Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

A NOVEL SEQUENTIAL DESIGN STRATEGY FOR GLOBAL SURROGATE MODELING

Karel Crombecq
Luciano De Tommasi

University of Antwerp
Middelheimlaan 1
Antwerp, 2000, BELGIUM

Dirk Gorissen
Tom Dhaene

Ghent University - IBBT
Sint Pietersnieuwstraat 41
Ghent, 9000, BELGIUM

ABSTRACT

In mathematical/statistical modeling of complex systems, the locations of the data points are essential to the success of the algorithm. Sequential design methods are iterative algorithms that use data acquired from previous iterations to guide future sample selection. They are often used to improve an initial design such as a Latin hypercube or a simple grid, in order to focus on highly dynamic parts of the design space. In this paper, a comparison is made between different sequential design methods for global surrogate modeling on a real-world electronics problem. Existing exploitation and exploration-based methods are compared against a novel hybrid technique which incorporates both an exploitation criterion, using local linear approximations of the objective function, and an exploration criterion, using a Monte Carlo Voronoi tessellation. The test results indicate that a considerable improvement of the average model accuracy can be achieved by using this new approach.

1 INTRODUCTION

The simulation of complex systems with multiple input and output parameters can be a time-consuming process. For example, Ford Motor Company reported on a crash simulation for a full passenger car that takes 36 to 160 hours to compute (Gorissen et al. 2007). Furthermore, the system under study is often a black box, with little or no additional information available about its inner working except for the output it generates.

The goal of *global* surrogate modeling or metamodeling (as opposed to *local* surrogate modeling) is to create a model that mimics the original system, but can be computed much faster. This model is constructed by performing simulations (called samples) at key points in the domain, analyzing the results, and creating a model that approximates the samples and the overall system behavior quite well. There are a wide variety of model types available, and which one is most suitable depends largely on the system that is to be modeled. Popular choices are polynomial and rational functions (Hendrickx and Dhaene 2005; Deschrijver, Dhaene, and Broeckhove 2004), Kriging models (Simpson, Lin, and Chen 2001; Beers 2005), neural networks (Balewski and Mrozowski 2004; Panayiotou, Cassandras, and Gong 2000) and radial basis functions (RBF) models (Lamecki, Kozakowski, and Mrozowski 2004). Once the model is constructed, it can be used to perform optimization and sensitivity analysis and to gain insight in the global structure and behavior of the function (Eres et al. 2003; Zhao and Knight 2004; Barton 1998).

Please note that *global* surrogate modeling differs from *local* surrogate modeling in the way the surrogate models are employed. In *local* surrogate modeling, local models are used to guide the optimization algorithm towards a global optimum. The local models are discarded afterwards. In *global* surrogate modeling, the goal is to create a model that approximates the behaviour of the simulator on the entire domain, so that the surrogate model can then be used as a full replacement for the original simulator, or can be used to explore the design space. Thus, the goal of global surrogate modeling is to overcome the long computational time of the simulator by providing a fast but accurate approximation, based on a one-time upfront modeling effort. This is fundamentally different from local surrogate modeling, where the goal is to find the optimum, and local surrogate models are a useful tool for achieving this goal. In this paper, we are only concerned with global surrogate modeling.

The simulation of one sample can be a very costly operation, so it is important that they are chosen carefully. Samples should be distributed over the design space in such a way as to provide a maximum amount of information about the behavior

of the system. A configuration of sample locations which tries to achieve this is called an experimental design. Examples of experimental designs are grids, Latin hypercubes and uniform designs. An excellent overview of experimental designs is given by Santner, Williams, and Notz (2003).

In traditional design of experiments (DOE), such an experimental design is fed to the simulator, which evaluates the selected sample locations, and a model is built using this data. This is essentially a one-shot approach, as all the samples are chosen at once and the modeling algorithm proceeds from there, without evaluating any additional samples later. The advantages of these methods are that they can be easily implemented and provide a good coverage of the domain without incorporating any prior knowledge of the system that is to be modeled.

Sequential design (which is also known as adaptive sampling (Lehmensiek, Meyer, and Müller 2002) or active learning (Sugiyama 2006)) further improves on this approach by making the algorithm into an iterative process. Sequential design methods analyze data (models and samples) from previous iterations in order to select new samples in areas that are more difficult to approximate, resulting in a more efficient distribution of samples compared to traditional design of experiments. In a typical sequential design method, first, an initial batch of samples is evaluated using a minimal experimental design. Then a model is built using this data and, based on the estimated accuracy of the model, the algorithm may decide that more samples are required. Exploitation-based methods estimate the error of the model over the design space and select new samples in locations where the estimated error is the largest. Exploration-based methods, on the other hand, try to improve the domain coverage by selecting samples in such a way that the design space is covered as uniformly as possible.

In the second section of this paper, a short overview is given of existing sequential design techniques. In the third section, we introduce a novel hybrid sequential design method which incorporates both an exploitation criterion based on local linear approximations and an exploration criterion using a Monte Carlo approximation of a Voronoi tessellation. This novel method is then applied in the fourth section to a use case from electronics and compared to other existing methods.

2 RELATED WORK

Over the last decade, a lot of research has been done on sequential design techniques in many different research fields, ranging from machine learning and artificial intelligence to engineering, with applications in e.g. electronics, mechanics and aerodynamics. In this section, we will give a short overview of existing sequential design techniques.

Please note that a lot of optimization algorithms use an iterative scheme similar to the one for sequential design, but with a completely different goal. These optimization algorithms may also employ sequential design techniques to minimize the number of samples required to find the global optimum. However, they are not concerned with finding a good global approximation on the entire design space. Because of this, a lot of optimization-oriented sequential design techniques focus more on exploitation and less on exploration, and might even ignore exploration completely. In this section, we will only consider related work on sequential design in the context of global surrogate modeling. For more information about sequential design in the context of optimization, please refer to (Knowles and Nakayama 2008; Forrester, Sobester, and Keane 2008).

2.1 Optimal Sequential Design

A large class of sequential design methods assume that the model type and its parameters are known in advance. This allows the algorithm to use the behavior of this model to guide the sampling process in the right direction. D-optimal designs minimize the determinant of the covariance matrix of the least squares estimates of the model parameters, while A-optimal designs minimize the trace (Chaloner and Verdinelli 1995; Santner, Williams, and Notz 2003). This minimization is achieved using sophisticated and computationally expensive optimization techniques.

Several sequential design methods use some aspects of optimal design to generate new samples. Busby, Farmer, and Iske (2007) split the design space into cells using a domain decomposition strategy, ensuring a certain degree of domain coverage. In each cell, their algorithm applies a local sequential optimal design. Gramacy and Lee (2006) use the treed Gaussian process model, which is an extension of the standard Gaussian process model, to approximate a black box system. After an initial batch of samples is selected using a Latin hypercube design, active learning methods (ALM and ALC) are used to select more samples in regions with high levels of uncertainty.

2.2 Generic Sequential Design

A sequential design method using optimal design methodology can be highly efficient if the model for which it was developed is suitable for the problem at hand. However, this may not always be the case, and it may not be known in advance which model type will perform the best. In fact, completely different model types may be used at the same time in a heterogeneous

modeling environment (Couckuyt et al. 2009). This motivates the need for a generic algorithm, which makes no presumptions about the model type, the behavior of the system or the amount of samples needed. Such an algorithm can only use output from the simulator and previously built (intermediate) models to decide where to sample next. These sampling algorithms will be examined and compared in this paper.

Generic sequential design methods have a major advantage over optimal sequential design strategies, especially in a black-box setting where little or nothing is known about the problem in advance. In this setting, choosing a model type for the problem comes down to guesswork, and if a bad choice is made, the optimal design that will be generated will not be optimal for another model type that will be tried later. A heteogeneous modeling environment can help solve this problem by automatically looking for models that match the problem at hand, while generating a sequential design that is not specifically tailored to one model type.

Exploration-based sequential design methods such as those described by Provost, Jensen, and Oates (1999) and Gehrke et al. (1999) try to give equal importance to each region of the design space, filling it up as evenly as possible. This is achieved by defining a density measure, which ranks regions in the domain according to their sampling density. The advantage of density-based sequential designs over one-shot experimental designs is that feedback from previous iterations of the algorithm can be used to avoid the evaluation of too many or too few samples. Exploration does not involve the responses of the system, because the goal is to fill up the input domain evenly.

Exploitation-based sequential design methods, on the other hand, use an error measure to guide sampling to areas which appear to be interesting. These may be highly non-linear areas, areas with discontinuous system behavior or areas containing (local) optima. Depending on the definition of the error measure, different areas will be focused on. The main problem with error-based methods is that they tend to focus too much on difficult locations, leaving large areas undersampled. Examples of error-based sampling methods are given by Geest et al. (1999), Thompson (2002) and Glassner (1995). Exploitation involves using the outputs of the previous function evaluations to guide the sampling process.

3 LOLA-VORONOI

The novel sampling technique introduced in this paper is a hybrid between exploitation-based and exploration-based sampling methods. It uses an exploitation criterion based on LOcal Linear Approximations of the system (LOLA) and an exploration criterion using a Voronoi tessellation of the design space. We will now briefly discuss both components separately. For more detailed information on the mathematical principles of LOLA-Voronoi and its implementation, please refer to (Crombecq 2008).

The LOLA component is motivated by the idea that the sampling rate should be proportional to the local linearity of the function. In regions where the system is almost linear, the output is easily predicted and less samples are needed than in regions of large variance. This linearity can be estimated by calculating a local linear approximation at each sample; if this local linear approximation is a good fit for nearby samples, the function must behave linearly in this region.

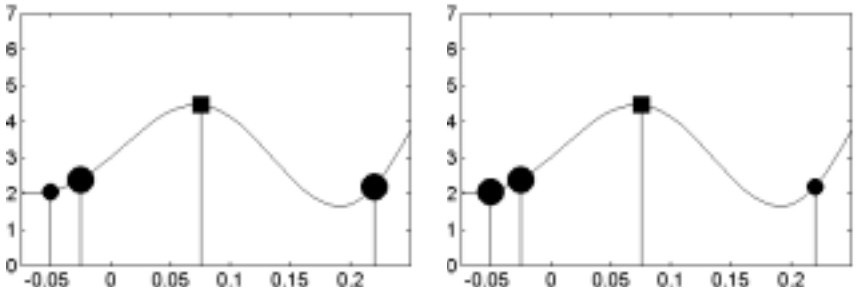
The best local linear approximation of a given function f is the gradient of this function, defined as:

$$\nabla f = \left(\frac{\partial f}{\partial x^1}, \frac{\partial f}{\partial x^2}, \ldots, \frac{\partial f}{\partial x^d} \right). \tag{1}$$

However, the derivative of the function that is to be modeled is rarely known in advance, so the gradient will have to be estimated. Because there is no prior knowledge on the locations of the available samples (the initial design might take any form), we cannot assume they are uniformly spread over the design space or form a certain pattern, which eliminates the use of traditional indirect gradient estimation methods such as finite differences (Fu 2005).

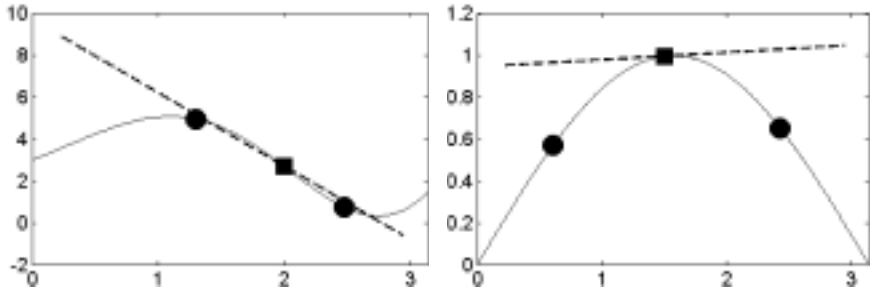
Therefore, estimating the gradient in each sample location comes down to choosing the set of neighboring samples that best represents the behavior of the function near that sample, so that this set of neighbouring samples can be used to estimate the gradient. This is illustrated in Figure 1. Least-squares regression is applied to the samples in the neighborhood set to get a gradient estimation. Finally, the difference between the true output value and the value of the gradient estimation at the neighboring points is used as a measure of linearity for the neighborhood. This is illustrated for the 1-dimensional case in Figure 2.

In addition to the exploitation criterion, an exploration criterion was implemented to overcome the inherent problems of purely exploitation-based methods (as described in Section 2.2). An approximation of the Voronoi tessellation of the entire design space is used to estimate the density of the samples in each region. When a sample has a large Voronoi cell size compared to all others, the region around this sample might be undersampled, and should be investigated more closely. The



(a) Good neighbourhood (b) Bad neighbourhood

Figure 1: Two different neighbourhoods of size 2 (2 samples) are visualized. The sample for which we need to find a neighbourhood is drawn as a square in the middle. The two samples which have been chosen as neighbours are drawn as large circles, while the third sample which is not in the neighbourhood set is drawn as a smaller circle. It is obvious that the neighbourhood chosen in 1(a) conveys much more information than the region shown in 1(b). Two samples which lie close to each other typically form a bad neighbourhood.



(a) Good fit (b) Poor fit

Figure 2: Two different situations are visualized. In both cases, the sample for which the gradient is estimated is drawn as a square, its respective neighbors as dots. The gradient estimations are drawn as dashed lines through the samples. As can be seen, the gradient is a decent fit in case 2(a); the line goes almost straight through the neighboring samples. This indicates that the function behaves linearly in this region, and that no additional sampling is necessary. In case 2(b), however, the gradient is a bad fit for the neighbouring samples. Hence, the function must be very non-linear in this area, and additional samples will be taken nearby.

Voronoi tessellation is used to guarantee a proper distribution over the entire design space, to ensure that no region remains permanently undersampled.

Once both the LOLA error and the relative Voronoi cell size have been calculated, all points are ranked according to a weighted measure of these two values. The highest scoring points will lie in a very non-linear and/or undersampled region. New samples are then selected in these regions. When these samples have been evaluated, the algorithm starts all over again.

4 EXPERIMENTAL SETUP

4.1 SUMO Research Platform

In order to compare the new method to other sampling strategies, LOLA-Voronoi was implemented in the SUrrogate MOdeling (*SUMO*) research platform (Gorissen et al. 2007 and Gorissen et al. 2009). This Matlab toolbox, designed for adaptive surrogate modeling and sampling, has excellent extensibility, making it possible for the user to add, customize and replace any component of the modeling process. It also has a wide variety of built-in test functions and test cases, as well as support for many different model types. Because of this, SUMO was the ideal choice for conducting this experiment.

The work-flow of SUMO is illustrated in Figure 3. First, an initial design (typically a sparse Latin hypercube) is generated and evaluated. Then a set of models is built, and the accuracy of these models is estimated using a set of measures (for example: cross-validation or an external validation test set). Each type of model has several parameters which can be modified, such as degrees of freedom for rational models, number and size of hidden layers in neural networks, smoothness for RBF models, and so on. These parameters are adjusted using a optimization method, and more models are generated until no further improvement can be made by changing the model parameters. If the desired accuracy has not yet been reached, a call is made to the adaptive sampling algorithm, which generates a set of new sample locations to be evaluated, and the algorithm starts all over again.

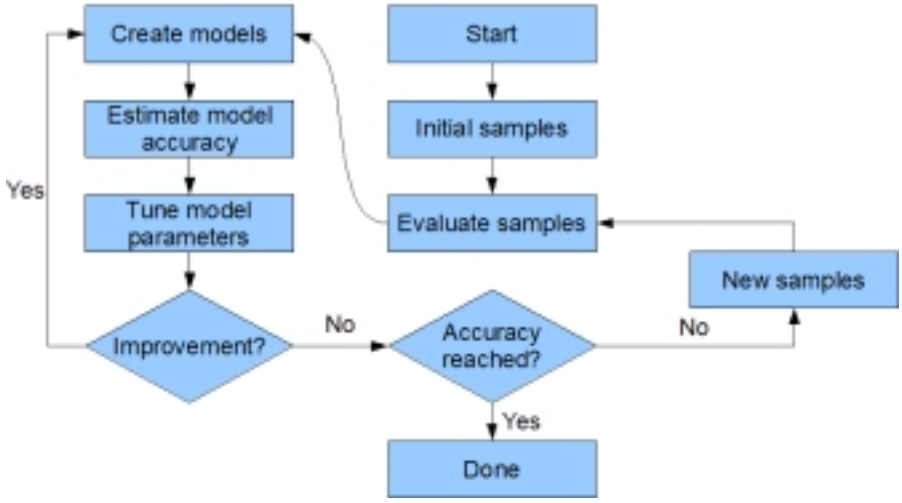


Figure 3: Flow-chart of the SUMO toolbox.

4.2 Test Case: an Electrical Low-Noise Amplifier

LOLA-Voronoi will be tested and compared against the other methods using a test-case from electronics: a narrowband Low Noise Amplifier (LNA), which is a simple RF circuit (Lee 2004). An LNA is the typical first stage of a receiver, having the main function of providing the gain needed to suppress the noise of subsequent stages, such as a mixer. In addition it has to give negligible distortion to the signal while adding as little noise as possible itself.

The performance figures of an LNA (gain, input impedance, noise figure and power consumption) can be determined by means of computer simulations where the underlying physical behavior is accurately taken into account. Each simulation typically requires a couple of minutes, too long for a circuit designer to work with efficiently. Instead a designer could use a very accurate surrogate model (based on circuit simulations) to quickly explore how the performance figures of the LNA scale with key circuit-design parameters, such as the dimensions of transistors, passive components, signal properties and bias conditions.

In this experiment, in order to keep the computation times manageable, the expensive simulations will be replaced by a first-order analytic model, which is a direct implementation of the formulas described below. Initial manual tests indicate that results obtained from this simplified model are also applicable to the physics-based simulator.

The schematic of the LNA is depicted in Figure 4. The functions that govern the approximate behavior of the LNA are shown below. The three input parameters are the inductances L_s, L_m, and the MOSFET width W. The outputs are the approximate input noise-current ($\sqrt{\overline{i_{in}^2}}$) and output noise-current ($\sqrt{\overline{i_{out}^2}}$). The input-noise current is defined by equations (2) to (6).

The remaining parameters have been set to fixed, typical values:

$$C'_{gs} = 1 \cdot 10^{-9}\mathrm{Fm}^{-1},$$

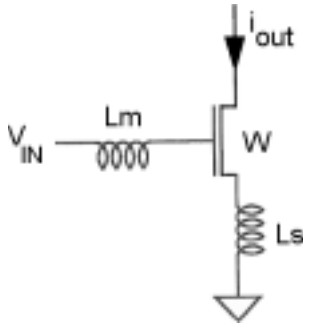


Figure 4: A Narrowband LNA.

$$f_{gs,in} = \frac{1 + j\omega L_s g'_m W}{1 - \omega^2 C'_{gs} W (L_s + L_m) + j\omega L_s g'_m W} \tag{2}$$

$$f_{ds,in} = \frac{\omega^2 C'_{gs} W (L_s + L_m)}{1 - \omega^2 C'_{gs} W (L_s + L_m) + j\omega L_s g'_m W} \tag{3}$$

$$\overline{i^2_{gs}} = W \cdot \overline{i^2_{gs}} \tag{4}$$

$$\overline{i^2_{ds}} = W \cdot \overline{i^2_{ds}} \tag{5}$$

$$\sqrt{\overline{i^2_{in}}} = \sqrt{|f_{gs,in}|^2 \cdot \overline{i^2_{gs}} + |f_{ds,in}|^2 \cdot \overline{i^2_{ds}} - 2 \cdot \mathrm{Im}(0.4 f_{gs,in} f^*_{ds,in}) \sqrt{\overline{i^2_{gs}} \cdot \overline{i^2_{ds}}}} \tag{6}$$

$$g'_m = 100\mathrm{AV}^{-1}\mathrm{m}^{-1},$$

$$\omega = 2\pi \cdot 5 \cdot 10^9 \mathrm{Hz},$$

$$\overline{i^2_{gs}}' = 2 \cdot 10^4 \mathrm{pA}^2\mathrm{Hz}^{-1}\mathrm{m}^{-1},$$

$$\overline{i^2_{ds}}' = 5 \cdot 10^6 \mathrm{pA}^2\mathrm{Hz}^{-1}\mathrm{m}^{-1}.$$

The meaning of the parameters is as follows: WC'_{gs} is the gate-source capacitance of the MOSFET, Wg'_m is the transconductance of the MOSFET, ω is the angular signal frequency, $\overline{i^2_{gs}}'$ is the gate-source noise current spectral density of the MOSFET and $\overline{i^2_{ds}}'$ is the drain source noise current spectral density of the MOSFET.

4.3 Choice of Models and Sampling Methods

LOLA-Voronoi will be compared against a pure exploitation method using the model error, an exploration method using a Voronoi tessellation and a random sampling scheme (as a base case), in an attempt to model the input-noise current ($\sqrt{\overline{i^2_{in}}}$). The input noise-current, which is depicted in Figure 5, was chosen for this experiment because it is the most challenging performance figure of the LNA (Gorissen et al. 2008). The goal is to compare the robustness of these sampling techniques in different modeling environments for a difficult problem.

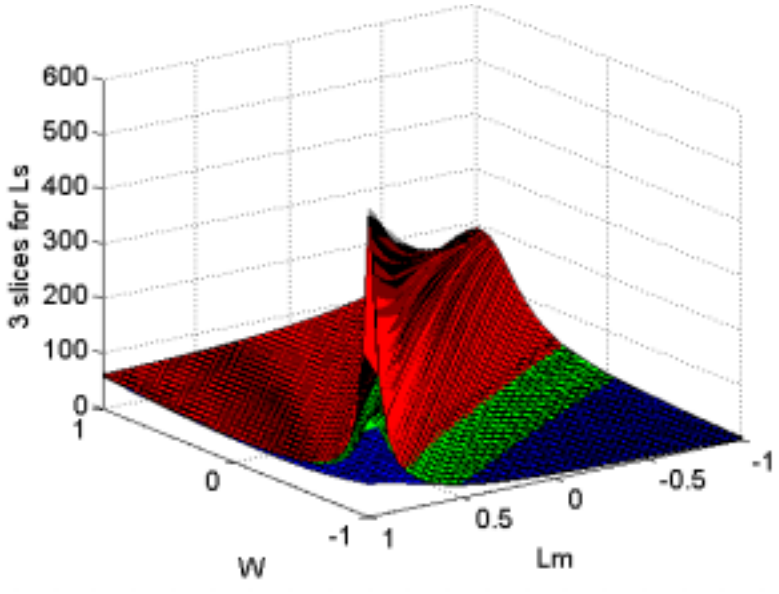


Figure 5: A plot of the input-noise current ($\sqrt{\overline{i_{in}^2}}$) of an LNA. The plot shows that the function is very flat, except for a tall ridge on the diagonal.

As can be seen on the figure, the surface behaves smoothly on most of the domain, except for a tall ridge on a diagonal. It it intuitively a good idea to select more samples in this highly dynamic region than in the relatively linear parts of the domain. Exploration-based methods will ignore the system behaviour, and will instead sample each region equally. Exploitation-based methods may sample only near the diagonal, completely neglecting the rest of the design space. LOLA-Voronoi will focus mainly in the diagonal, but will also select samples in the other regions of the design space, to guarantee a minimal coverage of the entire design space.

The exploitation-based method first constructs a very dense, randomly perturbated grid over the entire design space (typically 2500 points, even though the number may be higher for high-dimensional problems). It then evaluates and compares the best models from the previous iterations on this grid. This is done by subtracting the outputs from these models pair-wise from each other, and finding the locations on the grid where the difference is greatest, as described by Hendrickx and Dhaene (2005). Places where the models disagree indicate locations of high uncertainty, and will be sampled next. Because surrogate models can be evaluated fairly quickly, evaluating them over such a dense grid is usually not a problem. For very high-dimensional problems, however, the number of evaluations required to get a sufficiently dense grid may be too large, which makes this sampling strategy a poor choice for high-dimensional problems. This sampling strategy is very efficient at locating areas in the design space that are difficult to approximate, such as asymptotes or discontinuous regions, enabling the modeler to quickly increase accuracy in those regions. It, however, tends to undersample large regions of the design space and generate large clusters of samples, which might result in inaccurate global models. We will from now on refer to this strategy as Model Error sampling, because it estimates the approximation error by substracting the model outputs from each other.

The exploration-based method is essentially the Voronoi component of LOLA-Voronoi. It uses a Voronoi tessellation of the design space to find regions with large Voronoi cells. New samples are chosen in the largest Voronoi cells, as far from any other sample as possible. This method will distribute the samples evenly, which makes it very robust: if enough samples are taken, each portion of the design space will be sampled equally dense.

Finally, random sampling will be considered. A random sampling scheme just randomly selects samples in the design space, with no regards for previously evaluated samples. If enough samples are taken, random sampling will approximate a good space-filling design, while at the same time being the simplest and cheapest sampling method available. For a small sample set, however, large deviation from space-filling is to be expected, and the behaviour of this sampling scheme can be very erratic.

All of these methods will be compared against each other using 4 different model types: Kriging, artificial neural networks (ANN), radial basis functions (RBF) and (least squares) support vector machines (LS-SVM). For neural networks,

two implementations will be used: the default Matlab-implementation (M-ANN), which is rather slow, and the much faster Neural Network Based System Identification Toolbox (NNSYSID toolbox or N-ANN) developed by Nørgaard et al. (1996). The parameters of all these models are changed on the fly by SUMO to improve their accuracy.

This results in 20 different combinations of model types and sampling methods. To eliminate noise caused by random factors in the SUMO toolbox (such as the initial experimental design, the initial choice of model parameters and randomness in the parameter optimization method), each of these configurations will be run 4 times, resulting in a total of 80 runs of the SUMO toolbox.

The SUMO toolbox is configured so that each run starts with an initial design of 50 points in a Latin hypercube configuration. 25 new samples are taken after each modeling iteration, up to a total of 800 samples, after which the toolbox is halted. After each modeling iteration, the best model up to that point, along with its accuracy, is recorded on disk. In order to measure the accuracy of a model, a grid of 8000 samples (20^3) was evaluated in advance. This grid was used to calculate the root relative square error of the model on the grid. This means that models are evaluated by their true error, and not by an estimation of the error, to avoid bias in the results. The formula for the root relative square error is:

$$RRSE = \sqrt{\frac{\sum_{i=0}^{n} x_i - \bar{x}_i}{\sum_{i=0}^{n} x_i - \tilde{x}}},$$

where x_i is the true value at a sample location, $\bar{x}_i$ is the estimated value and $\tilde{x}$ is the average of the true values.

5 TEST RESULTS

A summary of the results of the test runs can be found in Table 1. A plot of the root relative square error of all the runs grouped by sampling method can be found in Figure 6. Each plot contains a line for each test run that was performed using that sampling method. The lines indicate that the accuracy increases (smaller error) as the number of samples increases; however, the rate of improvement differs between sampling methods, and also depends on the model that was used for that run.

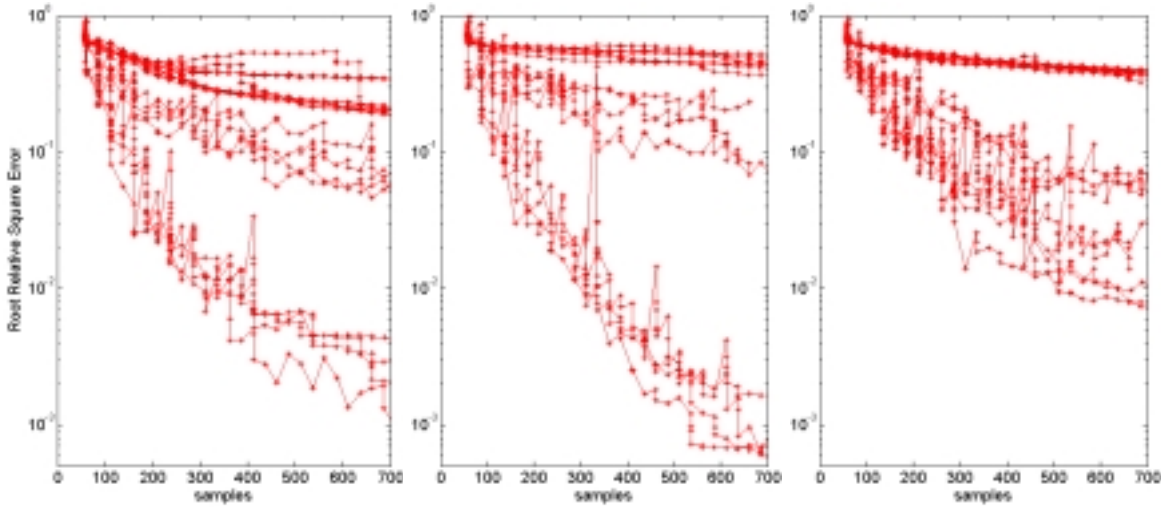
Table 1: Summary of test results in the 3-dimensional LNA-simulator. The best results for each model type are printed in bold. The worst results are printed in italic. M-ANN is the Matlab ANN toolbox, N-ANN is the NNSYSID toolbox. The average error is the average root relative square error of the 4 runs.

	Average Error				Standard Deviation			
	LOLA-Voronoi	Model Error	Voronoi	Random	LOLA-Voronoi	Model Error	Voronoi	Random
M-ANN	0.002	**0.0008**	*0.013*	0.007	0.001	0.0005	0.007	0.001
N-ANN	0.060	*0.141*	**0.049**	0.080	0.010	0.087	0.003	0.033
Kriging	**0.193**	*0.490*	0.376	0.360	0.007	0.000009	0.044	0.006
LS-SVM	**0.341**	0.412	0.388	*0.429*	0.005	0.032	0.016	0.015
RBF	**0.206**	*0.502*	0.377	0.386	0.015	0.037	0.010	0.006
Total	**0.160**	*0.309*	0.240	0.253	0.007	0.031	0.016	0.012

The overall best model was produced using the Model Error sampling scheme in combination with neural networks, resulting in a root relative square error of 4.343×10^{-4}. The neural networks toolbox of Matlab is the only model type that managed to achieve an accuracy greater than 10^{-2} or 1%. The other model types failed to achieve that level of accuracy in this difficult use-case.

The Matlab toolbox produces very smooth models, and hence does not suffer a lot from undersampling in very flat regions. An error-based measure is therefore free to focus completely on the most dynamic area, which is in this case the ridge along the diagonal axis, without giving up accuracy in the rest of the (relatively flat) design space. This results in very accurate models. LOLA-Voronoi also focuses on the difficult areas, but still samples the flat areas as well, ensuring a minimal coverage of the entire design space. This resulted in the only case in which LOLA-Voronoi is significantly outperformed by another sampling scheme.

In the case of the NNSYSID toolbox, the results were comparable in quality for all the sampling methods, except for the Model Error sampling scheme, which did noticeably worse. Some of the Model Error runs actually failed to produce any useful models, because the sampling algorithm initially completely missed the ridge and failed to locate it. This happened



(a) LOLA-Voronoi (b) Model Error (c) Voronoi

Figure 6: Plots of the root relative square error of all test runs as a function of time, grouped by sampling method. The Model Error method has the overall best model, but also the worst ones. The worst model produced by LOLA-Voronoi is still better than 11 Voronoi models and 8 Model Error models. This means that *half* of the total amount of Voronoi and Model Error-based runs perform worse than the worst LOLA-Voronoi run. Overall, models built using LOLA-Voronoi were considerably more accurate than models of the same type built with the other two sampling methods.

because the initial experimental design (a Latin hypercube) did not generate any samples on the ridge, causing the Model Error method to focus on (relatively) uninteresting regions, instead of exploring the design space to locate the ridge. This is a good illustration of the lack of robustness of the Model Error method.

With Kriging, LS-SVM and RBF, the best results were achieved using LOLA-Voronoi. The similarities in the results can be explained by the fact that all three methods internally use basis functions that depend on the euclidean distance of points from each other (LS-SVM uses an RBF kernel). Therefore, a good coverage of the design space is essential to success. This explains why a purely exploitation-based sampling scheme such as Model Error performs poorly with these model types, even compared to random sampling. Exploitation-based sampling tends to concentrate on specific areas which have been identified as interesting, leaving wide gaps in the design space, which makes it difficult to produce meaningful approximations in these undersampled areas. This observation is also confirmed by Gorissen et al. (2008) in a related study.

When the average is taken over all the model types, LOLA-Voronoi clearly performs the best across the board, producing models that are on average 33% better than the second best choice (Voronoi sampling). This can be seen on Figure 7, which groups the runs by sampling method and model type. Even though Model Error and Voronoi perform better with respectively M-ANN and N-ANN, LOLA-Voronoi follows close behind. Voronoi-based sampling is the second most robust method, performing decently in all test runs, but not excelling in any. Model Error sampling performs the worst overall, mainly due to its very poor performance with RBF and Kriging models.

6 CONCLUSIONS AND FUTURE WORK

The LOLA-Voronoi strategy was designed as a very robust, reliable and widely applicable sequential design method, able to produce good results with any model type, regardless of the problem that is to be modeled or the model that is used. To achieve this, the only information used to guide the sampling process consists of previously evaluated samples and their output values.

The test results confirm that LOLA-Voronoi is indeed a very robust, effective sequential design technique. It may not always guarantee the most accurate models, and in some cases it is slightly outperformed by other methods, but averaged over all model types it produced the best results. It is also very reliable, guaranteeing a proper coverage of the design space, while still focusing sufficiently on difficult areas.

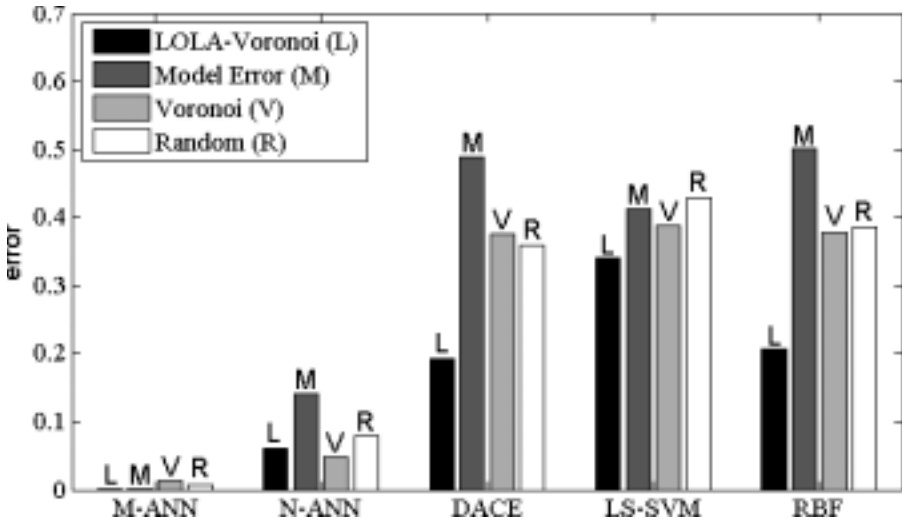


Figure 7: A bar chart of the root relative square error of all the runs performed in this experiment. The bars represent (from left to right) the runs performed using LOLA-Voronoi, Model Error, Voronoi and Random sampling schemes. It is clear from the chart that LOLA-Voronoi performs the best in the last three cases, and is only marginally outperformed by another sampling method in the first two. On average, it is by far the most reliable method.

LOLA-Voronoi is especially efficient for local approximation models such as RBF and Kriging, which assume a correlation between simulator output and geometrical distance in the design space. This property is magnified for problems in which the function is relatively flat in large parts of the design space and is highly dynamic in small regions. LOLA-Voronoi will quickly locate these unstable areas and focus on them, without neglecting the rest of the design space.

The Model Error method, while also very effective, is not as reliable as LOLA-Voronoi, as it may fail to locate important features of the system, resulting in useless models. Even though the overall best model is generated using the Model Error method, it is too unreliable and does not scale very well to high-dimensional problems.

Voronoi-based sampling is very reliable, but completely ignores system behaviour, and therefore tends to produce models that perform worse than LOLA-Voronoi. When using large amounts of samples, its performance is often only slightly better than random sampling. Its simplicity and reliability make it a viable choice if no better alternatives are available.

The robustness of LOLA-Voronoi comes at a cost, however. It is rather slow compared to other algorithms, mainly due to the expensive pre-processing required to estimate the gradient ($O(n^2)$ in the number of samples). However, this additional cost becomes negligible in a real-life environment in which sample evaluations may take hours or even days.

There are several ways of improving the efficiency of the algorithm, and these should be further explored and investigated. Heuristics can be employed to eliminate the expensive neighborhood search algorithm. Another improvement might be to replace the Voronoi tessellation by some other density measure. All these changes ultimately serve the goal of creating a fast, robust and flexible sampling technique.

ACKNOWLEDGMENTS

The authors would like to thank NXP Semiconducters and Jeroen Croon in particular for providing the LNA simulator code. This research was supported by FWO (Flemish Fund for Scientific Research) and by the European Commission through the Marie Curie Actions of its Sixth Program under contract number MTKI-CT-2006-041477.

REFERENCES

Balewski, L., and M. Mrozowski. 2004. Creating neural models using an adaptive algorithm for optimal size of neural network and training set. *15th International Conference on Microwaves, Radar and Wireless Communications* 2:543–546.

Barton, R. R. 1998. Simulation metamodels. In *Proceedings of the 30th Winter Simulation Conference*, ed. D. J. Medeiros and E. F. Watson, 167–174.

Beers, W. C. M. V. 2005. Kriging metamodeling in discrete-event simulation: an overview. In *Proceedings of the 37th Winter Simulation Conference*, ed. N. Steiger and M. E. Kuhl, 202–208.

Busby, D., C. L. Farmer, and A. Iske. 2007. Hierarchical nonlinear approximation for experimental design and statistical data fitting. *SIAM Journal on Scientific Computing* 29 (1): 49–69.

Chaloner, K., and I. Verdinelli. 1995. Bayesian experimental design: A review. *Statistical Science* 10:273–304.

Couckuyt, I., D. Gorissen, H. Rouhani, E. Laermans, and T. Dhaene. 2009. Evolutionary regression modeling with active learning: An application to rainfall runoff modeling. In *Proceedings of the International Conference on Adaptive and Natural Computing Algorithms*.

Crombecq, K. 2008. A gradient-based approach to adaptive surrogate modelling. Technical report, University of Antwerp.

Deschrijver, D., T. Dhaene, and J. Broeckhove. 2004. Adaptive model based parameter estimation, based on sparse data and frequency derivatives. *International Conference on Computational Science (LNCS 3037)*:443–450.

Eres, H., G. Pound, Z. Jiao, J. Wason, F. Xu, A. Keane, and S. Cox. 2003. Implementation of a grid-enabled problem solving environment in matlab. *International Conference on Computational Science (LNCS 2660)*:420–429.

Forrester, A., A. Sobester, and A. Keane. 2008. *Engineering design via surrogate modelling: A practical guide*. Wiley.

Fu, M. 2005. Stochastic gradient estimation, pre-print version of chapter 19. In *Handbook on Operations Research and Management Science: Simulation*, ed. S. G. Henderson and B. L. Nelson. Elsevier.

Geest, J. D., T. Dhaene, N. Fach, and D. D. Zutter. 1999. Adaptive cad-model building algorithm for general planar microwave structures. *IEEE Transactions on Microwave Theory and Techniques - Special Issue on Multilayer Microwave Circuits* MTT-47 (9): 1801–1809.

Gehrke, J., V. Ganti, R. Ramakrishnan, and W. Y. Loh. 1999. Boat - optimistic decision tree construction. *SIGMOD Record* 28 (2): 169–180.

Glassner, A. 1995. *Principles of digital image synthesis*. Morgan Kaufmann.

Gorissen, D., K. Crombecq, W. Hendrickx, and T. Dhaene. 2007. Adaptive distributed metamodeling. *High Performance Computing for Computational Science - VECPAR 2006* 4395:579–588.

Gorissen, D., L. D. Tommasi, K. Crombecq, and T. Dhaene. 2009. Sequential modeling of a low noise amplifier with neural networks and active learning. *Neural Computation & Applications* 18 (5): 485–494.

Gorissen, D., L. D. Tommasi, W. Hendrickx, J. Croon, and T. Dhaene. 2008. Rf circuit block modeling via kriging surrogates. *17th International Conference on Microwaves, Radar and Wireless Communications*.

Gramacy, R., and H. K. H. Lee. 2006. Adaptive design of supercomputer experiments. Technical report, Dept of Applied Math & Statistics, University of California.

Hendrickx, W., and T. Dhaene. 2005. Sequential design and rational metamodelling. In *Proceedings of the 37th Winter Simulation Conference*, ed. N. Steiger and M. E. Kuhl, 290–298.

Knowles, J., and H. Nakayama. 2008. Meta-modeling in multiobjective optimization. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, 245–284. Springer-Verlag.

Lamecki, A., P. Kozakowski, and M. Mrozowski. 2004. Cad-model construction based on adaptive radial basis functions interpolation technique. *15th International Conference on Microwaves, Radar and Wireless Communications* 2:799–802.

Lee, T. H. 2004. *The design of cmos radio-frequency integrated circuits 2nd ed.* Cambridge University Press.

Lehmensiek, R., P. Meyer, and M. Müller. 2002. Adaptive sampling applied to multivariate, multiple output rational interpolation models with application to microwave circuits. *International Journal of RF and Microwave Computer-Aided Engineering* 12 (4): 332–340.

Nørgaard, M., O. Ravn, L. Hansen, and N. Poulsen. 1996. The nnsysid toolbox - a matlab toolbox for system identification with neural networks. *Computer-Aided Control System Design*:374–379.

Panayiotou, C., C. Cassandras, and W.-B. Gong. 2000. Model abstraction for discrete event systems using neural networks and sensitivity information. 335–341.

Provost, F. J., D. Jensen, and T. Oates. 1999. Efficient progressive sampling. *Knowledge Discovery and Data Mining*:23–32.

Santner, T. J., B. J. Williams, and W. I. Notz. 2003. *The design and analysis of computer experiments*. Springer.

Simpson, T. W., D. K. J. Lin, and W. Chen. 2001. Sampling strategies for computer experiments: Design and analysis. *International Journal of Reliability and Applications* 2 (3): 209–240.

Sugiyama, M. 2006. Active learning in approximately linear regression based on conditional expectation of generalization error. *Journal of Machine Learning Research* 7:141–166.

Thompson, S. K. 2002. *Sampling*. Wiley.

Zhao, H., and D. Knight. 2004. Data driven design optimization methodology development and application. *International Conference on Computational Science (LNCS 3038)*:1611–3349.

AUTHOR BIOGRAPHIES

KAREL CROMBECQ received his master's degree in Computer Science from the University of Antwerp, Belgium in 2006. Currently he is a PhD student with the CoMP research group at the department of Computer Science and Mathematics of the University of Antwerp, Belgium. He is funded by an FWO fellowship (Flemish Fund for Scientific Research). His research interests lie in distributed surrogate modeling, adaptive sampling techniques and machine learning. His web address is <comp.ua.ac.be> and his email address for these proceedings is <Karel.Crombecq@ua.ac.be>.

DIRK GORISSEN received his master's degree in Computer Science from Antwerp University, Belgium in 2004. In addition he obtained a Master in Artificial Intelligence from the Katholieke Universiteit Leuven, Belgium in 2007 while a PhD student at the Computational Modeling and Simulation (COMS) research group at the University of Antwerp. Currently he is a PhD student at the INTEC Broadband Communication Networks (IBCN) research group at Ghent University, Belgium. The IBCN-group is also part of IBBT (Interdisciplinary institute for BroadBand Technology). His current research interests include: adapative global surrogate modeling and its application to real world problems, distributed computing and Artificial Intelligence. His web address is <sumo.intec.ugent.be> and his email address for these proceedings is <Dirk.Gorissen@ugent.be>.

LUCIANO DE TOMMASI received the Laurea degree (summa cum laude) in electronic engineering and the PhD degree in electrical engineering from the Universita di Napoli Federico II, Italy in 2003 and 2006, respectively. During 2005-2006, he was a Visiting Researcher at SINTEF Energy Research, Trondheim, Norway. Currently he is a Marie Curie Fellow with the University of Antwerp, Belgium, Department of Mathematics and Computer Science, working at NXP Semiconductors (Eindhoven) on behavioral modeling of RF circuit blocks. His research interests concern the reduced order and behavioral modeling of linear and nonlinear circuits and electrical systems. Dr. De Tommasi is a reviewer for the IEEE Transactions on Advanced Packaging. His web address is <comp.ua.ac.be> and his email address for these proceedings is <Luciano.DeTommasi@ua.ac.be>.

TOM DHAENE is a full professor in the INTEC Broadband Communication Networks research group at Ghent University, Belgium and associated member of the Electromagnetics Group. He received the Electrical Engineering and PhD degrees from Ghent University, in 1989 and 1993, respectively. From 1993 until 2000, he worked in the EDA industry (Alphabit, Hewlett-Packard, Agilent Technologies), where he was one of the key developers of ADS Momentum. From 2000 until 2007, he was a professor at the University of Antwerp. He is author of more than 125 journal and conference publications, and is the holder of 3 U.S. patents. His EDA software is successfully used by academic, government and business organizations worldwide. He is a Senior Member of the IEEE. His research interests include distributed scientific computing, signal integrity, model order reduction, surrogate modeling and macromodeling. His web address is <sumo.intec.ugent.be> and his email address for these proceedings is <Tom.Dhaene@ugent.be>.