Ordinary Kriging Metamodel-Assisted Ant Colony Algorithm for Fast Analog Design Optimization

Oghenekarho Okobiah[1], Saraju P. Mohanty[2], Elias Kougianos[3]
NanoSystem Design Laboratory (NSDL, http://nsdl.cse.unt.edu)[1,2,3]
Computer Science and Engineering, University of North Texas, USA[1,2]
Engineering Technology, University of North Texas, USA[3]
Email-ID: oghenekarhookobiah@my.unt.edu[1], saraju.mohanty@unt.edu[2], elias.kougianos@unt.edu[3].

***Abstract*—This paper explores an ordinary Kriging based metamodeling technique that allows designers to create a model of a circuit with very good accuracy, while greatly reducing the time required for simulations. Regression and interpolation based methods have been researched extensively and are a commonly used technique for creating metamodels. However, they do not take into account the effect of correlation between design and process parameters, which are critical in the nanoscale regime. Kriging provides an improved metamodeling technique which takes into effect correlation effects during the metamodel generation phase. The ordinary Kriging metamodels are subjected to an Ant Colony Optimization (ACO) algorithm that enables fast optimization of the circuit. This design methodology is evaluated on a sense amplifier circuit as a case study. The results show that the Kriging based metamodels are very accurate and the ACO based algorithm optimizes the sense amplifier precharge time with power consumption as a design constraint in an *average time of 3.7 minutes (optimization on the metamodel), compared to 72 hours (optimization on the SPICE netlist)*.**

***Keywords*-Nano-CMOS, Sense Amplifier, Robust Design, Metamodeling, Kriging Methods**

I. Introduction

Analog simulations use very accurate models and have the ability to accurately estimate performance measures. However, with the scaling of designs in the deep nanometer region and the increase in the level of complexity, exhaustive design space exploration through computer simulation has become more daunting and most often impractical. In nano-CMOS designs, the effects of process variation are increasingly becoming more dominant. These factors make design optimization very difficult and time consuming. Metamodeling has been one researched and applied solution to reduce the time burden of computer simulation while keeping the accuracy to an acceptable level.

Metamodels, by definition, are an approximate description of the performance response of design models. Essentially, metamodels are models of a simulation model (hence the term *meta*) [1], [2]. The use of metamodels abstracts the time complexity of analog simulations while capturing in detail the behavior of the design. This gives the designer quick access to a sufficiently accurate design space exploration tool. Commonly used metamodeling techniques include Response Surface Modeling (RSM), linear and low-order polynomial regression techniques [3], [4], [1], and neural networks [5], [6], [7]. Linear and low-order polynomial regression techniques generally provide a better fit for local neighborhoods but are less accurate for global design spaces [2], [8]. Due to the oscillatory characteristic of polynomial fits, designs with rapidly changing data points are not well fitted which is the case with nano-CMOS designs [9]. In generating the metamodels, regression techniques assume the errors due to variation across the design space are random and thus equally approximate the error over points on the fitting surface. For many design processes, this error is not random but is correlated with other process and design parameters. In nano-CMOS designs, the correlation can vary significantly across the global design space and even locally, hence making correlation effects a significant factor in accurate metamodeling. Therefore, there is need to implement a metamodel which captures these correlation effects to improve its accuracy.

This paper proposes a Kriging based metamodeling technique that generates accurate metamodels with the error due to correlation taken into consideration. Kriging techniques were originally introduced in the early 1950's in geostatistical analysis and have been applied to many other fields [10], [11], [12] and recently in VLSI [13], [14]. Kriging based techniques generate interpolating functions for each estimated point using the correlation effect between design points in the local space. Each point response is estimated with a unique set of weights. One major improvement of Kriging over conventional regression is that the estimated response at sample points is the same as the actual response at these points (although it may differ in-between). The generated Kriging metamodel is then subjected to an Ant Colony Optimization (ACO) based algorithm for fast optimization. [15], [16], [17]. Originally used for discrete optimization problems, there has been recent research to adapt ACO for continuous functions [16], [18], [17], [19]. *The novel contributions of this paper are the application of Kriging to generate metamodels that account for correlation effects in nano-CMOS designs, and the use of ACO-based algorithms for fast optimization*.

The rest of this paper is organized as follows. In section II, relevant prior research is summarized. Section III discusses

[0]This research is supported in part by NSF awards CNS-0854182 and DUE-0942629 and SRC award P10883.

the background of Kriging methods. The proposed algorithm is presented in Section IV. In section V experimental results are presented. Conclusions and directions for future research are presented in Section VI.

II. Related Prior Research

The research and application of metamodeling as a design methodology has been proposed in the literature before. A popular metamodeling technique is low-order polynomial regression [12], [3], [4], [1]. In [3], an analysis of different metamodels generated with a number of sampling techniques is presented. While low-order polynomial regression techniques are suitable for generating response surfaces, they are only accurate for local neighborhoods and present poor fitting for global design spaces [9], [2], [5], [20]. Regression techniques average the errors in calculating weights over the design space. This creates an oscillating effect for fast changing data, especially when the autocorrelation error between design points varies significantly. Geometric programming is presented in [20]. This solves convex problems deduced from circuit design equations expressed in polynomial forms. The approximations made in deducing the circuit equations reduce the accuracy, even though they are suitable for global optimization. Neural Networks (NN) have also been explored for metamodel generation [6], [5], [7], [21]. In optimizing metamodels for design objectives, Simulated Annealing and Tabu Search algorithms are explored in [3]. In [22], simulated-annealing algorithm is used to optimize a sense amplifier over Kriging metamodels. ACO originally was proposed for discrete combinatorial problems and is also being actively explored for application on continuous problems [15], [16], [17]. This paper implements an ACO based algorithm.

III. Ordinary Kriging Method for Metamodeling

The fundamental idea behind Kriging methods is that the predicted outputs are weighted averages of sampled data. The weights are unique to each predicted point and are a function of the distance between the point to be predicted and observed points. The weights are chosen so that the prediction variance is minimized [23], [10].

The general expression of a Kriging model is of the following form [22]:

$$y(\mathbf{x_0}) = \sum_{j=1}^{L} \lambda_j B_j(\mathbf{x}) + z(\mathbf{x}), \tag{1}$$

where $y(\mathbf{x_0})$, is the predicted response at design point $(\mathbf{x_0})$ $\{B_j(\mathbf{x}), j = 1, \cdots, L\}$ is a specific set of basis functions over the design domain D_N, λ_j are fitting coefficients (also known as weights) to be determined and $z(\mathbf{x})$ is the random error. Kriging differs from common least squares based approaches in that $z(\mathbf{x})$ is assumed to be a random process and not independent, unique to each weight and not distributed identically. This process has mean μ, variance σ^2, and correlation function $r(\mathbf{s}, \mathbf{t}) = \mathrm{Corr}(z(\mathbf{s}), z(\mathbf{t}))$ (called the *variogram* in geophysics) which are assumed known.

The variogram is used to derive the Kriging weights, λ_j. The autocorrelation of the design points is characterized by the covariance function [24]. The weights are chosen so that the Kriging variance is minimized. There are different variations of the Kriging model which include simple, ordinary and universal Kriging. The proposed metamodel in this paper is based on the ordinary Kriging technique which assumes a mean that is constant in the local domain of a predicted point.

In ordinary Kriging techniques, the weights are chosen to minimize the variance under the unbiasedness constraint $E(z(\hat{\mathbf{x}}) - z(\mathbf{x})) = 0$. $\sum_{j=1}^{n} \lambda_j = 1$, Hence the weights are given by the following expression:

$$\begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu \end{pmatrix} = \Gamma^{-1} \begin{pmatrix} \gamma(e_1, e_0) \\ \vdots \\ \gamma(e_n, e_0) \\ 1 \end{pmatrix}, \tag{2}$$

where μ is a Lagrange multiplier. Γ is the covariance matrix of the observed points and for ordinary Kriging is given as:

$$\Gamma = \begin{pmatrix} \gamma(e_1, e_1) & \cdots & \gamma(e_1, e_n) & 1 \\ \vdots & \ddots & \vdots & 1 \\ \gamma(e_n, e_1) & \cdots & \gamma(e_n, e_n) & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \tag{3}$$

where $\gamma(e_1, e_2) = E(|Z(e_1) - Z(e_2)|^2)$.

Estimation of the correlation between sampled points and a predicted point is done with the semivariogram model. Based on the nature of the observed data points, the empirical model could be fit to either spherical, linear, Gaussian or exponential theoretical models. The smoothness of the predicted points is affected by the theoretical model used. A steeper model reduces the smoothness because it places more weight on closer neighbors. The most common model used is the spherical and is expressed by the following expression:

$$\gamma(h) = C_0 + C\left(\frac{3h}{2a} - \frac{1}{2}\left(\frac{h}{a}\right)^3\right) \text{ for } 0 < h \leq a, \tag{4}$$

where C_0 and C are fitting coefficients and a is a shape factor.

IV. The Proposed Design Flow using Kriging Metamodels and Ant Colony Algorithm

Algorithm 1 shows the complete flow of the proposed design that incorporates ordinary Kriging and the ACO algorithm. A sense amplifier used in conventional DRAMs is used to demonstrate the proposed design optimization methodology.

Ant Colony Optimization (ACO) algorithms are inspired by the foraging behavior of ant species and are metaheuristic random searching algorithms. The majority of ACO algorithms have been applied to discrete combinatorial optimization problems such as routing, scheduling, and timetabling [15], [25]. A classic example in demonstrating the ACO algorithm is the Traveling Salesman Problem (TSP). The basic stages of ACO based algorithms are metaheuristic which means they can be easily modified for application on a wide range of optimization problems [25]. Recently, there has been more

Algorithm 1 The Proposed Design Flow.

Create baseline design.
Identify Figures of Merit (FoMs) (verify functionality).
Create physical layout.
Perform DRC/LVS and RLCK extraction.
Identify design parameters and parameterize netlist.
$Metamodel\ Generation$.
Perform Latin HyperCube Sampling to generate design points for metamodel.
Generate Kriging metamodels using mGStat tool.
$Optimization$.
while ($Optimization\ objective\ not\ met$) **do**
    Perform ACO based algorithm.
end while
return $Optimized\ Design$.

Algorithm 2 ACO Based Heuristic Algorithm Setup.

1: Initialize parameters.
2: Set termination conditions.
3: Generate random node ants.
4: Perform pheromone update.
5: **while** ($Termination\ condition\ not\ met$) **do**
6:     Generate node ants with pheromone probability.
7:     Update pheromone.
8: **end while**
9: **return** $result$.

Algorithm 3 ACO Heuristic Algorithm for Sense Amplifier.

Initialize $number\ of\ ants\ (solutionset)$
Initialize iteration counter: $counter \leftarrow 0$
Start with initial baseline solution $\widehat{(SA_i)}$
Generate metamodel functions for each FoM of $\widehat{(SA_i)}$ with Ordinary Kriging.
Consider the objective of interest T_{PC_i}
Generate random ant nodes AL, W_i, where i = 1,2,...,N_{ant}.
Assign initial pheromone, τ_i
$counter \leftarrow max\_Iteration$
while ($counter > 0$) **do**
    Generate ant solutions T_{PC_i}
    Rank solutions $\widehat{(SA_i)}$ in set from best to worst.
    Update pheromone, increase pheromone for best solution and evaporate pheromone for all others
    $result \leftarrow T_{PC_i}$
    Generate new ant nodes AL, W_i,
    $counter \leftarrow counter - 1$
end while
return $result$

research to extend the application of the ACO algorithms to continuous function problems [18], [16], [17], [19].

The basic characteristics of ACO algorithms include the incremental construction of solutions and the use of pheromone updates to guide point explorations. The basic stages for the ACO metaheuristic are as follows:

1) Initialize variables and set conditions
2) Construct ant nodes
3) Perform local search (optional)
4) Update pheromones

After the initialization step, the algorithm iterates between steps 2-4 until the condition for termination is met.

ACO algorithms for continuous problems differ from discrete optimizations in the selection of ant nodes. In [18], [25], [16], [17], [26], [27] several modifications to ACO for continuous function optimizations have been presented. One major way of adapting the ACO for continuous functions is dividing the solution space into different intervals and having each ant or node search an interval for an optimal solution. The nodes can also directly search the continuous function. The proposed algorithm in this paper is most closely related to the approach proposed in [16], where the design space is sampled for search nodes. Where the proposed algorithm in this paper differs from [16] is that each decision variable is a given node and not a set of nodes. In this case a solution by a node is also assumed a "path" traversed by an ant, keeping in line with the original ACO framework. With the discretization of the continuous problem space, the ACO algorithm can be easily used to generate optimal solutions.

The pheromones are updated using the following:

$$\tau_i = (1 - \rho)\ .\ \tau_i + \rho\ .\ \Delta\tau_i, \tag{5}$$

where ρ is the rate of evaporation, and the pheromone τ_i in the i-th iteration is updated only for the best solutions. The criteria and number of best solutions can be decided independently for each optimization problem.

The proposed algorithm is shown in Algorithm 2. The details of implementation are included in steps 6 and 7. For each iteration, a new set of ant solutions (feasible design points) is generated. The solutions are updated with an evaporating pheromone. The best solution is however updated with more pheromone thus increasing the probability of the path (node) being traversed by more search ants. The iterations are continued until the termination condition is met which, in this case, is a maximum number of allowed iterations. The speed of convergence of the ACO algorithms can be controlled by the rate of pheromone update. A series of runs show that for this circuit, an average of 100 iterations converge the algorithm to an optimal solution.

The optimization of the Kriging metamodels is done with the proposed ACO base algorithm. For this algorithm (3), a metamodel generated with 500 sample data points is used. The optimization goal is to minimize the precharge time T_{PC}, without violating the power constraint. The parameters are initialized for random design points of the NMOS length and width, L_n and W_n. The algorithm updates the pheromones by evaporation but remembers the best solution, thereby increasing the probability of more search ants in that direction.

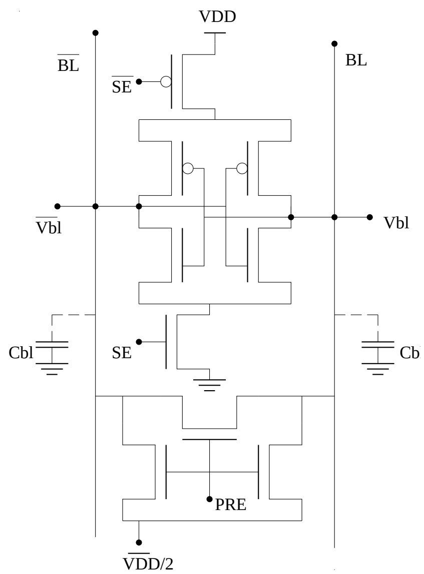


Fig. 1. Sense amplifier circuit with sizing for 45nm CMOS.

V. Experimental Setup and Results

A. Case Study Circuit

The experimental circuit used in this paper is a sense amplifier used in DRAMs. The circuit schematic is shown in Fig. 1. The sense amplifier is crucial to the correct operation of the DRAM. Its performance is significantly affected by process variation which has to be taken into account during the design process, making it an ideal example circuit for this study.

The circuit is characterized for Figures of Merit (FoMs) which include the **precharge and equalization time (T_{PC})** — the time used to equally precharge the bitlines for sense operations, **sense delay (T_{SD})** — the time it takes for a sufficient voltage sharing to appear on the bitlines, **power consumption(P_{SA})** — average power consumed including static and dynamic power and **sense margin (V_{SM})** — the minimum amount of voltage that must appear on the bitlines for correct amplification, as discussed in detail in [28].

Functional simulation of the sense amplifier is shown in Fig. 2. It shows the various operating states of the sense amplifier: the write, hold and read stages. Initial transistor dimensions are based on nomimal values for a 45 nm process design kit. L_n and W_n are used as design parameters (the parametric analysis shows the NMOS transistors dominate the FoM). The design objective could be single or multi-objective; in this case a single objective with design constraint is presented. For this design, the optimization objective is the precharge time T_{PC}, while power consumption is used as a constraint. T_{PC} is a significant FoM in the overall speed of the DRAM.

B. Metamodel Generation and Accuracy

This section presents the methodology for the metamodel generation and analyzes their accuracy. As a baseline for comparison for simulation time, a response surface was generated. A total of 10,000 simulations were run to created the "golden"

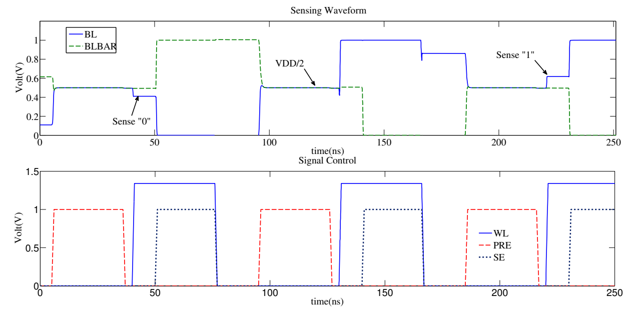


Fig. 2. Sense amplifier functional simulation.

surfaces which are not shown in this paper for brevity. For ordinary Kriging, the sample data points were generated using the Latin Hypercube Sampling (LHS) technique. Three sets of ordinary Kriging metamodels were generated with the mGstat toolbox [29] using 100, 200 and 500 sampling points. For each set, a metamodel was generated for each FoM. Figure 3 shows the surface plots generated through the ordinary Kriging technique.

A statistical summary of the accuracy of the metamodels is shown in Table II. The metamodel predictions are compared to the exhaustive surfaces via the Mean Square Error (MSE), Root Mean Square Error (RMSE), the correlation coefficient R^2 and the standard deviation (STD). MSE and RMSE are defined by the following expressions:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \widehat{Y}_i\right)^2, \tag{6}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \widehat{Y}_i\right)^2}, \tag{7}$$

where the summation runs over the N points used to generate the metamodels, and Y_i and $\widehat{Y}_i$ are the actual and predicted responses, respectively at point i.

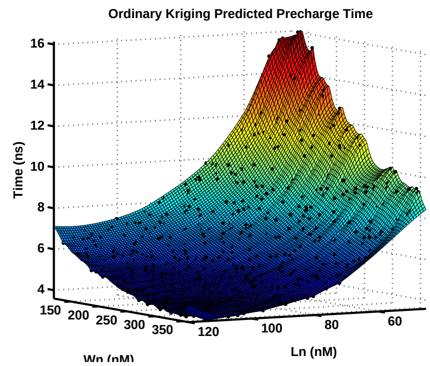
Table II shows that the metamodels created are quite accurate with very low RMSE values and average R^2 values in the range of 0.98–0.99 with the R^2 values for P_{SA} a little lower than the other FoMs. As expected, the metamodels generated from 500 LHS sampling points are generally more accurate than the ones generated from 200 or 100 sampling points. The time required for the metamodel generation is 3.69 minutes which is significantly lower than the 72 hours required for an exhaustive simulation.

C. Optimization Results

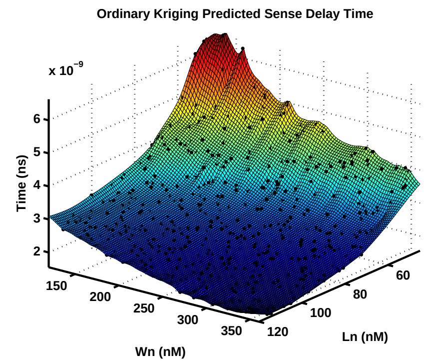
The values for the optimized design are shown in Table III. T_{PC} has been reduced by 65. 77 % while P_{SA} was increased by 0.85 %. The final design parameters for L_n and W_n are 65 nm and 300 nm, respectively. The final design also increases the area cost for the physical layout by 23.10%. The total average time taken for design optimization using the sense amplifier as the case study circuit is 3.9 minutes. The bulk of the time is consumed in the metamodel generation as the

TABLE I
CHARACTERIZATION OF FIGURES OF MERIT.

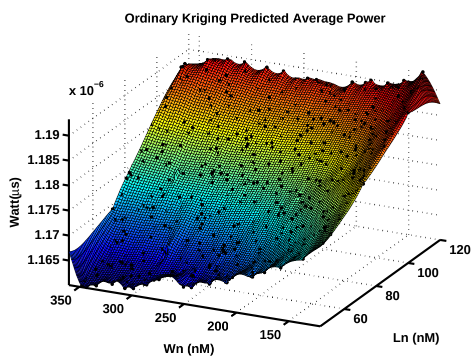
Design	Precharge time, T_{PC} (ns)	Sense delay, T_{SD} (ns)	Power, P_{SA} (μW)	Sense Margin, V_{SM} (mV)	Area μm^2
Schematic	18.02	7.46	1.16	29.33	-
Layout	18.20	7.45	1.17	29.25	6.045



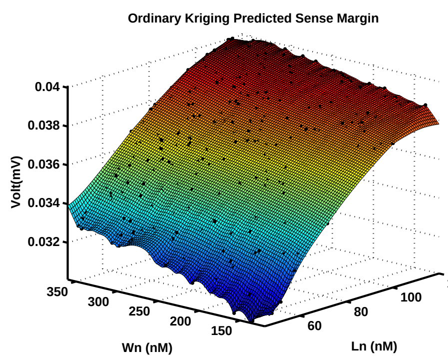
(a) Response for precharge time



(b) Response for sense delay



(c) Response for average power



(d) Response for sense margin

Fig. 3. Kriging predicted surfaces for FoMs using 500 $\{L_n,W_n\}$ sample points

ACO algorithm converges in an average time of 1.36 sec. The process from design space exploration to optimization is reduced by a factor of approximately $10^3\times$. The final (optimized) layout is shown in Fig. 4.

VI. CONCLUSIONS AND FUTURE RESEARCH

This paper presented a new design methodology that combines Kriging techniques for metamodel generation and ACO based algorithms. Kriging provides an improved method for metamodeling that takes into account the correlation effects between points in the design and process space for nano-CMOS designs. It also improves the accuracy of the metamodels. It has very low RMSE when compared to exhaustive surfaces but it takes longer time to generate than conventional methods. ACO based algorithms have also been applied to the metamodels for fast optimizations. The precharge time T_{PC} is improved by 65.77 % within the power constraints. This methodology has been demonstrated using two design parameters. In future research, this methodology will be extended to more parameters and multi-objective optimization goals.

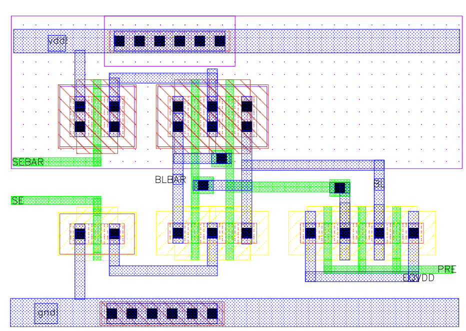


Fig. 4. Final physical design for sense amplifier in 45nm technology.

TABLE III
CHARACTERIZATION OF FIGURES OF MERIT.

Design	Precharge time, T_{PC} (ns)	Sense delay, T_{SD} (ns)	Power, P_{SA} (μW)	Sense Margin, V_{SM} (mV)	Area μm^2
Schematic	18.02	7.46	1.16	29.33	-
Layout	18.20	7.45	1.17	29.25	4.294
Optimized	6.23	2.58	1.18	35.56	5.286
Improvement	65.77 %	65.37 %	-0.85 %	21.57 %	23.10 %

TABLE II
STATISTICAL ANALYSIS OF THE KRIGING PREDICTED RESPONSES

FoMs	Ordinary Kriging		
Samples	100	200	500
Precharge			
MSE	2.20×10^{-19}	5.23×10^{-20}	1.84×10^{-20}
$RMSE$	4.69×10^{-10}	2.29×10^{-10}	1.36×10^{-10}
R^2	0.9650	0.9917	0.9971
STD	4.32×10^{-10}	2.03×10^{-10}	1.27×10^{-10}
Sense Delay			
MSE	4.22×10^{-20}	1.16×10^{-20}	4.75×10^{-21}
$RMSE$	2.05×10^{-10}	1.08×10^{-10}	6.89×10^{-11}
R^2	0.9529	0.9871	0.9947
STD	1.89×10^{-10}	9.39×10^{-11}	6.26×10^{-11}
Power			
MSE	1.84×10^{-17}	1.08×10^{-17}	1.02×10^{-11}
$RMSE$	3.44×10^{-09}	3.29×10^{-09}	3.20×10^{-09}
R^2	0.8384	0.8525	0.8606
STD	1.19×10^{-09}	9.47×10^{-10}	6.06×10^{-10}
Sense Margin			
MSE	1.12×10^{-07}	3.41×10^{-08}	9.47×10^{-09}
$RMSE$	3.35×10^{-04}	1.85×10^{-04}	9.73×10^{-05}
R^2	0.9804	0.9940	0.9983
STD	2.98×10^{-04}	1.62×10^{-04}	9.05×10^{-05}

REFERENCES

[1] W. Biles, J. Kleijnen, W. van Beers, and I. van Nieuwenhuyse, "Kriging Metamodeling in Constrained Simulation Optimization: An Explorative Study," in *Winter Simulation Conference*, 2007, pp. 355 –362.

[2] B. Ankenman, B. Nelson, and J. Staum, "Stochastic Kriging for Simulation Metamodeling," in *Winter Simulation Conference*, 2008, pp. 362–370.

[3] O. Garitselov, S. Mohanty, E. Kougianos, and P. Patra, "Nano-CMOS Mixed-Signal Circuit Metamodeling Techniques: A Comparative Study," in *International Symposium on Electronic System Design*, 2010, pp. 191 –196.

[4] G. Dellino, J. Kleijnen, and C. Meloni, "Robust Simulation-Optimization Using Metamodels," in *Winter Simulation Conference*, 2009, pp. 540 – 550.

[5] A. Khosravi, S. Nahavandi, and D. Creighton, "Developing Optimal Neural Network Metamodels Based on Prediction Intervals," in *International Joint Conference on Neural Networks*, 2009, pp. 1583 –1589.

[6] C. W. Zobel and K. B. Keeling, "Neural Network-based Simulation Metamodels for Predicting Probability Distributions," *Comput. Ind. Eng.*, vol. 54, pp. 879–888, May 2008.

[7] I. Sabuncuoglu and S. Touhami, "Simulation Metamodelling With Neural Networks: An Experimental Investigation." *International Journal of Production Research,*, vol. 40, no. 11, pp. 2483–2505, 2002.

[8] J. Staum, "Better Simulation Metamodeling: The Why, What, and How of Stochastic Kriging," in *Winter Simulation Conference (WSC), Proceedings of the 2009*, 2009, pp. 119 –133.

[9] E. Siah, T. Ozdemir, J. Volakis, P. Papalambros, and R. Wiese, "Fast Parameter Optimization Using Kriging Metamodeling [antenna EM modeling/simulation]," in *Antennas and Propagation Society International Symposium, 2003. IEEE*, vol. 2, June 2003, pp. 76 – 79 vol.2.

[10] W. Van Beers, "Kriging Metamodeling in Discrete-Event Simulation: An Overview," in *Proceedings of the Winter Simulation Conference*, 2005.

[11] B. Harrington, Y. Huang, J. Yang, and X. Li, "Energy-Efficient Map Interpolation for Sensor Fields Using Kriging," *Mobile Computing, IEEE Transactions on*, vol. 8, no. 5, pp. 622 –635, May 2009.

[12] M. Zakerifar, W. Biles, and G. Evans, "Kriging Metamodeling in Multi-objective Simulation Optimization," in *Winter Simulation Conference (WSC), Proceedings of the 2009*, Dec. 2009, pp. 2115 –2122.

[13] G. Yu and P. Li, "Yield-Aware Analog Integrated Circuit Optimization Using Geostatistics Motivated Performance Modeling," in *IEEE/ACM International Conference on Computer-Aided Design*, 2007, pp. 464 – 469.

[14] H. You, M. Yang, D. Wang, and X. Jia, "Kriging Model Combined with Latin Hypercube Sampling for Surrogate Modeling of Analog Integrated Circuit Performance," in *International Symposium on Quality of Electronic Design*, 2009, pp. 554 –558.

[15] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant Algorithms for Discrete Optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.

[16] X.-M. Hu, J. Zhang, H. S.-H. Chung, Y. Li, and O. Liu, "SamACO: Variable Sampling Ant Colony Optimization Algorithm for Continuous Optimization," *Trans. Sys. Man Cyber. Part B*, vol. 40, pp. 1555–1566, December 2010.

[17] L. Chen, H. Sun, and S. Wang, "Solving Continuous Optimization Using Ant Colony Algorithm," in *Proceedings of the 2009 Second International Conference on Future Information Technology and Management Engineering*, 2009, pp. 92–95.

[18] K. Socha and M. Dorigo, "Ant Colony Optimization for Continuous Domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155 – 1173, 2008.

[19] J. Zhang, W.-N. Chen, J. hui Zhong, X. Tan, and Y. Li, "Continuous Function Optimization using Hybrid Ant Colony Approach With Orthogonal Design Scheme," in *Proceeding of SEAL 2006, LNCS 4247, 2006*, 2006, pp. 126–133.

[20] V. Aggarwal, "Analog Circuit Optimization using Evolutionary Algorithms and Convex Optimization," Master's thesis, Massachusetts Institute of Technology, May 2007.

[21] R. A. Kilmer, A. E. Smith, and L. J. Shuman, "Computing Confidence Intervals for Stochastic Simulation Using Neural Network Metamodels," *Computers and Industrial Engineering*, vol. 36, no. 2, pp. 391 – 407, 1999.

[22] O. Okobiah, S. P. Mohanty, E. Kougianos, and O. Garitselov, "Kriging-Assisted Ultra-Fast Simulated-Annealing Optimization of a Clamped Bitline Sense Amplifier," in *Proceedings of the 25th IEEE International Conference on VLSI Design*, 2012.

[23] N. A. C. Cressie, *Statistics for Spatial Data*. New York: Wiley, 1993.

[24] G. Bohling, "Kriging," Kansas Geological Survey, Tech. Rep., 2005.

[25] M. Dorigo, M. Birattari, and T. Sttzle, "Ant Colony Optimization – Artificial Ants as a Computational Intelligence Technique," *IEEE Comput. Intell. Mag.*, vol. 1, pp. 28–39, 2006.

[26] L. Hong and X. Shibo, "On Ant Colony Algorithm for Solving Continuous Optimization Problem," in *Proceedings of the 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2008, pp. 1450–1453.

[27] X. Xue, W. Sun, and C. Peng, "Improved Ant Colony Algorithm for Continuous Function Optimization," in *Control and Decision Conference (CCDC), 2010 Chinese*, May 2010, pp. 20 –24.

[28] O. Okobiah, S. P. Mohanty, E. Kougianos, and M. Poolakkaparambil, "Towards Robust Nano-CMOS Sense Amplifier Design: A Dual-Threshold versus Dual-Oxide Perspective," in *Proceedings of the 21st ACM Great Lakes Symposium on VLSI*, 2011, pp. 145–150.

[29] *mGstat: A Geostatistical MATLAB Toolbox*. [Online]. Available: mgstat.sourcefourge.net