sampling density in some areas.

This article proposes a unified approach to circuit modeling and optimization for accelerated development of analog integrated circuits (ICs). By building a machine learning model that emulates circuit behavior, the design optimization phase can be automated without concerns of simulation time and/or licensing costs. An increased benefit is obtained when the developed model is further used in re-design tasks or pre-silicon verification phases, for which no further simulations are required.

The proposed circuit modeling technique presents some improvements over prior publications with regard to the data acquisition process. In the proposed approach the sampling is done taking into account error thresholds from the modeled outputs modeled therefore the outputs that pass this threshold do not propose new points. The proposed approach is tested and compared to prior implementations on a set of 2D functions which serve as a good benchmark to test the algorithm. Afterwards, the results are replicated on two IC use cases, a single-stage operational amplifier and a two-stage operational amplifier.

The developed models are subsequently employed in an optimization task to efficiently navigate the design space and determine optimal circuit parameter settings. Acting as surrogate models for time-consuming simulations, they enable rapid assessment of design options, significantly speeding up convergence to optimal solutions while minimizing the need for extensive simulations or physical evaluations. For this, the most popular algorithm in the literature (Bayesian Optimization) was used and the results of running the optimization process using the actual simulation environment versus using the developed ML models as a placeholder are compared.

The remainder of this article is organized as follows: Section 2 presents a state-of-the-art review highlighting certain trends, advantages and opportunities that can be investigated. Section 3 presents the theoretical aspects of the proposed approach and is divided into two parts.The first part highlights the improved modeling technique for analog ICs, discusses the expansion to multiple -output modeling, and covers the machine learning algorithm options and the testing method for assessing model perfor-

mance. The second part provides a brief explanation of the optimization algorithm used. Section 4 presents the results for the proposed approach described in the previous section. The modeling approach is first tested on a set of synthetic functions and compared to previous implementations reported in literature and afterwards on two IC use cases. Afterwards, the developed models for the two IC use cases are employed in several ML-in-the-loop optimization runs and compared to simulator-in-the-loop runs with regard to the proposed implementations and results. Lastly, in Section 5, some conclusions are drawn along with a few further research topics.

2. State-of-the-art review

In the scientific literature there is a growing trend of ML applications in the IC design process. Several phases have benefited from ML-based automation, such as IC optimization [10, 11], pre-silicon verification [7, 8] and some aspects of automated layout [9] have been addressed in several publications. The main focus of this research is on IC behavior modeling for optimization frameworks so further in this chapter the focus will be discussing these two aspects of the IC design flow.

A comprehensive publication list regarding IC sizing/optimization and modeling is listed in the Table 1.

It what regards common use cases it can be noted that variations of operational amplifiers (OAs) with one, two or three stages as well as variations such as operational transconductance amplifiers (OTAs) are frequently proposed as use cases since they are fundamental building blocks present in almost all analog integrated circuits. In [32], a comparison for the number of simulations used in the development of machine learning models for the Miller OA use case is presented with numbers up to 10000 simulations reported to size a circuit, other reports [15, 18] give ranges between 500 and 5000 simulation points for either modeling or optimizing a set topology with reasonable accuracy. A few publications report higher number of samples (above 10000) used in developing the machine learning models [16, 24, 31] and are using artificial neural networks in order to develop accurate regression models. This number is primarily dependent on the total number of parameters used but gives us a rough estimate of the number of simulation points needed for certain IC applications.

Algorithm-wise, artificial neural networks (NNs) seem to be the preferred machine learning algorithm being successfully employed in sizing tools [12, 14, 15] and sizing tools coupled with an optimizer [13, 23, 25]. Several reasons can be attributed for this preference, namely the ease of implementation, robustness and flexibility with bigger data sets. For the circuit design optimization task a considerable amount of implementations are done using different Bayesian optimization (BO) approaches [18, 19, 11, 27].

Also, with an exponential grow in popularity we mention a few publications that employ large language models (LLMs) in the optimization process, either combined with optimization algorithms [33, 34, 35] or just the LLM with a simulator in the loop [36]. The use of LLM adds a dimension of domain knowledge to rapidly generate viable design points to remedy BO's inefficiency [34]. The mentioned publications report fewer simulations/iterations compared to classical optimizer-in-the-loop approaches to reach a set of required specifications. Furthermore, in [35] a novel class of circuits is tackled, namely DC-DC converters, with clear advantages over the compared standard BO approach obtaining overall better performances.

In what regards the data acquisition process for acquiring the training and testing data sets, very few publications address this problem leaving room for extra time spent on running simulations and training models. In this sense, the article proposes an improved adaptive sampling scheme based on active learning which has been shown in previous publications to have better results that fixed sampling schemes commonly used in literature. A more comprehensive explanation of the mechanisms involved will be given in the following sections of this article.

Beyond integrated circuit design, active data sampling and intelligent parameter optimization frameworks have found extensive utility across diverse engineering applications. For instance, advanced machine learning and sampling paradigms are increasingly applied to optimize beamforming and mode configuration in complex wireless communication networks [37, 38, 39].

3. Machine learning based analog integrated circuit sizing approach

In this section, the theoretical aspects of the proposed approach to analog circuit sizing/optimization are presented. The section is divided into two parts. Firstly, we explain the adaptive sampling methodology employed in developing the integrated circuit models. The sampling scheme is first described for a single output and then expanded to a multi output scheme for a faster model development with minimal losses regarding accuracy. The testing methodology is is also described in this chapter, motivating the use of various error metrics their advantages and drawbacks regarding the model performances.

Secondly, we provide a concise explanation of the optimization framework into which the previously developed ML model was integrated to work with a Bayesian optimization approach. BO is a sequential design strategy for global optimization of black-box functions that are expensive to evaluate. It builds a probabilistic surrogate model, typically a Gaussian process (GP), to estimate the objective function and uses an acquisition function to decide where to sample next, balancing exploration and exploitation. As previously stated, by embedding the ML model within the optimization process, we aim to enhance the overall system performance by making a fast and lightweight ML model which can be used in optimization approaches. The last part of the section outlines the structure of the framework, the role of the ML model within it, and the rationale behind the chosen optimization approach.

3.1. Adaptive sampling-based metamodel development

A ML model $m(x)$ which is trained with a set of N input-output points $x = [x_1, x_2, ..., x_N]$ and $y = [y_1, y_2, ..., y_N]$ given by an unknown function $f(x)$ as it is shown in (1),

Table 1: Comparison between papers published in literature.

Ref.	Algorithm used	Circuit use case	Task	Samples/Accuracy
[5]	Differential evolution&Bayesian inference	LDO	Optimization	- / -
[6]	Differential evolution&Bayesian inference	LDO	Optimization	- / -
[12]	Artificial neural networks	OA	Sizing tool	20000/92%
[13]	Artificial neural networks	OA& OTA	Sizing tool coupled with optimizer	- / -
[14]	Artificial neural networks	Various RF circuits	Sizing tool	-/>95%
[15]	Artificial neural networks	OA	Sizing tool	5000/ -
[16]	Artificial neural networks	OTA	Sizing tool	120000/-
[17]	Particle swarm optimization	OA	Optimization	- / -
[18]	Bayesian optimization	OA & others	Optimization	- / -
[19]	Bayesian optimization	OA & others	Optimization	300-3000 / -
[10]	Cuckoo Search Algorithm	OA	Optimization	- / -
[11]	Bayesian optimization	OA & LNA	Optimization	- / -
[20]	Evolutionary algorithms&deep learning	Several OA	Optimization	10000 / >80%
[21]	Bayesian optimization&Reinforcement learning	Several OA	Optimization	- / -
[22]	Graph neural networks&Reinforcement learning	LDO	Optimization	20000 / -
[23]	Artificial neural networks, Genetic Algorithms	Several OA	Sizing tool coupled with optimizer	- / -
[24]	Artificial neural networks	OA	Sizing tool	150000 / -
[25]	Artificial neural networks	Several OA	Sizing tool coupled with optimizer	- />95%
[26]	Reinforcement learning	Several OA	Optimization	- / -
[27]	Bayesian optimization	Several circuits	Optimization	- / -
[28]	Artificial neural networks	OA	Sizing tool	13500 / >90%
[29]	Gaussian process regression	Several circuits	Sizing tool	- / -
[30]	Bayesian optimization&Evolutionary algorithms	OA & LDO	Optimization	2000-15000 / -
[31]	Artificial neural networks	Active filter & OTA	Optimization	1900000 / >95%

can replicate the unknown function's behavior with reasonable accuracy, given that the N training points are sufficient.

$$y = f(x) \tag{1}$$

The predicted output $\tilde{y}$ would ideally be identical to y but in practical cases this is true within an accepted boundary ϵ as it is shown in (2) and (3).

$$\tilde{y} = m(x) \tag{2}$$

$$\tilde{y} = f(x) + \epsilon \tag{3}$$

Figure 1 showcases the general procedure of using ML models in order to predict a black-box function's response.

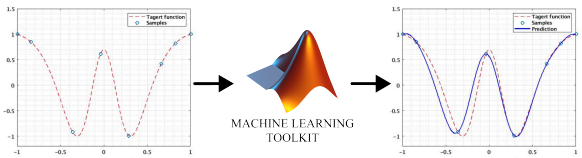

Figure 1: Metamodeling principle.

As stated previously, the acquisition process is traditionally done through fixed sampling schemes such as grid, random (RND), or Latin hypercube sampling (LHS), among various other approaches. In certain cases, these sampling schemes leave room for improvement by not taking into account aspects such as overall sampling density, output value swings or other information that can be extracted from the modeled circuit. This too, can be observed in the general example from Figure 1 in the area where points are not dense enough the predicted output differs more than in the regions which are sampled more densely.

In this sense, adaptive sampling schemes [40] have advantages over fixed sampling schemes by sampling the input domain via an acquisition function $a(x)$ which takes into account aspects such as, model variance, sample density, or output swings. One such approach is sampling by model's uncertainty/variance which was previously used with success for IC behavior modeling [1, 2, 3]. This way samples are selected based on the variation the model's response in order to minimize the samples needed to develop a ML model. In order to obtain the best model for a given task, an intelligent approach must be taken. Consequently, an efficient sampling scheme can significantly benefit the overall model-development process.

It has been shown previously in [1, 2, 3] that adaptive sampling methods based on active learning can offer a significant cost-benefit increase by incrementally sampling regions where the model error might improve significantly. The general method of sampling is highlighted in Figure 2. By developing an acquisition function based on the model's uncertainty, we can overall improve overall model prediction, and hence, the overall fitting of the model. Previous research has shown the advantage of uncertainty-based sampling over fixed sampling methods both on synthetic functions and on IC use cases. The process of obtaining this uncertainty metric is discussed in further sections of this article.

The uncertainty sampling approach was preferred because it is computationally inexpensive. Any other adaptive sampling approach can be employed in the further presented sampling schemes.

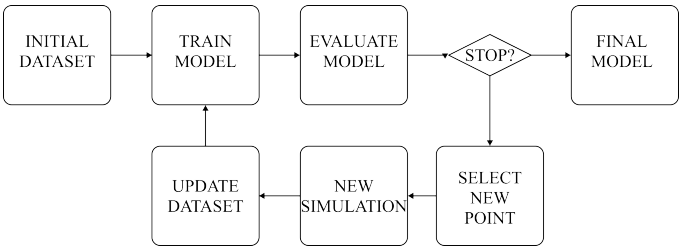

Figure 2: Active learning general principle.

For the proposed uncertainty sampling scheme, the most appropriate ML algorithm is Gaussian process regression (GPR) [41] since it provides a mean prediction and a corresponding uncertainty function as outputs. The uncertainty output represents a 95% confidence interval of the mean prediction. This uncertainty output is used to improve the overall model behavior by sampling in the regions where its value is at a maximum.

A general example on a generic 1D function is given in Figure 3.

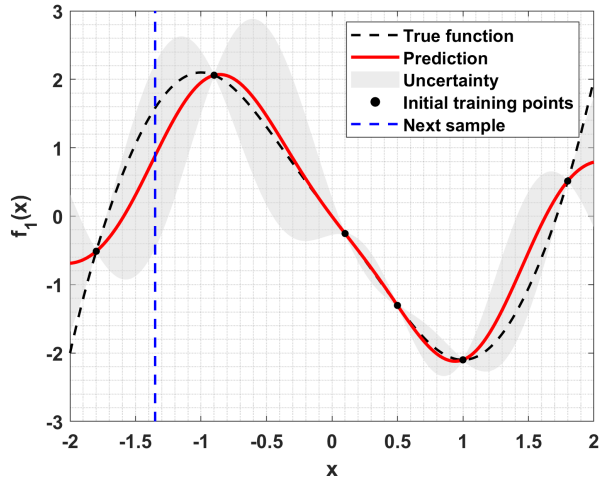

Figure 3: Training and prediction of a general function with GPR.

For this example, the best sample to improve model behavior is at the dashed blue line, since at that point the uncertainty of the prediction is at its highest. As a result, Figure 4 shows the new prediction with the added sample point included.

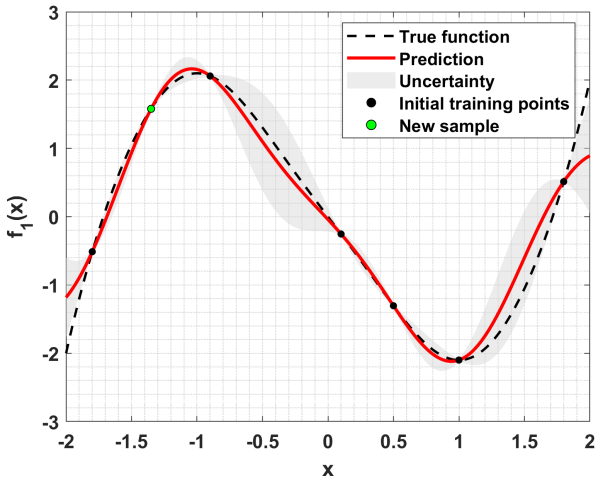

Figure 4: Training with the new sample example.

It is easily observable that in regions where points are not present or are sparse, the prediction differs from the target function, which is expected, and the uncertainty output of the GPR is significantly larger. Thus, by sampling in regions where this output is high improves the model. In some cases, GPR might be infeasible for several reasons. One of these reasons can be the high computational complexity of the algorithm, which for the GPR algorithm is $O(N^3)$. This can lead to high training times for large or high-dimensional data sets.

3.1.1. *Multiple output extension*

Since a single simulation run provides measurements for several outputs, the acquisition process naturally favors the development of multi-output sampling schemes. In this approach, all the outputs are modeled at the same time yielding from a speed-up in this case. The model development scheme presented in this article is based on an improved approach that has been used in prior research [3]. The improved sampling scheme is depicted in Figure 5.

The difference from the initial sampling approach is that for certain outputs the sampling will be stopped when they reach certain threshold errors imposed by the end user. This way a further simulation effort reduction is obtained by not oversampling some regions and at each iteration only P samples are added, with $P \leq N$ where N is the number of outputs being modeled and P is the number of outputs which are proposing a new point.

As a trivial example, we can take 2 general functions f_1 and f_2, defined as follows:

$$f_1(x) = x^3 - 3 \cdot x + 0.1 \cdot \sin(1.5\pi x) \tag{4}$$

$$f_2(x) = x^4 - 2 \cdot x^2; \tag{5}$$

The hypothesis is that f_1 and f_2 are dependent on each other and in a given application, the variation of x either $1D$ or nD in a more general case can provide f_1 and f_2 at the same time. Given this aspect, each function will have an optimum sampling point which is depicted by the green point in both Figure 6 and Figure 7. Training each function only with its respective selected sampling points would result in the red-line function which would further

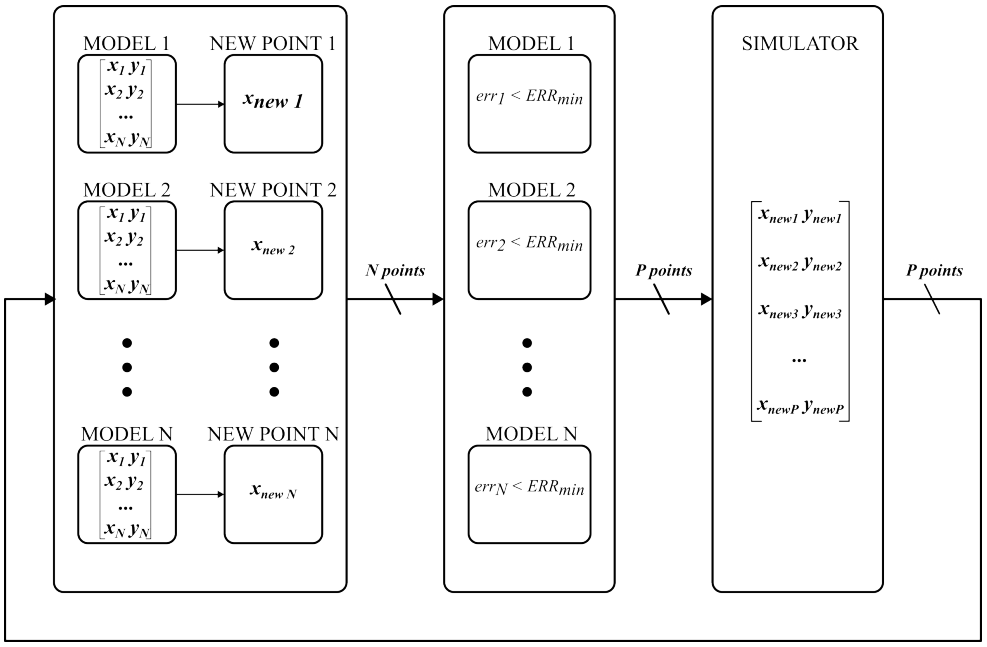

Figure 5: Improved multi output metamodeling sampling scheme.

require more points, and the functions trained with both points, which have been acquired by one function evaluation each, would give us the blue-line function, which it is obviously more similar to the true function.

The diagram in Figure 5 can be further explained as follows. Let there be N separate models, m trained at iteration t with their respective data sets $D_n^{(t)}$, where $n \in \{1...N\}$

$$m_n(\mathbf{x}) \mid \mathcal{D}_n^{(t)} = \left\{(\mathbf{x}_n, y_n)\right\}_{n=1}^{N} \tag{6}$$

Each model uses an acquisition function $\alpha_n(\mathbf{x})$ (which relies on the model's current uncertainty predictions) to propose exactly one new optimal candidate point:

$$\mathbf{x}_{\text{new},n} = \arg\max_{\mathbf{x}\in\mathcal{X}} a_n\left(\mathbf{x}; m^{(t)}\right) \tag{7}$$

This generates a set of N candidate points:

$$\mathcal{X}_{\text{candidates}} = \{\mathbf{x}_{\text{new},1}, \mathbf{x}_{\text{new},2}, \ldots, \mathbf{x}_{\text{new},N}\} \tag{8}$$

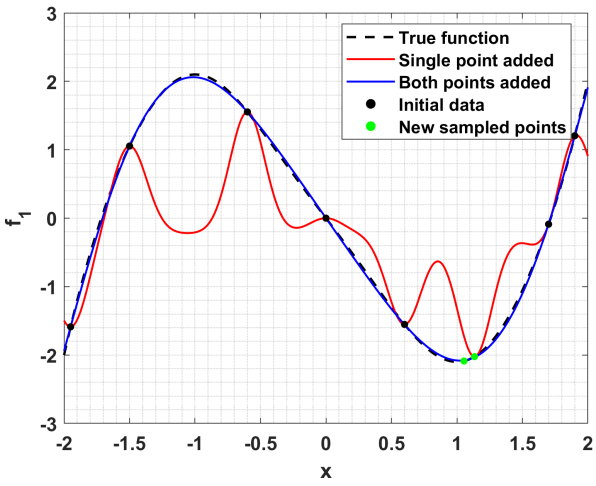

Figure 6: Modelling of f_1.

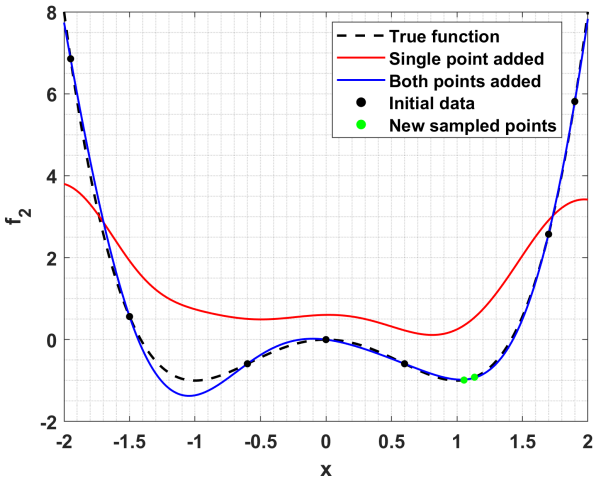

Figure 7: Modelling of f_2.

We define a selection indicator function $\mathbb{I}_n$ for each candidate point:

$$\mathbb{I}_n = \begin{cases} 1 & \text{if } \mathrm{err}_n(\mathbf{x}_{\mathrm{new},n}) < \mathrm{ERR}_{\mathrm{min}} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Applying this filter reduces the N candidates down to a subset of P points ($P \leq M$):

$$\mathcal{X}_{\mathrm{selected}} = \{\mathbf{x}_{\mathrm{new},n} \mid \mathbb{I}_m = 1\} = \{\mathbf{x}_{\mathrm{new},1}, \ldots, \mathbf{x}_{\mathrm{new},P}\} \tag{10}$$

Then the selected P samples are simulated using the ground-truth simulator and obtain a new batch of training samples:

$$\mathcal{D}_{\mathrm{batch}} = \left\{(\mathbf{x}_{\mathrm{new},p}, y_{\mathrm{new},p})\right\}_{p=1}^{P} \tag{11}$$

Finally, the newly simulated data points are used to update the training sets for the next iteration $t \leftarrow t+1$:

$$\mathcal{D}_n^{(t+1)} = \mathcal{D}_n^{(t)} \cup \mathcal{D}_{\mathrm{batch}} \quad \forall\, n \tag{12}$$

3.1.2. Testing machine learning models

For testing machine learning models different error metrics can be used. A set of popular error metrics are presented in Table 2 each of them having particular advantages.

In the experiments presented, results will be plotted using the relative RMSE (both in absolute value and percentage-wise) which takes into account also the output magnitude. The relative RMSE has an advantage over the other metrics because it takes into account the output swing by dividing the RMSE by the L_2 norm. For a better evaluation several error metrics can be considered, such as using both the ME and relative RMSE as in [1, 2].

Table 2: Usual error functions used in ML testing.

fun no.	Equation
Maximum error (ME)	$max(y_t - m(x_t))$
Mean average error (MAE)	$\frac{1}{n}\sum_{i=1}^{n}(y_t - m(x_t))$
Mean square error(MSE)	$\frac{1}{n}\sum_{i=1}^{n}(y_t - m(x_t))^2$
Root mean square error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_t - m(x_t))^2}$
Relative RMSE	$\frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_t - m(x_t))^2}}{\sqrt{\sum_{i=1}^{n}(y_t)^2}}$
Relative RMSE [%]	$\frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_t - m(x_t))^2}}{\sqrt{\sum_{i=1}^{n}(y_t)^2}} \cdot 100$

Typically, the test set is a subset of the total available points, commonly between 10% and 20%. Given that we are trying to asses the validity of our sampling methods we took a larger testing set only for developmental purposes which will give us a more stable error over the number of samples added to the model. For the synthetic benchmark functions a set of $n = 200$ points was used which covers the search space densely enough and for the ICs use cases a set of $n = 1000$ points will be used. The test set will be and independent set which will not be used in the training of the ML models.

3.2. Optimization algorithms for integrated circuits

The optimization of analog ICs has the scope of maximizing or minimizing certain design specifications in a given search space. Literature proposes several frameworks in this sense, with various optimization algorithms having favorable results such as Bayesian optimization [19, 10, 11], Differential evolution [5, 6] or Genetic algorithms [20]. Further we will use Bayesian optimization with various objective function configurations given their popularity.

Bayesian optimization is one of the most accurate methods for determining a global optimum, the framework can outperform grid and random search but can be slow when many hyperparameters are involved [42].

The Bayesian optimization approach consists of two essential parts. The first part is the probabilistic surrogate model of the objective function, which is updated incrementally when new data is observed. With the probabilistic surrogate model, the model uncertainty can also be evaluated. The second part is an acquisition function, which is used to explore the state space based on the surrogate model optimally. The acquisition function aims to balance the exploration and exploitation during the optimization process. A typical acquisition function which is also the one we used in our experiments is the Expected Improvement (EI) function. A detailed guide and implementation of Bayesian optimization can be found in [43] and [44].

The optimizer will aim to find the minimum of the unknown black-box function $f(x)$ defined by equation (13). The general mode of BO operation is presented in equation (14). The algorithm will aim to find the point x_m which minimizes the response of $f(x)$.

$$f(x) : X \rightarrow \mathbb{R}^n, X \subseteq \mathbb{R}^n \tag{13}$$

$$x_m = argmin(f(x)), x \in X \tag{14}$$

3.3. The overall proposed system

In Figure 8 we can observe the entire system depicted. Firstly, the model development which is done with MATLAB [45] and LTSpice [46] but other alternatives can be used as well. When the end user is satisfied with the overall accuracy of the model or has reached the imposed simulation budget, the model can afterwards be used in the optimization process in order to size a predefined circuit topology to a set of design specifications. The advantage of this approach is that the model can be reused in several designs with different sets of specifications without further simulator use. Similarly, the bottleneck introduced by either the time needed to perform the simulations or the licensing costs attributed to the simulation environment becomes negligible.

In the presented results a BSIM3 model for the transistors was used in a standard $0.18\mu m$ CMOS PDK. Nonetheless, the current approach allows for a easy technology

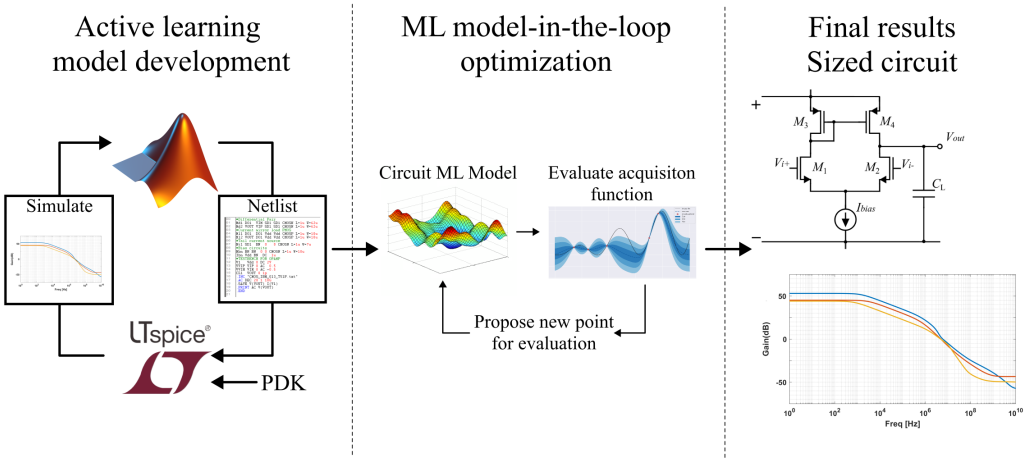

Figure 8: Overall proposed system

change, migration to another process can be done only by changing the PDK file.

For the optimization of hyperparameters the same BO framework can be employed. In the presented experiments Bayesian optimization with k-fold cross-validation was used. This is a powerful technique for hyperparameter tuning that balances efficiency and robustness. Bayesian optimization models the performance of hyperparameters as a probabilistic function and intelligently explores the search space to find the optimal settings. By incorporating k-fold cross-validation, the method ensures that performance estimates are reliable and not biased by a single train-test split. This combination allows for more informed decisions during the search process, leading to better generalization and model performance while reducing the number of evaluations needed compared to grid or random search.

As a kernel function the squared exponential function was used, which introduced two hyperparameters, namely, signal variance (σ_f^2) and length scale (σ_l) according to equation (15):

$$k(x, x') = \sigma_f^2 e^{-\frac{(x-x')^2}{2\sigma_l^2}} \tag{15}$$

Additionally, the estimate of noise standard deviation (σ_0) was considered as a hyperparameter in order to regulate over and under fitting of the model.

4. Experimental results

In this section, experimental results for the proposed approach are presented. Firstly, results on the proposed improved modeling approach will be presented on a set of synthetic functions which provides an easy way to test laborious algorithms and afterwards on the two mentioned IC use cases. The results are compared from the number of samples used to reach a certain error and also the error obtained in the imposed simulation budget.

Secondly, the developed models will be used in various optimization runs with BO and several optimization objectives in order to asses the utility of the proposed modeling technique.

4.1. Metamodel development

4.1.1. Synthetic function results

It is good practice to test complicated or time-consuming algorithms first on a set of low-dimension synthetic functions in order to assess the validity of the proposed algorithms. Below, we will highlight some preliminary results for a set of surrogate functions of 2D dimension (one graphical example being provided in Figure 9 and the rest in Table 3) which serve as a good baseline evaluation when designing new algorithms. These functions are in part popular benchmark functions found in literature [47] while a few are designed by hand.

Table 3: Equations for the functions used in the experiments.

Name	Equation
F1	$\sum_{i=1}^{2} 50x_i sin(\sqrt{\lvert 50x_i \rvert})$
F2	$\sum_{i=1}^{2} 1000(1 + exp(-\frac{2x_i^2}{15}) -2exp(-2x_i^2))cos^2(2x_i))$
F3	$\prod_{i=1}^{2} exp(-2x_i^2)$
F4	$\prod_{i=1}^{2} 0.3exp(-\frac{(x_i-0.5(-1^i))^2}{0.05})+ 0.6exp(-\frac{(x_i-0.6(-1^i))^2}{0.05}) + exp(-\frac{x_i^2}{0.05})$
F5	$\sum_{i=1}^{2} sinc(\pi x_i)$
F6	$\sum_{i=1}^{2} sinc(\pi x_i) + tanh(\pi x_i)$
F7	$\sum_{i=1}^{2} x_i^2 + 0.5exp(-\frac{x_i^2}{0.2})$
F8	$\sum_{i=1}^{2} \lvert x_i^3 \rvert + sinc(\pi x_i + 1)$

Every sampling method will be run for a total of 500 training samples, and averaged for a number of 10 runs to ensure an accurate interpretation of the errors. For the multi-output improved approach, the threshold for the models will be 0.001 which will mean a relative RMSE

of 0.1%. This is realistic for the 2D functions given the complexity of the functions and the size of the testing set.

In practice, these thresholds cannot be known *a priori* because the exact performance landscape of a new circuit block is completely hidden prior to sampling. Instead, threshold selection is a direct reflection of the analog designer's intuition balanced against their simulation budget. This will further be the case for our IC use cases as well.

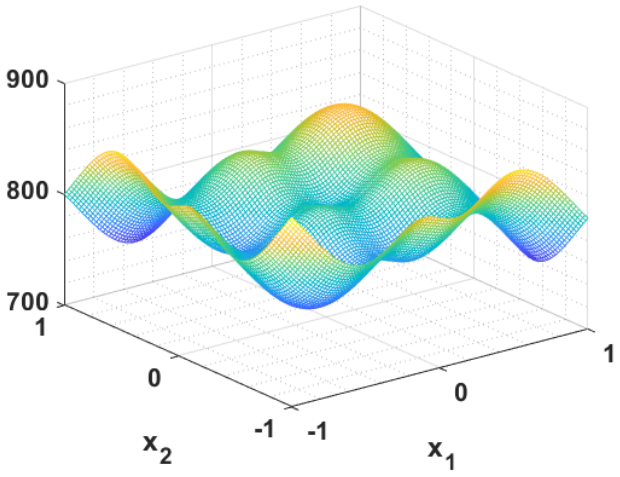

Figure 9: Example of function used in experiments.

From the results in Figure 10 and Tables 4 and 5 it can be observed that the ALS sampling for a single output performs the best, which is expected. Following that, we can see that on average, the ALS-MO sampling schemes (where ALS-MO is the standard parallel sampling approach and ALS-MO2 is the improved approach with the threshold implemented) yield results better than the fixed methods and slightly worse than the single-output sampling, which again is expected. Functions F_3 and F_4 are the simplest functions and hence are the easiest to model; for these functions, the differences between the methods are minimal.

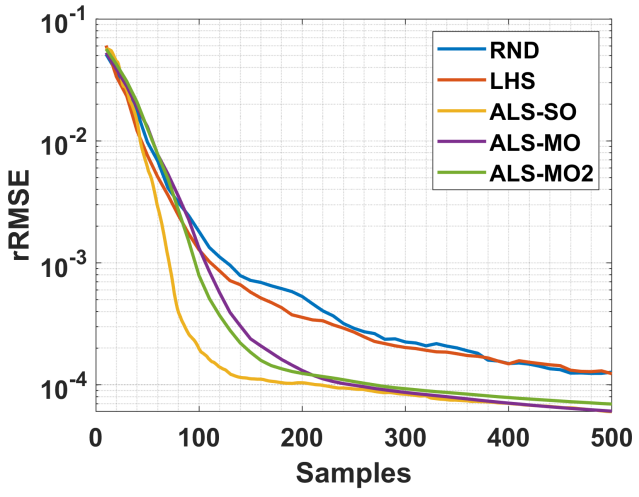

Figure 10: Results of modeling over the number of sample for one of the synthetic functions.

Table 4: Surrogate function - samples needed to reach 0.1 % error

Fun.	RND	LHS	ALS		
			SO	MO1	MO2
F1	104	99	70	90	84
F2	130	105	82	105	98
F3	140	120	80	120	120
F4	400	400	260	360	360
F5	320	260	100	160	140
F6	340	300	100	170	120
F7	130	120	90	110	100
F8	260	260	100	160	135

Table 5: Surrogate function - error at 300 samples

Fun.	RND	LHS	ALS		
			SO	MO1	MO2
F1	0.02%	0.02%	0.007%	0.01%	0.01%
F2	0.02%	0.02%	0.009%	0.009%	0.009%
F3	0.04%	0.04%	0.04%	0.04%	0.04%
F4	0.2%	0.16%	0.007%	0.18%	0.18%
F5	0.01%	0.09%	0.02%	0.03%	0.02%
F6	0.1%	0.1%	0.003%	0.03%	0.04%
F7	0.02%	0.02%	0.01%	0.01%	0.01%
F8	0.1%	0.08%	0.04%	0.04%	0.04%

The key takeaway from these results is that for functions that are modeled better (i.e., with lower error), there is no significant difference whether the respective model requests a new point or not, since it already provides an overall good representation of the function. Furthermore, channeling those points toward other functions offers a steeper descent in error relative to the number of samples, resulting in a better overall modeling of our functions.

4.1.2. Integrated circuit use cases

Based on the literature review from Table 1, two circuits were chosen as use cases, which are presented in Figure 11 and 12. In Figure 11 a single-stage operational amplifier is presented and in Figure 12 a two-stage operational amplifier which will serve as a more complex use case. The circuit in Figure 11 will have 6 input parameters to be modeled. They represent transistor lengths and width: namely, $W_{1,2}, W_{3,4}$ and W_5 for the widths and $L_{1,2}, L_{3,4}$ and L_5 for the lengths. The circuit in Figure 12 will have 4 more input parameters represented by the output stage parameters, W_6, L_6, W_7 and L_7. The other components present in the figures will be set at a fixed value in the presented experiments.

The input search domains for these use cases are as follows: the widths can vary between [10 μm, 500 μm] and the lengths between [0.5 μm, 5 μm]. More specific modeling tasks can employ different search spaces for each variable, as defined by the end-user.

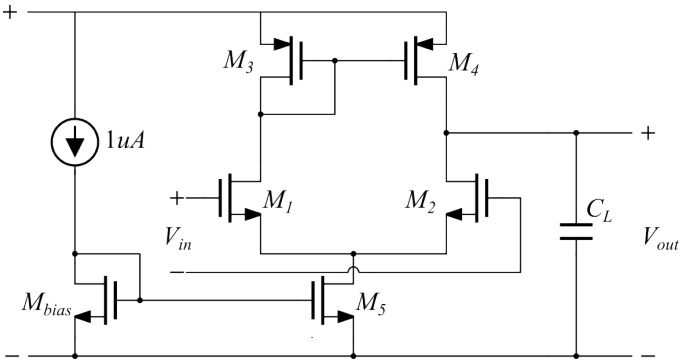

Figure 11: Single stage OA use case.

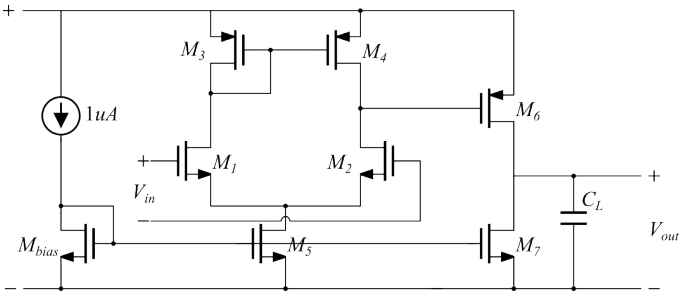

Figure 12: Two-stage OA use case.

For these two use cases the outputs modeled will be a series of alternating current (AC) parameters, namely the AC Gain, common mode rejection ratio (CMRR) and power supply rejection ratio (PSRR). Given that these AC parameters are waveform dependent on the frequency of operation the values are averaged over several frequency intervals. The intervals as it can be observed in the Figure 13 are $[1Hz - 10kHz]$, $[10kHz - 100kHz]$, $[100kHz - 1MHz]$ and $[1MHz - 10GHz]$.

For the single stage OA the modeling budget will be 3000 points and for the two stage OA the budget will be raised to 5000 simulation points which are congruent with other papers propose as simulation budgets. As well as in the synthetic functions experiment the errors will be averaged over a number of 10 runs to have an accurate reading of the error gradient. For these use cases, only the fixed sampling methods implementation and the improved multi-output sampling approach are presented. Furthermore, these ALS-MO models will be used in an BO optimization approach which will be presented in the following section.

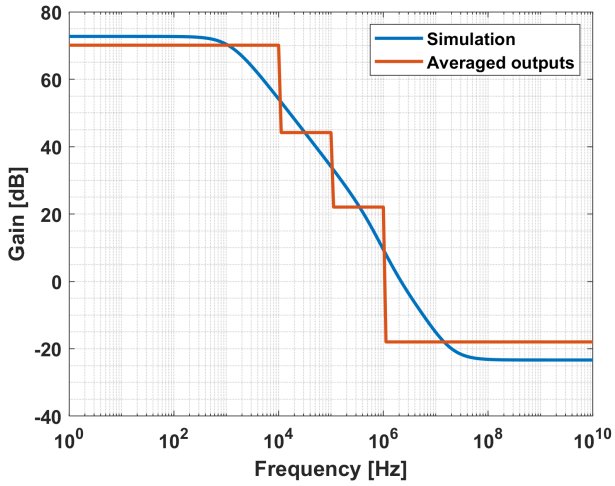

Figure 13: Measurement method for the AC outputs.

Figure 14 shows the error evolution across the number of samples. Table 6 and Table 7 contain the extracted data for all the modeled outputs. It can be observed that in this case some outputs not reach the desired 5% relative error but are relatively close. Sample-wise it can be concluded that a reduction of at least 33% was achieved for this threshold.

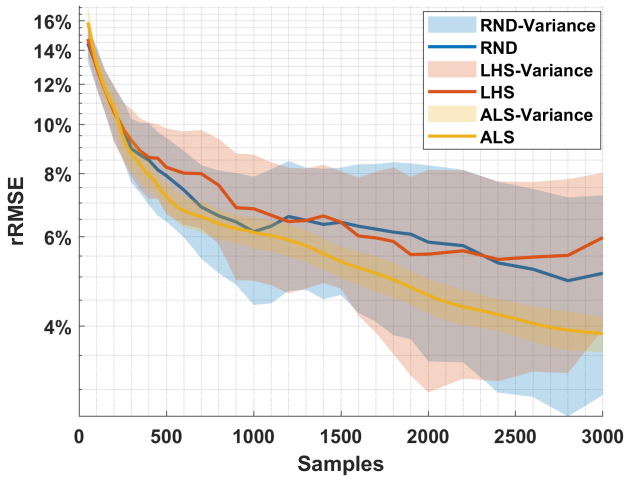

Figure 14: Modeling error for $Gain_{100k-1M}$ - single stage OA usecase

Table 6: Single stage OA – samples needed to reach 5 % error

Output	RND	LHS	ALS - MO
$Gain_{1-10k}$	230	220	180
$Gain_{10-100k}$	2500	–	1500
$Gain_{100-1M}$	–	1800	980
$Gain_{1M-10G}$	1200	1200	960
$CMRR_{1-10k}$	1150	980	570
$CMRR_{10k-100k}$	2950	2600	600
$CMRR_{100k-1M}$	2300	2800	580
$CMRR_{1M-10G}$	510	520	300
$PSRR_{1-10k}$	100	100	100
$PSRR_{10k-100k}$	–	2900	2000
$PSRR_{100k-1M}$	1550	2200	1800
$PSRR_{1M-10G}$	1300	1100	1150

Table 7: Single stage OA – error at 3000 samples

Output	RND	LHS	ALS - MO
$Gain_{1-10k}$	2.7%	2.5%	1.5%
$Gain_{10-100k}$	5%	6%	3.9%
$Gain_{100-1M}$	5.5%	4.8%	3.9%
$Gain_{1M-10G}$	4.5%	4%	3%
$CMRR_{1-10k}$	4.7%	4.5%	3.4%
$CMRR_{10k-100k}$	4.6%	4.7%	3.4%
$CMRR_{100k-1M}$	4.6%	4.4%	2.8%
$CMRR_{1M-10G}$	3.2%	3.4%	2.5%
$PSRR_{1-10k}$	2.1%	1.9%	1.5%
$PSRR_{10k-100k}$	5.4%	4.9%	4.5%
$PSRR_{100k-1M}$	4.5%	4.5%	4.5%
$PSRR_{1M-10G}$	4%	4%	4%

Figure 15 along with Table 8 and Table 9 presents the modeling process for the two-stage OA use case. Overall, the results are consistent with the synthetic function benchmark set and the single-stage OA. For this use case, a larger reduction in the required samples was achieved, representing at least a 48% reduction. Furthermore, it is important to note that for $Gain_{1M-10G}$ and all $CMRR$ outputs, the error in the fixed sampling case is significantly higher than in the adaptive case, further demonstrating the advantages of this approach.

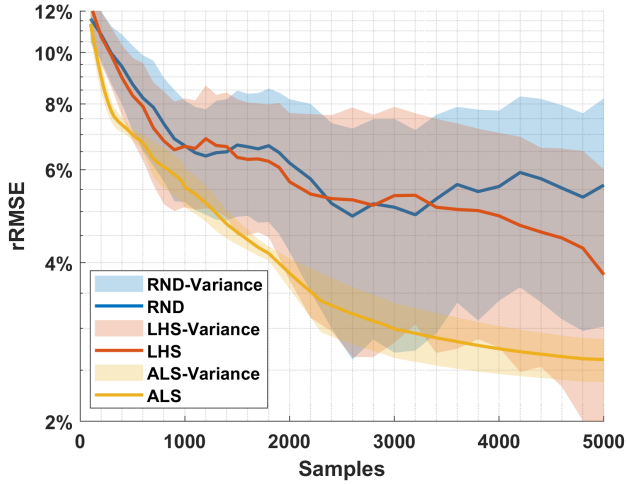

Figure 15: Modeling error for $CMRR_{1-10k}$- two stage OA usecase

Table 8: Two stage OA – samples needed to reach 5 % error

Output	RND	LHS	ALS - MO
$Gain_{1-10k}$	2800	2800	2450
$Gain_{10-100k}$	2600	3200	2300
$Gain_{100-1M}$	4500	–	2500
$Gain_{1M-10G}$	–	–	800
$CMRR_{1-10k}$	2550	3500	850
$CMRR_{10k-100k}$	2100	2000	600
$CMRR_{100k-1M}$	8500	1400	550
$CMRR_{1M-10G}$	1900	1800	550
$PSRR_{1-10k}$	2600	2600	2100
$PSRR_{10k-100k}$	3800	2700	2600
$PSRR_{100k-1M}$	4000	4200	2200
$PSRR_{1M-10G}$	4500	–	2000

Table 9: Two stage OA – error at 5000 samples

Output	RND	LHS	ALS - MO
$Gain_{1-10k}$	4.5%	4.5%	4.5%
$Gain_{10-100k}$	4.5%	4.5%	4.5%
$Gain_{100-1M}$	6%	6.8%	4.4%
$Gain_{1M-10G}$	7.5%	7%	**0.4**%
$CMRR_{1-10k}$	5%	4%	**2**%
$CMRR_{10k-100k}$	4.5%	4.9%	**1**%
$CMRR_{100k-1M}$	3%	2.4%	**1.5**%
$CMRR_{1M-10G}$	3.4%	3.7%	**0.8**%
$PSRR_{1-10k}$	4.3%	3%	2%
$PSRR_{10k-100k}$	5.2%	4.7%	5%
$PSRR_{100k-1M}$	5%	5%	4.5%
$PSRR_{1M-10G}$	4.5%	4.9%	4%

4.2. Sensitivity analysis of the models

In this section, we aim to investigate certain aspects which are subject to variability, such as the predicted output regarding the number of averaged models, the variation in errors for all the modeled outputs, and the variability subject to the threshold errors of the outputs.

Firstly, the importance of segmenting will be investigated. Given that we are approximating a continuous waveform with a "discrete" approximation, in theory, a more compact interval sequencing would result in higher model fidelity regarding predictions. In practice, the segmentation of the frequency intervals is defined by the available computational power and, ultimately, by the experience of the analog designer.

Figure 16 highlights the results of several models trained with the same simulation data set but with different intervals used for averaging. It is clearly more reliable to divide the waveform into more models, using smaller frequency intervals, while keeping in mind the computational expense that comes with it. For more simple waveforms, as it is shown in Figure 16 between pole-zero positions can be estimated with simple piece-wise approximations and difference between 2 successive intervals.

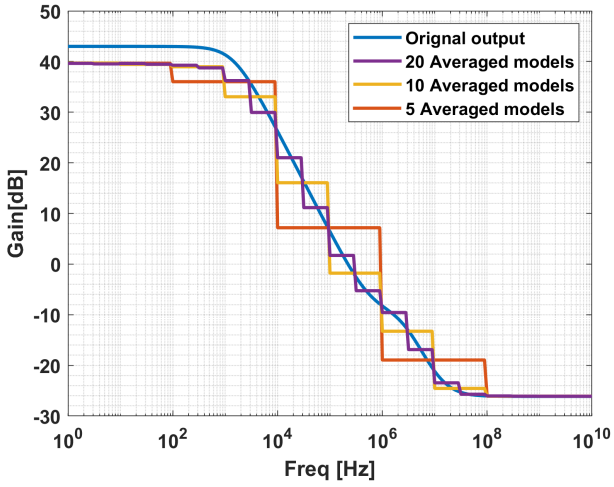

Figure 16: Variation of prediction based on output segmenting

For this, the yellow trace in Figure 16 highlights clearly this since the models are averaged over each decade. For

example, the two decades between $10kHz - 1MHz$ have a difference of $-18dB$ which correlates closely with a $-20dB$ decrease from the pole positioned around $1kHz$. Between $100kHz - 10MHz$ the difference drops to around $-10dB$ which is explained by the position of a zero which can be observed on the blue waveform, which represents the actual simulation.

Figure 17 and Figure 18 highlight the variation of modeling error for each output. These variations are computed from the means that are presented in Table 7 and Table 9 over the 10 previously mentioned runs. Equations (16 - 17) present the way in which these variations are computed:

$$\tilde{E} = \frac{1}{N}\sum_{i=1}^{N} E_i \tag{16}$$

$$Variance[\%] = \frac{\frac{1}{N}\sum_{i=1}^{N}|E_i - \tilde{E}|}{\tilde{E}} \times 100 \tag{17}$$

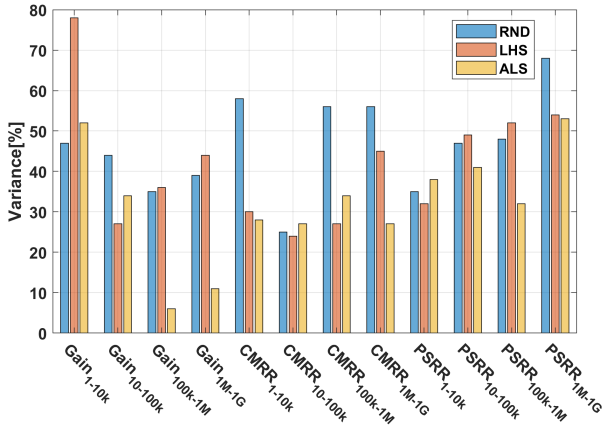

Figure 17: Variation of the end model errors over the 10 runs – single stage OA

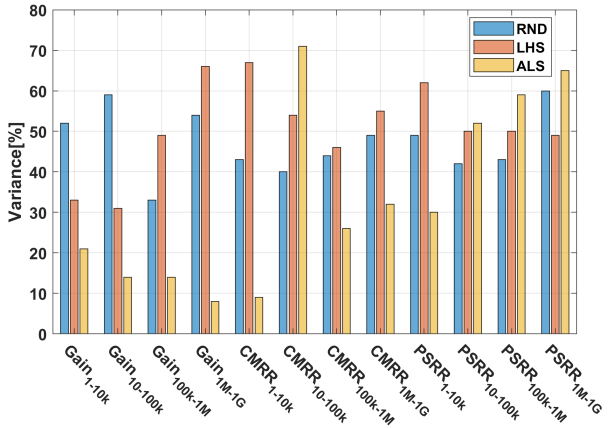

Figure 18: Variation of the end model errors over the 10 runs – two stage OA

On average, for both use cases, the proposed adaptive sampling technique has a smaller variance of around 30% from the mean error, whereas their fixed sampling counterparts between $40-50\%$ depending on the use case. Because this variation is relative to the mean error and not an absolute value, the resulting values are larger, but they highlight that the adaptive sampling method consistently yields better results.

Lastly, the influence of the threshold error on the size of the training data should be examined. To address this, a relation between the number of samples used in training and the testing error should be determined.

As an example, the single stage OA was selected, firstly the $Gain_{100k-1M}$ dependency on the data size was modeled and afterward, all the model outputs were considered. Empirically, when training the models, the points at which certain key errors (15%, 10%, 5%, etc.) were reached and the number of samples required to reach those errors were collected in order to determine the dependency. A ML model was trained for this purpose, and a strong nonlinear dependency can be observed from Figure 19.

Figure 20 the training data was obtained from all the modeled outputs to get a sense of general complexity of the circuit. In this example two distinct plateaus can be observed, one around the 7% and the other one around the 5% error threshold. This indicates that the training data size is beginning to yield diminishing returns.

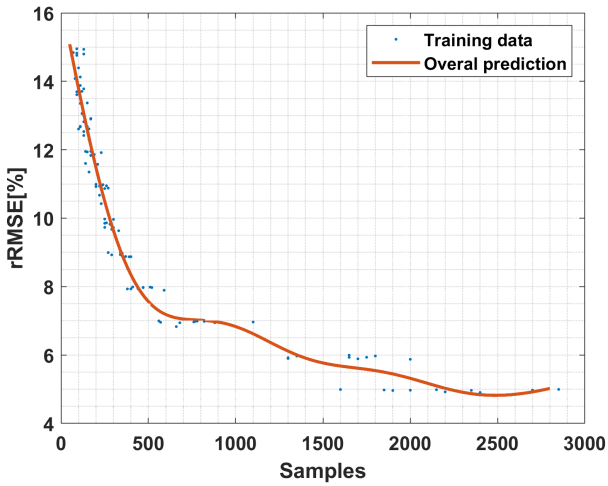

Figure 19: Dependency of number of samples and threshold error for $Gain_{100k-1M}$ - single stage OA.

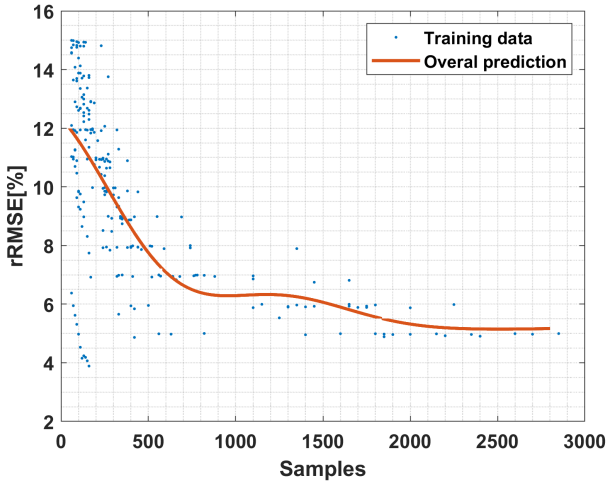

Figure 20: Dependency of number of samples and threshold error for all the outputs combined.

Naturally, the exact behavior of the error dependency remains highly circuit- or application-specific. However, a purely linear trend is highly unlikely, and an overall nonlinear relationship is expected nonetheless across most

multi-output systems. This inherent non-linearity underscores the necessity of a threshold-based sampling approach capable of capturing complex inter-dependencies without over-allocating simulation resources.

4.3. Metamodel aided Bayesian optimization for IC sizing

The main scope of this article, is to investigate if there are any significant differences in using a ML-models-in-the-loop as a placeholder for actual simulation environments with regards to overall simulator use and the proposed solution. Even if the ML model approach does not use fewer evaluation points, the possibility of reuse in other tasks, such as redesign or verification, implies an advantage regarding the total use of the simulator.

As an acquisition function for the BO framework, the Expected Improvement (EI) function will be used, as it is one of the most popular functions. Also, since the BO framework returns the minimum of the function, all the objective functions will be negated in order to maximize the proposed objectives.

Since BO is a single-objective optimizer, to optimize an IC that has multiple outputs to be maximized/minimized, a compounded function can be defined to assess the quality of the proposed design provided by the ML algorithm. This function is usually called a Figure of Merit (FoM) as well.

For the subsequent optimization runs, the models will utilize the 2000 training sample models for the single-stage operational amplifier and the 3000 training samples models for the two-stage operational amplifier. These budgets assure a less than 5% error (or more than 95% accuracy) for most of the modeled outputs, as observed in Figure 14 and Figure 15.

4.3.1. Single stage operational amplifier use case

The first test was done on a single-objective optimization performed for the single-stage OA. For this test, the goal was to obtain a high DC gain; thus, only the first AC gain output of the modeled circuit was used.

In Figure 21 the results for the single objective optimization approach are presented. The iteration budget for the single stage amplifier was set to 200, similar to what other publications report. In Figure 22 we can we the simulation results for the circuits proposed by the BO run, and it can be observed there are no significant differences.

A more sophisticated approach is presented in the next figures. The proposed FoM_1 expression is shown in equation (18). This approach aims to maximum equally the AC Gain and the CMRR up to 100 kHz. If desired, a weighted function can be designed in order to favor one output over the other by assigning different weights to the equation terms.

$$FoM_1 = -\frac{ACGain_{1Hz-1kHz} + ACGain_{10-100kHz}}{2} - \frac{CMRR_{1Hz-1kHz} + CMRR_{10-100kHz}}{2} \quad (18)$$

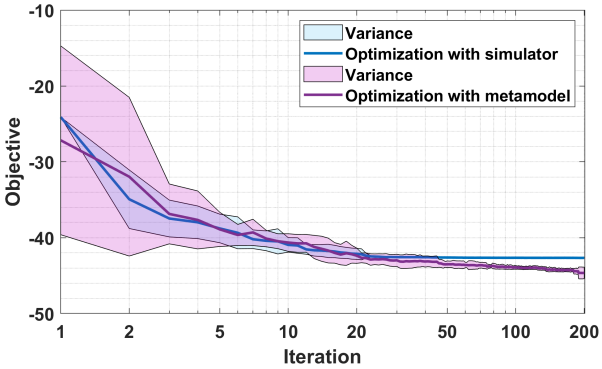

Figure 21: Optimization run for single objective optimization.

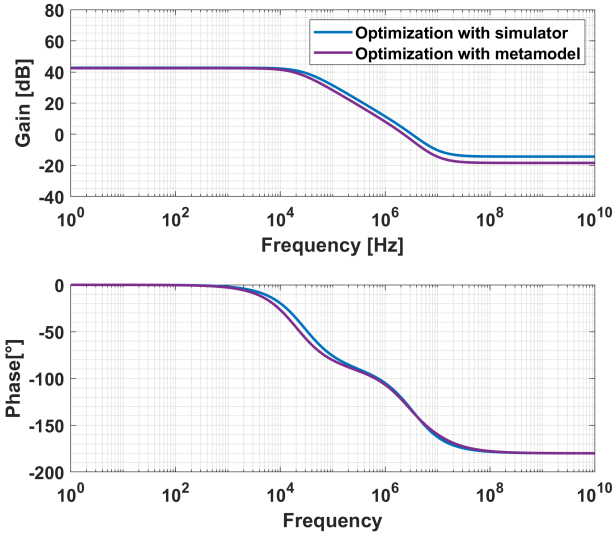

Figure 22: Best candidates for the previous run optimization.

For the second run, we can observe a difference between the predicted objective with the ML model and the predicted objective with the simulator as shown in figure 23 and 24. This is mainly due to systematic error in the previously developed model, but it remains within tolerable margins.

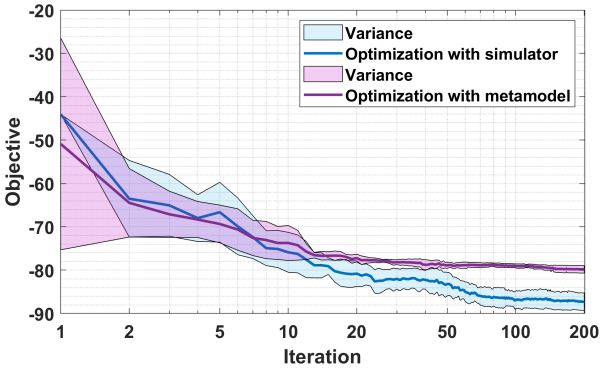

Figure 23: Optimization run for multi objective optimization of the single stage operational amplifier.

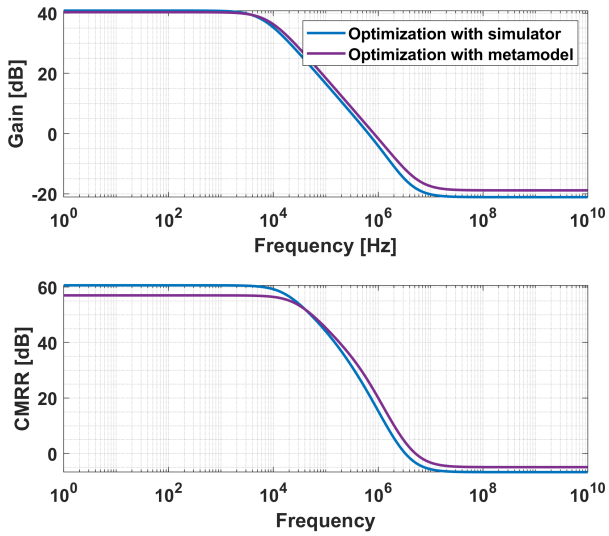

Figure 24: Best candidates for the previous run optimization.

4.3.2. *Two stage operational amplifier use case*

Further, we employed the same optimization objective from (18) for the second use case, which was the two-stage operational amplifier that has 10 input factors as shown in figure 25 and 26.

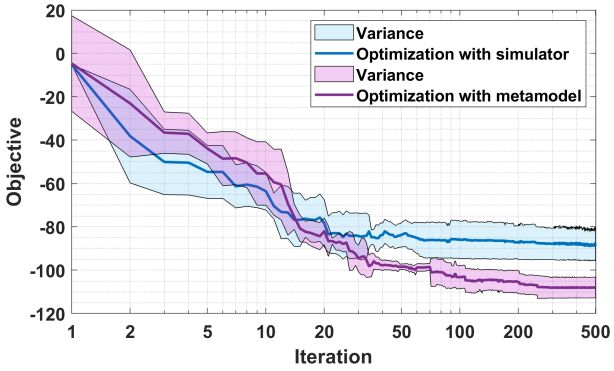

Figure 25: Optimization run for the first multi objective optimization of the two stage amplifier.

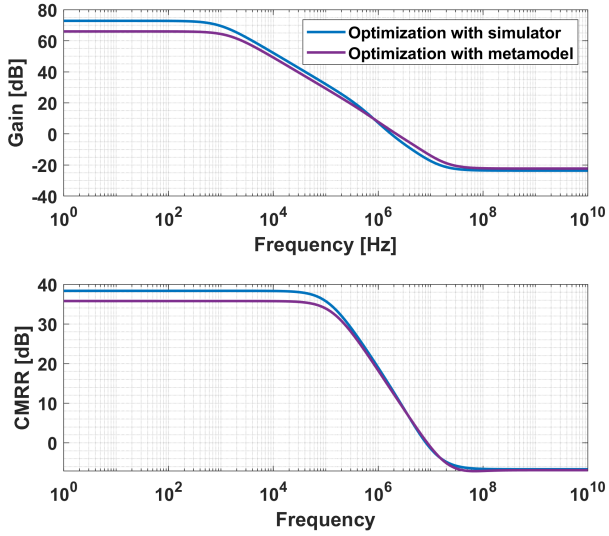

Figure 26: Best candidates for the previous run optimization.

Lastly, we performed an optimization with all three outputs (AC Gain, CMRR and PSRR) as shown in figure 27 and 28, in the same format as previously shown and depicted in equation (19).

$$FoM_2 = -\frac{ACGain_{1Hz-1kHz} + ACGain_{10-100kHz}}{2} - \frac{CMRR_{1Hz-1kHz} + CMRR_{10-100kHz}}{2} - \frac{PSRR_{1Hz-1kHz} + PSRR_{10-100kHz}}{2} \quad (19)$$

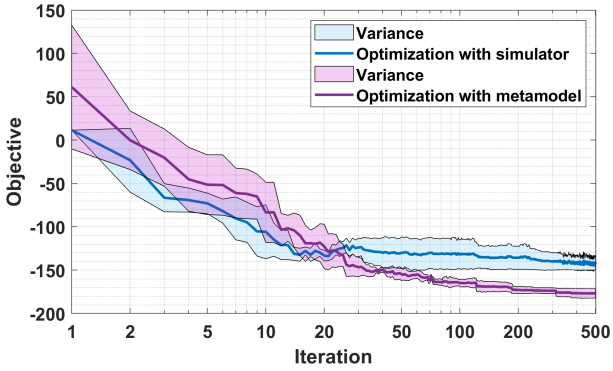

Figure 27: Optimization run for the second multi objective optimization of the two stage amplifier.

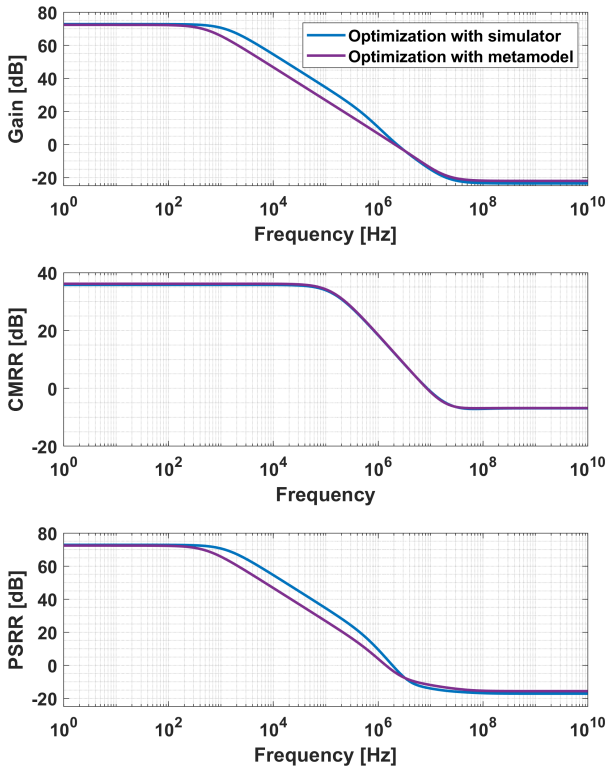

Figure 28: Best candidates for the previous run optimization.

5. Conclusions

To conclude the article results, firstly the modeling approach will be discussed. An improvement to previous

iterations of the proposed approach was done. By adding a threshold error to the modeled outputs and only taking into account points proposed by the models which are above the set threshold brings an improvement for the models which take more samples to model. The setting of the threshold can be either arbitrarily and adjusted during the sampling process if needed or either it can be set taking into account designer expertise and knowledge, the latter being the preferred approach.

A Bayesian optimization framework of analog integrated circuits was employed. The proposed approach used ML models in-the-loop instead of the actual simulation environment as an alternative aims to reduce heavy simulator use. The results which employed models with high accuracy (> 95%) had little to no significant difference in the end results between the two tested approaches. This might suggest that higher error models can be used (such as 10% relative RMSE) with a similar degree of success and should be further investigated.

In this article a much larger search space of the parameter design space is used than what is typically reported in literature, widths being varied between $[10\mu m; 500\mu m]$ and the lengths between $[0.5\mu m; 5\mu m]$. This can easily help implement designs for a wider range of applications. Similar approaches used in literature usually set lower ranges for example up to $100 - 200\mu m$ for the widths and up to $2\mu m$ for the lengths of the transistors.

The central advantage of this approach is same model reuse for different tasks or different phases of the analog design cycle (such as design/optimization and pre-silicon verification afterwards) which are reported in literature. Also, given that the optimization algorithm can sometimes be entrapped in local optima, the ML model approach has a net advantage given that in theory an infinite number of evaluations can be done in the bounded search space.

The proposed approach can be easily migrated to other adaptive sampling strategies and integrated with several more advanced optimization frameworks, such as evolutionary algorithms or reinforcement learning, which might further improve the outcomes regarding circuit performance and overall model accuracy. The proposed framework aligns with standard industry design practices while offering both implementation simplicity and high computational efficiency. The primary advantage of this approach lies in the separation of the data acquisition phase from the optimization pipeline. Once the initial machine learning model is trained, the heavy computational burden of the analog simulator is entirely bypassed, allowing any chosen optimization algorithm to evaluate candidate designs instantly. This modularity makes the framework exceptionally useful in scenarios where multiple operational amplifier designs must be generated rapidly under varying constraints.

While a traditional simulation-backed Bayesian optimization (BO) loop requires launching a time-consuming SPICE simulation at every single iteration, the proposed model-in-the-loop framework evaluates candidate designs instantly via mathematical model. Consequently, subsequent optimization runs or adjustments to the design objectives (such as modifying the Figure of Merit weights) can be executed with virtually zero additional computational overhead. This shift from simulation-heavy to model-driven optimization fundamentally scales the efficiency of the design automation workflow, enabling rapid iterative redesign and verification without exhausting limited hardware simulator licenses.

As further research which might be of interest to explore how modeling behaves when different thresholds are set for the multi output approach and how it relates to the overall modeling. Further on how the overall models error relate to the ability to predict viable circuit sizing values. As mentioned in the previous paragraph higher error models could still be able to reach a robust circuit sizing.

Conflict of Interest The authors declare no conflict of interest.

References

[1] V. Grosu, E. David, L. Goras, G. Pelz, "Modelling integrated circuits behavior using an active learning approach based on gaussian process regression", *"2023 International Semiconductor Conference (CAS)"*, pp. 245–248, 2023, doi:10.1109/CAS59036.2023.10303653.

[2] V. Grosu, L. Goras, E. David, G. Pelz, "Using neural network based active learning for modelling integrated circuits behavior", *"2023 International Symposium on Signals, Circuits and Systems (ISSCS)"*, pp. 1–4, 2023, doi:10.1109/ISSCS58449.2023.10190907.

[3] V. Grosu, E. David, L. Goras, G. Pelz, "Multiple output modelling of integrated circuits behavior using an active learning approach", *"2024 20th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)"*, pp. 1–4, 2024, doi:10.1109/SMACD61181.2024.10745387.

[4] V. Grosu, E. David, L. Goras, G. Pelz, "On the modelling possibilities of integrated circuits behavior using active learning principles", *Romanian Journal on Information Science and Technology (ROMJIST)*, vol. 27, no. 2, pp. 183–195, 2024, doi:10.59277/ROMJIST.2024.2.05.

[5] C. Vişan, M. Sieberer, H. Cucu, "Designer-like automated circuit sizing for multiloop ldo", *"2023 International Semiconductor Conference (CAS)"*, pp. 103–106, 2023, doi:10.1109/CAS59036.2023.10303725.

[6] C. Vişan, O. Pascu, M. Stănescu, E.-D. Şandru, C. Diaconu, A. Buzo, G. Pelz, H. Cucu, "Automated circuit sizing with multi-objective optimization based on differential evolution and bayesian inference", *Knowledge-Based Systems*, vol. 258, p. 109987, 2022, doi:https://doi.org/10.1016/j.knosys.2022.109987.

[7] A. Rusu, E. David, M.-D. Ţopa, V. Grosu, A. Buzo, G. Pelz, "On approaching multivariate ic pre-silicon verification using ml-based adaptive algorithms", *"2024 IEEE 30th International Symposium on On-Line Testing and Robust System Design (IOLTS)"*, pp. 1–3, 2024, doi:10.1109/IOLTS60994.2024.10616057.

[8] A. Rusu, E. David, V. Grosu, M. Topa, A. Buzo, B. Carbunescu, G. Pelz, "On multivariate electrical performance machine learning driven pre-silicon ic adaptive verification", *IEEE Access*, vol. 12, pp. 136436–136450, 2024, doi:10.1109/ACCESS.2024.3463393.

[9] D. Basso, L. Bortolussi, M. Videnovic-Misic, H. Habal, "Fast ml-driven analog circuit layout using reinforcement learning and steiner trees", *"2024 20th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)"*, pp. 1–4, 2024, doi:10.1109/SMACD61181.2024.10745442.

[10] P. P. Prajapati, M. V. Shah, "Automatic circuit design of cmos miller ota using cuckoo search algorithm", *International Journal of Applied Metaheuristic Computing (IJAMC)*, vol. 11, no. 1, pp. 36–44, 2020, doi:10.4018/IJAMC.2020010103.

[11] J. Huang, S. Zhang, C. Tao, F. Yang, C. Yan, D. Zhou, X. Zeng, "Bayesian optimization approach for analog circuit design using multi-task gaussian process", *"2021 IEEE International Symposium on Circuits and Systems (ISCAS)"*, pp. 1–5, 2021, doi:10.1109/ISCAS51556.2021.9401205.

[12] M. Fukuda, T. Ishii, N. Takai, "Op-amp sizing by inference of element values using machine learning", "*2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*", pp. 622–627, 2017, doi:10.1109/ISPACS.2017.8266553.

[13] G. İslamoğlu, T. O. Çakici, E. Afacan, G. Dündar, "Artificial neural network assisted analog ic sizing tool", "*2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*", pp. 9–12, 2019, doi:10.1109/SMACD.2019.8795293.

[14] P.-O. Beaulieu, E. Dumesnil, F. Nabki, M. Boukadoum, "Analog rf circuit sizing by a cascade of shallow neural networks", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 12, pp. 4391–4401, 2023, doi:10.1109/TCAD.2023.3282570.

[15] N. Kahraman, T. Yildirim, "Technology independent automated sizing methodology based on artificial neural networks: An application to cmos opamp design", "*2014 International Conference on Computational Science and Computational Intelligence*", vol. 1, pp. 476–481, 2014, doi:10.1109/CSCI.2014.85.

[16] N. Lourenço, E. Afacan, R. Martins, F. Passos, A. Canelas, R. Póvoa, N. Horta, G. Dundar, "Using polynomial regression and artificial neural networks for reusable analog ic sizing", "*2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*", pp. 13–16, 2019, doi:10.1109/SMACD.2019.8795282.

[17] P. P. Prajapati, M. V. Shah, "Automated sizing methodology for cmos miller operational transconductance amplifier", "*Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing*", pp. 301–308, Springer, 2018, doi:doi.org/10.1007/978-981-10-5699-4_29.

[18] W. Lyu, F. Yang, C. Yan, D. Zhou, X. Zeng, "Multi-objective bayesian optimization for analog/rf circuit synthesis", "*Proceedings of the 55th Annual Design Automation Conference*", DAC '18, Association for Computing Machinery, New York, NY, USA, 2018, doi:10.1145/3195970.3196078.

[19] B. He, S. Zhang, F. Yang, C. Yan, D. Zhou, X. Zeng, "An efficient bayesian optimization approach for analog circuit synthesis via sparse gaussian process modeling", "*2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*", pp. 67–72, 2020, doi:10.23919/DATE48585.2020.9116366.

[20] A. Lberni, M. A. Marktani, A. Ahaitouf, A. Ahaitouf, "Analog circuit sizing based on evolutionary algorithms and deep learning", *Expert Systems with Applications*, vol. 237, p. 121480, 2024, doi:10.1016/j.eswa.2023.121480.

[21] J. Gao, W. Cao, X. Zhang, "Rose: Robust analog circuit parameter optimization with sampling-efficient reinforcement learning", "*2023 60th ACM/IEEE Design Automation Conference (DAC)*", pp. 1–6, 2023, doi:10.1109/DAC56929.2023.10247991.

[22] Z. Li, A. C. Carusone, "Design and optimization of low-dropout voltage regulator using relational graph neural network and reinforcement learning in open-source sky130 process", "*2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*", pp. 01–09, 2023, doi:10.1109/ICCAD57390.2023.10323720.

[23] Y. Li, Y. Wang, Y. Li, R. Zhou, Z. Lin, "An artificial neural network assisted optimization system for analog design space exploration", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2640–2653, 2020, doi:10.1109/TCAD.2019.2961322.

[24] R. Mina, G. E. Sakr, H. Nassif, "Enhancing transistor sizing in analog ic design using a circuit-focused semi-supervised learning", "*2023 IEEE 4th International Multidisciplinary Conference on Engineering Technology (IMCET)*", pp. 223–228, 2023, doi:10.1109/IMCET59736.2023.10368264.

[25] Y. Uhlmann, T. Moldenhauer, J. Scheible, "Differentiable neural network surrogate models for gm/id-based analog ic sizing optimization", "*2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*", pp. 1–6, 2023, doi:10.1109/MLCAD58807.2023.10299834.

[26] Z. Zhao, L. Zhang, "Deep reinforcement learning for analog circuit sizing", "*2020 IEEE International Symposium on Circuits and Systems (ISCAS)*", pp. 1–5, 2020, doi:10.1109/ISCAS45731.2020.9181149.

[27] W. Lyu, F. Yang, C. Yan, D. Zhou, X. Zeng, "Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design", "*Proceedings of the 35th International Conference on Machine Learning*", vol. 80 of *Proceedings of Machine Learning Research*, pp. 3306–3314, PMLR, 2018.

[28] N. Takai, M. Fukuda, "Prediction of element values of opamp for required specifications utilizing deep learning", "*2017 International Symposium on Electronics and Smart Devices (ISESD)*", pp. 300–303, 2017, doi:10.1109/ISESD.2017.8253353.

[29] A. C. Sanabria-Borbón, S. Soto-Aguilar, J. J. Estrada-López, D. Allaire, E. Sánchez-Sinencio, "Gaussian-process-based surrogate for optimization-aided and process-variations-aware analog circuit design", *Electronics*, vol. 9, no. 4, 2020, doi:10.3390/electronics9040685.

[30] C. Vişan, M. Boldeanu, G. Nicolae, H. Cucu, C. Burileanu, A. Buzo, "Evolutionary bayesian optimization for automated circuit sizing", *Knowledge-Based Systems*, vol. 318, p. 113483, 2025, doi:https://doi.org/10.1016/j.knosys.2025.113483.

[31] F. Daylak, S. Ozoguz, "Automated neural network-based optimization for enhancing dynamic range in active filter design", *Electronics*, vol. 14, no. 4, 2025, doi:https://doi.org/10.3390/electronics14040786.

[32] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, *et al.*, "Machine learning for electronic design automation: A survey", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 26, no. 5, pp. 1–46, 2021, doi:https://doi.org/10.1145/3451179.

[33] D. V. Kochar, H. Wang, A. P. Chandrakasan, X. Zhang, "Ledro: Llm-enhanced design space reduction and optimization for analog circuits", pp. 141–148, 2025, doi:10.1109/ICLAD65226.2025.00011.

[34] Y. Yin, Y. Wang, B. Xu, P. Li, "Ado-llm: Analog design bayesian optimization with in-context learning of large language models", "*Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*", pp. 1–9, 2024, doi:10.1145/3676536.3676816.

[35] N. S. K. Somayaji, P. Li, "Llm-uso: Large language model-based universal sizing optimizer", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 45, no. 4, pp. 1907–1920, 2026, doi:10.1109/TCAD.2025.3604643.

[36] C. Liu, E. A. Olowe, D. Chitnis, "Llm-based ai agent for sizing of analog and mixed signal circuit", pp. 90–94, 2025, doi:10.1109/NewCAS64648.2025.11107079.

[37] C. Zhang, J. Jiang, Y. Zhao, "Euclidean space with orbital angular momentum", "*2019 IEEE International Conference on Communications Workshops (ICC Workshops)*", pp. 1–6, 2019, doi:10.1109/ICCW.2019.8756875.

[38] C. Zhang, J. Jiang, Y. Zhao, X. Jiang, "New degrees of freedom for beamforming manipulation in mimo transmission with oam", "*2019 IEEE Globecom Workshops (GC Wkshps)*", pp. 1–6, 2019, doi:10.1109/GCWkshps45667.2019.9024467.

[39] H. Niu, J. An, T. Wu, J. Chen, Y. Zhao, Y. Liang Guan, M. Di Renzo, M. Debbah, G. K. Karagiannidis, H. Vincent Poor, C. Yuen, "Introducing meta-fiber into stacked intelligent metasurfaces for mimo communications: A low-complexity design with only two layers", *IEEE Transactions on Wireless Communications*, vol. 25, pp. 3016–3032, 2026, doi:10.1109/TWC.2025.3600915.

[40] J. N. Fuhg, A. Fau, U. Nackenhorst, "State-of-the-art and comparative review of adaptive sampling methods for kriging", *Archives of Computational Methods in Engineering*, vol. 28, pp. 2689–2747, 2021, doi:https://doi.org/10.1007/s11831-020-09474-6.

[41] C. K. Williams, C. E. Rasmussen, *Gaussian processes for machine learning*, Cambridge, MA: MIT press, 2006, doi:https://doi.org/10.7551/mitpress/3206.001.0001.

[42] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, "Algorithms for hyper-parameter optimization", vol. 24, pp. 2546–2554, 2011.

[43] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization", *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016, doi:10.1109/JPROC.2015.2494218.

[44] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, D. Zhou, "An efficient bayesian optimization approach for automated optimization of analog circuits", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1954–1967, 2018, doi:10.1109/TCSI.2017.2768826.

[45] MathWorks, "Matlab version: 9.13.0 (r2023b)", 2023.

[46] LinearTechnology, "Ltspice (version 24.1)", 2025.

[47] V. Plevris, G. Solorzano, "A collection of 30 multidimensional functions for global optimization benchmarking", *Data*, vol. 7, no. 4, p. 46, 2022, doi:https://doi.org/10.3390/data7040046.