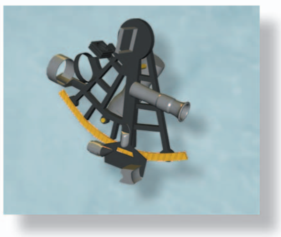


Application Notes

A Software Framework for Automated Behavioral Modeling of Electronic Devices

■ Dirk Gorissen, Dirk Deschrijver, Tom Dhaene, and Daniel De Zutter

Behavioral models such as macromodels, surrogate models, metamodels, and response surface models have many applications in diverse research domains such as aerodynamics [1], hydrology [2], mechanical engineering [3], and many more. When considering the design flow of electronic devices, these models are often used to characterize the time- or frequency-dependent behavior of an electronic component while taking all electromagnetic (EM) phenomena into account: crosstalk, attenuation, dispersion, and coupling effects for example [4]. Such models are of crucial importance for efficient design space exploration, design optimization, and sensitivity analysis [5], [6]. A key advantage is that they are calculated independently of the device's physics and that they are valid for over a wide range of design variables, taking into account multiple geometrical layout or substrate features. Additionally, the models can easily be linked together in a model cascade.

Some examples of behavioral models include polynomial/rational functions [7]–[10], Kriging models [11], [12], artificial neural networks (ANNs) [13], [14], and support vector machines (SVMs) [15]. To obtain a reliable model that satisfies all design requirements, significant challenges need to be addressed, such as which data collection strategy to use, which model type is most applicable, how to rank different models according to quality, etc. At the same time, electronic design automation (EDA) experts are typically not familiar with the intricacies of these design choices. Their primary concern is obtaining an accurate replacement model with minimal computational overhead. The selection of model types, model parameter optimization, and sampling strategy are of lesser or no interest to them since these are just necessary intermediate steps to solve the overall design problem [16].

In this article, a unified and automated modeling framework is presented that can assist an EDA domain expert in generating accurate behavioral models [17]. It drives the underlying system-level simulator and at the same time builds and tunes the model in such a way that the model accuracy and compactness are maximized. On the one hand, it does not require particular assumptions about the device under test, but on the other hand, as no algorithm is optimal for every problem, full control is still left with the EDA domain expert such that problem-specific assumptions or customizations can easily be applied. Therefore, this work can lower the barrier of entry for domain experts, promote benchmarking between existing methods, and facilitate the transfer of knowledge from behavioral modeling researchers to EDA domain experts.

Dirk Gorissen (dirk.deschrijver@gmail.com), Tom Dhaene, and Daniel De Zutter are with the Department of Information Technology (INTEC) at Ghent University—IBBT, Gaston Crommenlaan 8, 9050 Ghent, Belgium. Dirk Deschrijver is a post-doctoral researcher of The Research Foundation Flanders (FWO-Vlaanderen).

Digital Object Identifier 10.1109/MMM.2012.2205836
Date of publication: 13 September 2012

Global Behavioral Modeling

From an abstract level, the system-level computer simulator can be seen as an unknown multivariate function $f : \Omega \mapsto \mathbb{C}^q$ that is defined on some domain $\Omega \subset \mathbb{R}^d$ and whose function values $Y = \{f(\mathbf{x}_1)\ldots, f(\mathbf{x}_k)\} \subset \mathbb{C}^q$

are known at a fixed set of pairwise distinct sample points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subset \Omega$ [18]. The behavioral model is then a suitable function $\tilde{f}$ that closely resembles f as measured by some criterion ξ, where ξ is defined as a triplet that consists of following parts:

$$\xi = (\Lambda, \varepsilon, \tau). \tag{1}$$

Λ is a model quality estimator, i.e., a function that assigns a positive score to a behavioral model where lower scores indicate a more desirable model. Many implementations of Λ have been described: the error in the samples, the hold-out, bootstrap, cross validation, jack-knife, Akaike's information criterion (AIC), etc. [19]. An implementation of Λ is typically associated with an error function ε. While Λ specifies the model quality estimation algorithm such as cross validation, ε specifies what error function should be used to calculate the actual quality score, e.g., mean relative error or maximum absolute error. Finally, τ is the model quality target desired by the user.

The behavioral modeling problem (i.e., finding the best approximation $\tilde{f}^*$) for a given set of data points $D = \{(\mathbf{x}_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_k, f(\mathbf{x}_k))\}$ can be formally defined as

$$\tilde{f}^* = \arg\min_{t \in T} \arg\min_{\theta \in \Theta} \Lambda(\varepsilon, \tilde{f}_{t,\theta}, D) \tag{2}$$

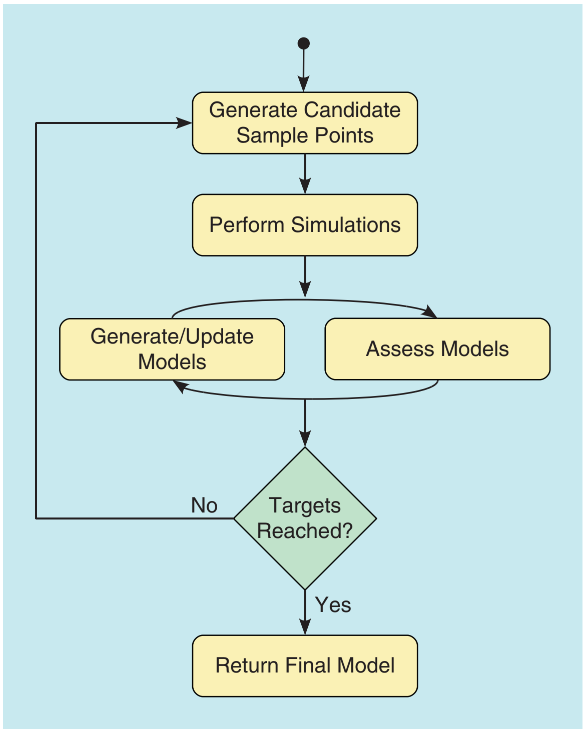


Figure 1. *General flowchart for adaptive global behavioral modeling. The outer sampling loop will iteratively sample the design space while an inner modeling loop will generate models that accurately capture the data selected each iteration.*

such that

$$\Lambda(\varepsilon, \tilde{f}^*_{t,\theta}, D) \leq \tau, \tag{3}$$

where $\tilde{f}_{t,\theta}$ is the parametrization θ (from a parameter space Θ) of $\tilde{f}$, and $\tilde{f}_{t,\theta}$ is of model type t (from a set of model types T).

The first minimization over $t \in T$ is the task of selecting a suitable model type, i.e., a rational function, a neural network, a spline, etc. This is the model type selection problem. In practice, one typically considers only a single $t \in T$, though others may be included for comparison. In some cases, the choice of model type is also linked to physical properties of a system such as causality, stability, or passivity [20]–[22]. Then, given a particular model type t, the task is to find the model parameter assignment θ that minimizes the model quality measure Λ, e.g., determine the optimal order of a polynomial model or the optimal neural network topology. This is referred to as the "hyperparameter optimization problem," though generally both minimizations (over t and over θ) are simply referred to as the "model selection problem."

In order to construct $\tilde{f}$, the data set needs to be populated. Traditionally, the size and distribution of the data is chosen up-front using standard experimental designs such as a latin hypercube design. However, since $f(\cdot)$ is expensive to compute, it becomes important to avoid unnecessary simulations. Consequently, since the complexity of the response surface is not known up front, defining an a priori data distribution is undesired. Instead, data points should be selected iteratively, at locations where the benefit to the model will be the greatest.

One starts by constructing an initial experimental design using one of the many algorithms available from the theory of Design and Analysis of Computer Experiments (DACE) [23]. The task is then to generate a new set of maximally informative samples based on one or more criteria. Examples of such criteria include distance from other points, distance from optima, prediction uncertainty, etc. In each iteration of a sampling algorithm, a best approximation model is obtained, leading to a sequence of models. The overall number of data samples should be kept to a minimum, while at the same time maximizing the benefit to the model $\tilde{f}$ with respect to ξ.

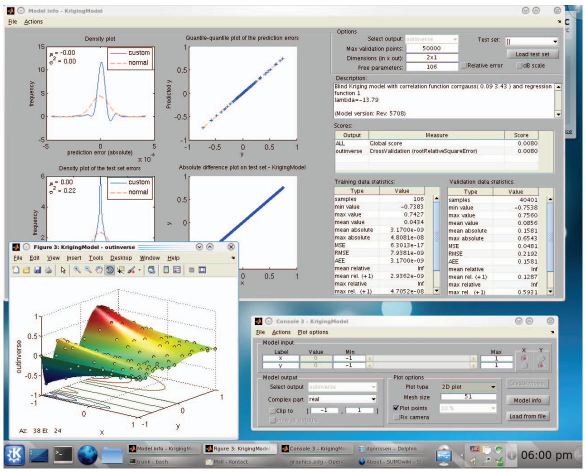
This process is called "adaptive sampling" [24], but is also known as "active learning" [25], "reflective exploration" [7], "Optimal Experimental Design" [26], "Sequential Exploratory Experimental Design" [27], and "sequential design" [28].

Figure 1 shows how the model generation and data collection fit together in a high-level control flow.

There are two main parts to the flowchart: an outer adaptive sampling loop and an inner adaptive modeling loop. Given a set of sample points, the inner loop will optimize the model hyperparameters (e.g., neural network topology) until no further improvement is possible on the given set of samples. Having reached a minimum of the hyperparameter space [the inner minimization in (2)], the algorithm then signals the sample selection loop to determine a new, maximally informative set of simulation points. These are then evaluated and merged intelligently with the existing sample points and the adaptive modeling process is allowed to resume. This whole process continues until the user-defined accuracy has been reached or some user-defined stopping criterion is met. The final behavioral model is returned to the user.

The flow of control in Figure 1 forms the basis for every data-based approximation methodology. For example, note the similarities with the work by Zhang in [29]. Of course, as is, Figure 1 has little practical value since it is very vague. Like the expectation-maximization algorithm [30] it is really a meta-algorithm that can have many different actual instantiations, depending on what techniques are used for each step. However, it should be clear that there are a large number of modeling options and choices available to the designer: different model types, different experimental designs, different sample selection strategies, different model selection criteria, different hyperparameter optimization strategies, etc. All choices that are difficult to make in an a priori manner [31].

Consequently, this generic control flow is taken as a starting backbone, extended with pluggable components for each of the different steps. This results in a generic solution that can still be customized for a specific problem if needed. In addition, custom extensions ensure that the whole is more than a bag of tools but that some automation (e.g., for model type selection) is available if needed. The resulting modeling platform can easily help an EDA domain expert to choose between different techniques and help him apply advanced modeling methods to his problem with minimal overhead. Depending on the configuration, the modeling platform can run autonomously, for example if just a quick-and-dirty model is needed, or under full manual control with problem specific methods and customizations (to ensure optimal efficiency accuracy for a given problem). The framework in question is the MATLAB Surrogate Modeling (SUMO) Toolbox.



Model info - KrigingModel
File Actions
Density plot
custom
normal
prediction error (absolute)
Quantile-quantile plot of the prediction errors
Density plot of the test set errors
Absolute difference plot on test set - KrigingModel
Figure 3: KrigingModel - outinverse
File Edit View Insert Tools Desktop Window Help
outinverse
Az: 38 El: 24
Console 3 - KrigingModel
File Actions Plot options
Create movie
Model info
Load from file
06:00 pm

Options

Select output: outinverse
Max validation points: 50000
Dimensions (in x out): 2x1
Free parameters: 106
Test set: []
Relative error
dB scale

Description:
Blind Kriging model with correlation function corrgauss(0.09 3.43) and regression function 1
lambda=−13.79

(Model version: Rev: 5708)

Scores:

Output	Measure	Score
ALL	Global score	0.0080
outinverse	CrossValidation (rootRelativeSquareError)	0.0080

Training data statistics:

Type	Value
samples	106
min value	−0.7383
max value	0.7427
mean value	0.0434
mean absolute	3.1700e-09
max absolute	4.8081e-08
MSE	6.3013e-17
RMSE	7.9381e-09
AEE	3.1700e-09
mean relative	Inf
mean rel. (+1)	2.9362e-09
max relative	Inf
max rel. (+1)	4.7052e-08

Validation data statistics:

Type	Value
samples	40401
min value	−0.7538
max value	0.7560
mean value	0.0856
mean absolute	0.1581
max absolute	0.6543
MSE	0.0481
RMSE	0.2192
AEE	0.1581
mean relative	Inf
mean rel. (+1)	0.1287
max relative	Inf
max rel. (+1)	0.5931

Model input

Label	Value	Min	Max
x	0	-1	1
y	0	-1	1

Model output
Select output: outinverse
Complex part: real
Clip to [-1 , 1]
Show all outputs

Plot options
Plot type: 2D plot
Mesh size: 51
[✓] Plot points 10 %
Fix camera

Figure 2. *Screenshot of SUMO Toolbox. (depicted with permission from http://www.sumo.intec.ugent.be).*

SUMO Toolbox

The SUMO Toolbox illustrated in Figure 2 [32], [33] is a flexible tool that integrates different modeling approaches and implements an adaptive behavioral model construction algorithm based on the flowchart in Figure 1. Given an EDA simulation engine [e.g., Simulation Program with Integrated Circuit Emphasis (SPICE), SpectreRF, High Frequency Structure Simulator (HFS), Momentum, Computer Simulation Technology (CST) etc.) the toolbox computes a model within the time and accuracy constraints set by the user. Different plug-ins are supported:

- **Model Types**
  - rational functions
  - Kriging
  - splines
  - SVM
- **Model Parameter Optimization Algorithms**
  - particle swarm optimization
  - efficient global optimization
  - simulated annealing
- **Sample Selection**
  - random
  - error based
  - density based
- **Sample Evaluation Methods**
  - local
  - on a cluster or grid.

The behavior of each software component is configurable through a central XML [34] configuration file, and components can easily be added, removed, or replaced by custom implementations. This is illustrated in Figure 3. In addition, the toolbox provides meta-plug-ins, which include, e.g., evolutionary algorithms to automatically select the best model type for a given problem (see [31] for details) or the possibility to use multiple model selection criteria in concert [35].

There is built-in support for high-performance computing. On the modeling side, the model generation process can take full advantage of multicore CPUs if the MATLAB Parallel Computing Toolbox is available, and even of a complete cluster or grid if the MATLAB Distributed Scheduler is available. This can result in a significant increase in speed for model types where the fitting process can be expensive such as in neural networks.

Likewise, sample evaluation (simulation) can occur locally with the option to take advantage of multicore architectures or on a separate computer cluster or

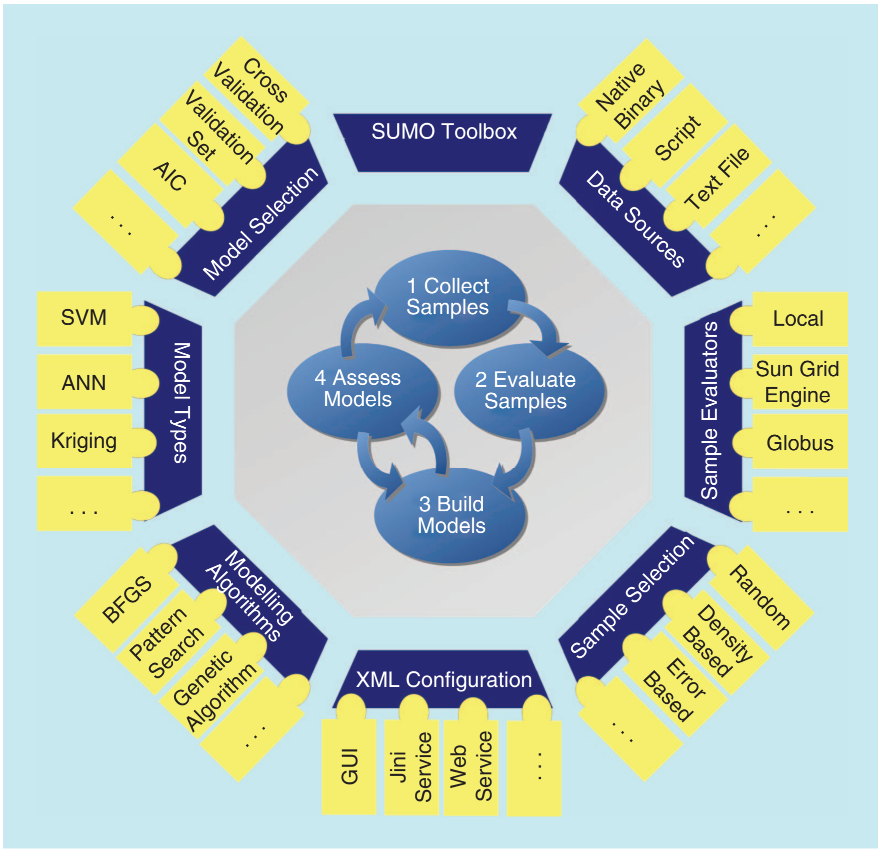


Figure 3. *Illustration of how the SUMO Toolbox enables algorithms and methods to be combined in different ways through the use of a plug-in-based infrastructure (depicted with permission from [32]).*

grid, possibly accessed through a remote head node. All interfacing with the grid middleware (submission, job monitoring, rescheduling of failed/lost simulation points, etc.) is handled transparently and automatically (see [36] for more details). Also, the sample evaluation component runs in parallel with the other components (nonblocking) and not sequentially. This allows for an optimal use of computational resources.

The SUMO Toolbox has already been applied successfully to a very wide range of applications, including RF circuit block modeling [32], hydrological modeling [37], electronic packaging [38], aerodynamic modeling [39], automotive data modeling [35], and EM compatibility [50]. Besides global modeling capabilities, the SUMO Toolbox also includes a powerful optimization framework based on the Efficient Global Optimization framework developed by Jones [40].

Applications

To demonstrate the performance of the framework described above, two EM device modeling problems are discussed: a passive component (bandstop filter) and an active device [low-noise amplifier (LNA)].

EM Behavioral Modeling of a Bandstop Filter

The first application concerns the modeling of a parametrized double-folded microstrip stub bandstop filter [6]. The filter with ports P_1 and P_2 is shown in Figure 4.

The substrate is 0.1270 mm thick with a relative dielectric constant $\epsilon_r = 9.9$ and a loss tangent $\tan\delta = 0.003$. The lines are infinitely thin and perfectly conducting with $W = 0.1219$ mm. The parametric macromodel of the scattering matrix is built as a function of the varying length of each folded segment $L \in [1.97, 2.41]$ mm and varying spacing between a folded stub and the main line $S \in [0.06, 0.24]$ mm over the frequency range [5, 20] GHz. The EM simulation engine used is ADS Momentum.

To assess the accuracy of the models objectively, a dense $50 \times 30 \times 151$ ($L \times S \times$ frequency) reference grid was calculated. It is important to note that this data set is not used during the modeling process in any way since, typically, such a reference grid is not available. It is simply used to objectively test the quality of the models a posteriori.

Modeling Settings

The SUMO Toolbox v6.2.1 is configured with the generic ANN and multivariate rational modeling plug-ins. No problem-specific tuning or settings were used. Thus the results will be indicative of how well a behavioral model can be obtained if the framework is applied without further problem-specific knowledge, using only concepts from a data-modeling perspective. The multivariate rational models are based on a custom implementation [41]. The order selection is performed using a genetic algorithm (population size: 30, number of generations: 20), thus this need not be done manually. The model quality estimator, $\Lambda(\cdot)$, is fivefold cross validation [42], [19] configured with a mean square error (MSE) function (ε). The ANN models

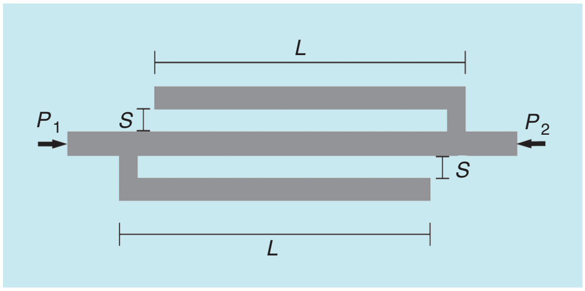


Figure 4. *Double-folded microstrip stub bandstop filter.*

are based on the MATLAB Neural Network Toolbox and are trained with Levenberg-Marquardt back-propagation with Bayesian regularization [43], [44] (600 epochs). Since the ANN models do not support complex data directly, the real and imaginary components are fitted separately using an ANN model with two outputs. The topology and initial weights are determined by an evolutionary strategy-like algorithm, with 25 models being generated each modeling iteration. To assess the model quality and drive, the topology selection, $\Lambda(\cdot)$is taken as the sum of two criteria: the in-sample error (using an MSE) and the linear reference model (LRM) score [45]. The combined scores for each output (real/imaginary) are then added together to obtain the overall score of the model. The LRM score penalizes a model if it exhibits unwanted bumps or ripples between the sample points. It can be seen as a kind of smoothness penalty that has the added benefit of keeping the neural network model complexity low. We found that the advantage of using these two metrics together is that they produce better ANN models and are much faster to evaluate than cross validation.

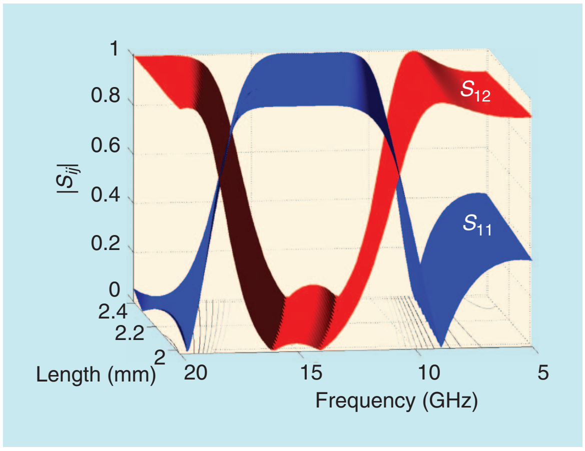


Figure 5. *Plot of the final rational model for $|S_{11}|$ and $|S_{12}|$ at $S = 0.131$.*

Sampling Settings

The modeling starts with an optimal Latin hypercube design of four points augmented with the corner points in the two-dimensional (2-D) $L \times S$ space. Each iteration a new sample is selected using the local linear approximation (LOLA)-Voronoi adaptive sampling algorithm [46]. LOLA-Voronoi identifies new sample locations by performing a trade-off between exploration, which is covering the design space evenly, and exploitation, concentrating on regions where the actual response is nonlinear. LOLA-Voronoi's strengths are that it scales well with the number of dimensions, makes no assumptions about the underlying problem or behavioral model type, and works in both the $\mathbb{R}$ and $\mathbb{C}$ domains. LOLA can automatically identify nonlinear regions in the domain and sample these more densely than the more linear, flatter regions. To do this, LOLA-Voronoi depends only on previously evaluated data points.

Because frequency is sampled automatically by ADS Momentum, LOLA-Voronoi samples in the 2-D instance space defined by the geometric parameters L and S. New samples are submitted to ADS Momentum, which returns a set of S-parameters over the frequency range of interest. In order to select a sample in the reduced 2-D design space (without the frequency parameter), slices are taken at multiple frequencies, and LOLA-Voronoi is used on each slice separately. This results in a nonlinearity estimation for each frequency slice, covering the entire 2-D design space. These estimations are aggregated into one score, which is used to select new samples in locations with the highest nonlinearity over the entire frequency range.

Momentum is configured to return 31 frequency samples, and the SUMO Toolbox is set to terminate after 136 instance simulations ($=136 \times 31 = 4{,}216$ data points). Therefore, instead of using an explicit target accuracy value, simulations are performed until the computational budget is exhausted ($\tau = -\infty$).

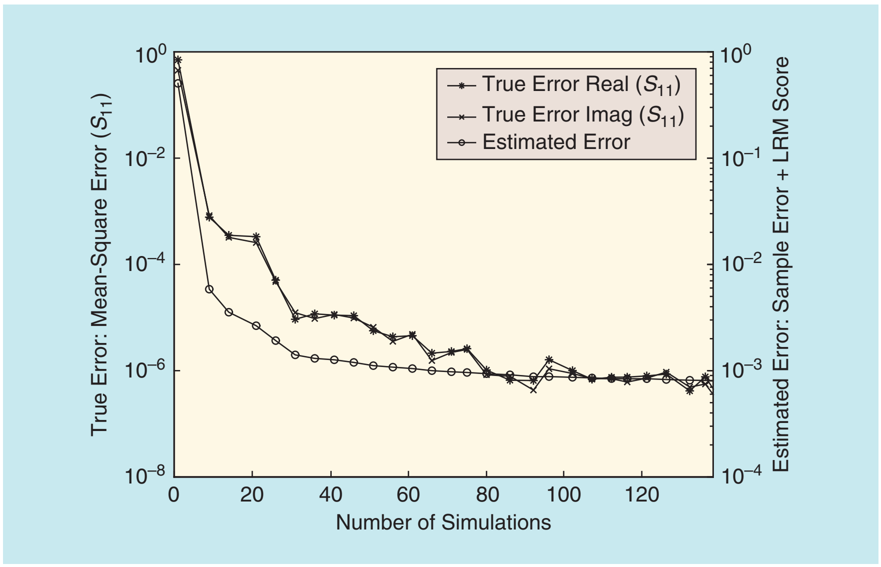


Figure 6. *Evolution of the true and estimated error over the reference data during the ANN topology optimization. Note that the true error is not minimized directly, it is only shown here for reference.*

Results

Figure 5 shows a plot of the final rational models for S_{11} and S_{12}. For conciseness, the following discussion will only treat S_{11}. The results for S_{12} are completely analogous.

Figure 6 shows that the ANN model generation code is able to reduce the true error evaluated over a dense

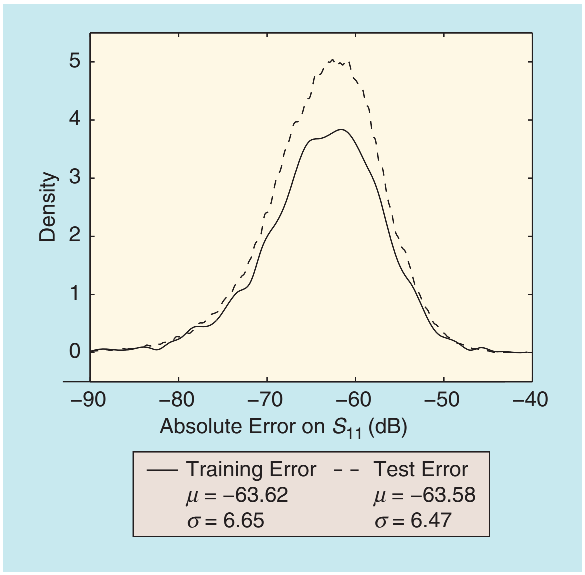


Figure 7. *Probability density function of the absolute errors of the final ANN model over the training and reference data after 136 simulations.*

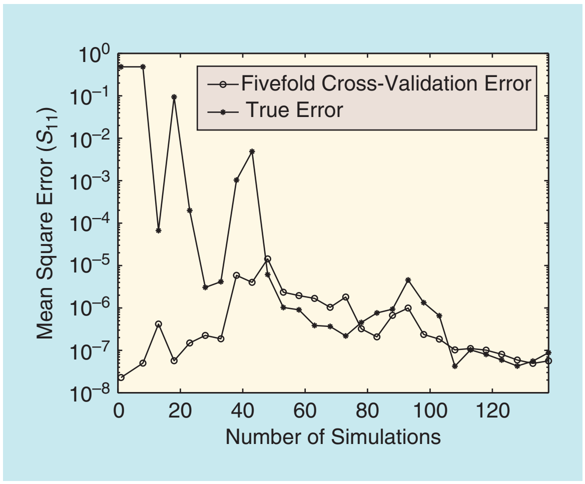


Figure 8. *Evolution of the true error and estimated error over the reference data during the rational model order selection.*

reference grid quite effectively while minimizing the estimated error. This means that the in-sample error/LRM combination is a good approximator of the true accuracy. In line with previous results [45], we see that the decrease in true error is quite steady with no large jumps.

Figure 7 shows the probability density function of the absolute errors (obtained using kernel density estimation [47]) for the final best ANN model found by the toolbox after 136 momentum simulations. The figure shows that very good accuracy is achieved. Note also the small difference between the training and test curves, meaning there is no overfitting and the models show good generalization. The final ANN model for S_{11} is a $3-8-16-2$ network (210 parameters) and for S_{12} a $3-12-15-2$ network (275 parameters). This notation denotes the number of elements of the input vector and the number of neurons in each layer of the ANN.

Let us now examine the results for the multivariate rational functions. The evolution of the true error and estimated error during the model generation process is shown in Figure 8. Compared to the ANN results in Figure 6, we see that the error reduction is more erratic in the rational case, particularly in the beginning when only little data is available. This is due to the use of cross validation as the accuracy estimator. Even though we ensure an even distribution of the different folds, when data is relatively sparse, cross validation is known to give biased results [19] and can mislead the order selection procedure.

Again, good accuracy is achieved on S_{11}, with the mean accuracy being better than for the ANN models. Given the rational nature of the underlying transfer function, this should not be surprising. However, it should be noted that the modeling effort was not the same for both model types. Since the rational functions are fast to construct and train, we can afford to build more of them during each modeling iteration than neural network models (since these are much slower to train). In this case, $20 \times 30 = 600$ rational models are built each modeling iteration versus only 25 neural networks.

EM Behavioral Modeling of a Low-Noise Amplifier

The second application is concerned with the accurate capturing of the nonlinear behavior of an LNA. An LNA is characterized by performance figures such as voltage gain, linearity, and noise figure, which are functions of design parameters such as width and length of transistors, bias conditions, and values of passive components [48]. The goal of the design process is to figure out one or more sets of design parameters resulting in a circuit, which fulfills the specifications, that is, constraints given on the performance. More information on the modeling of this problem can be found in [32].

Obtaining the required circuit design parameters can be done through an approximation of the circuit performance figures based on one or more behavioral models. This is referred to as a "forward model" of the circuit. A forward model can be either obtained via direct modeling of circuit performance using the one-step approach or by using intermediate surrogate models of a convenient set of behavioral parameters (e.g., admittances and noise functions) and by computing performances via analytical equations in a post-processing or two-step approach. This is illustrated in Figure 9.

To illustrate the indirect modeling approach, we consider the LNA and attempt to reproduce the behavior of the input-noise current response variable $\sqrt{i_{in}^2}$. The input parameters are the MOSFET width W, the inductances L_s, L_m, and the frequency, leading to a

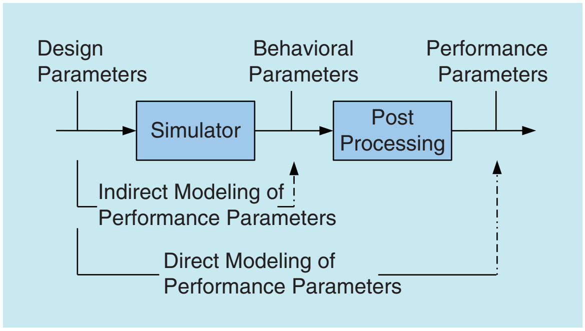

Design Parameters
Behavioral Parameters
Performance Parameters
Simulator
Post Processing
Indirect Modeling of Performance Parameters
Direct Modeling of Performance Parameters

Figure 9. *Direct and indirect modeling of the LNA performance parameters.*

four-dimensional (4-D) problem [32]. To illustrate the direct-modeling approach, we model the *IIP*2 performance parameter (denoting the second-order nonlinearity). We also use this approach to illustrate the scalability of the modeling algorithms used. Instead of fixing the number of inputs, we vary them from two to six. This will give insight into how the model accuracy and required number of data points changes as the dimensionality is increased from 2-D to six-dimensional (6-D). The relevant input parameters are the transistor width W, the source inductance L_s, the load resistance *RL*, the voltage bias of the transistor V_{GS}, the

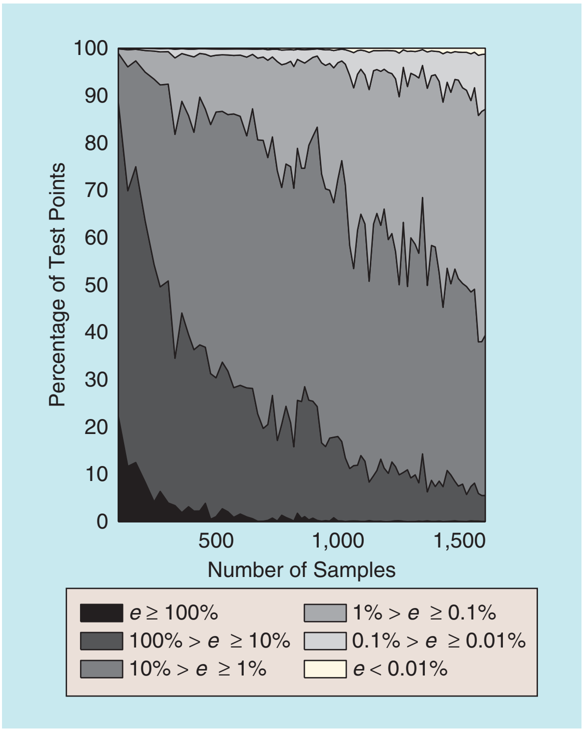


Figure 10. *Evolution of the true error histogram (using a relative error) over the reference data during ANN model generation for the 4-D indirect LNA modeling problem. More lighter regions mean a higher percentage of the reference points have a low error. The final model after 1,580 points has a mean relative error of 2.5%.*

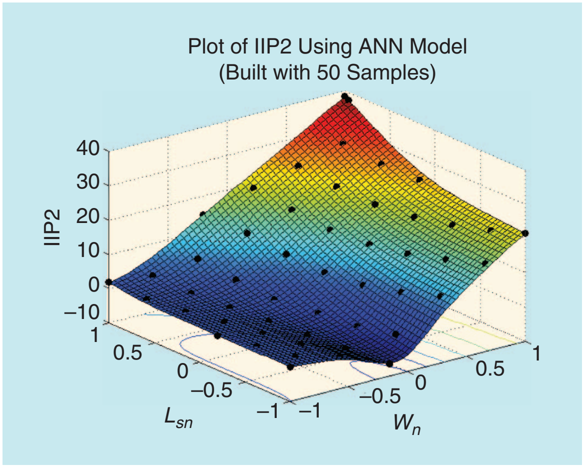


Figure 11. *Plot of the final 2-D ANN model for IIP2 with* $W = 100 * 10^{-6} * 10^{W_n}$ *m and* $Ls = 0.2 * 10^{-9} * 10^{L_{sn}}$ *H. Selected data samples are marked (·).*

transistor length *L*, and the resistance in series with the generator *RS* (the generator series resistance).

Modeling Settings

We use the same setup as the previous section but now only use ANN models since experience showed these to perform best on this problem. To assess the accuracy of the produced models objectively, reference test sets of size $51^2, 15^3, 11^4, 7^5, 5^6$ are available. Again, remember that these do not influence the modeling process itself.

Sampling Settings

The same sample selection algorithm is used as with the filter application, only this time there is no autosampling (the frequency is not treated specially) and the sample budget is extended to 1,580 points in the indirect case and 3,000 points in the direct case. The data source is now a custom MATLAB script instead of ADS Momentum.

Results

Figure 10 shows the evolution of the relative error histogram over the reference data for the indirect problem. After 1,580 points, the error histogram corresponds to a mean relative error over all test data of 2.5% for the final 4-D model.

For the direct-modeling problem, Figure 11 shows a visualization of the selected data samples and the model for the 2-D case. Figure 12 shows the true accuracy curve for each number of inputs from 2-D to 6-D. The true accuracy is calculated as the root relative square error (RRSE) on each test set. The RRSE is defined as

$$RRSE(\mathbf{y}, \tilde{\mathbf{y}}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{4}$$

where $\mathbf{y}, \tilde{\mathbf{y}}, \bar{y}$ are the true, predicted, and mean true response values, respectively.

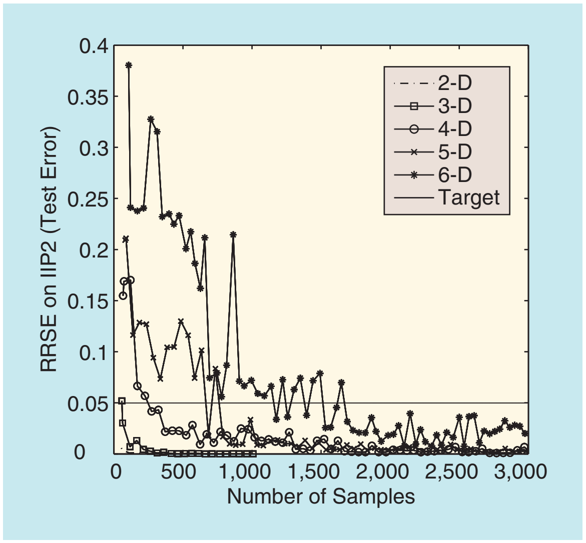


Figure 12. *Evolution of the true error across different dimensions when modeling the second-order nonlinearity (IIP2) of the LNA performance parameters.*

The curves in Figure 12 depict how good the LRM sample error combination is at minimizing the error on the reference grid. Desirable features are a smooth, monotonic decrease of the error as a function of the number of data points. The steeper the descent, the better. Erratic jumps should be avoided, but temporary increases in error are permitted. The error may temporarily increase if adding new data points reveals new features in the data or a new interpretation. What was thought to be a good model may turn out to be less accurate given the new information.

Figure 12 shows that satisfactory accuracy, which in this case is defined as a RRSE score of 0.05, can be reached in all cases, with convergence being particularly fast in the 2-D and three-dimensional (3-D) case. A second observation is that the curves for five-dimensional (5-D) and 6-D are rather erratic, much more so than the 2-D–4-D curves. The most likely reason for this lies in the fact that the LRM algorithm uses too few test points to estimate the model smoothness in higher dimensions [45]. Table 1 shows the running times on a Laptop with Intel(R) Core(TM) i7-2760QP CPU at 2.4 GHz with 8 GB of RAM and a 64-bit operating system.

Conclusion

The continuous emergence of new devices and circuit design techniques has led to the development of a wide range of behavioral modeling methods that facilitate design space exploration, optimization, and sensitivity analysis. This article discussed how the use of data modeling techniques can be leveraged in a flexible modeling framework to facilitate domain experts in their modeling task. By way of example, accurate behavioral models with good generalization and scalability were generated for EM bandstop filter and LNA modeling problems. Sample selection and model complexity setting were handled fully autonomously. Future work includes the integration of more specialized fitting algorithms (such as [49] or [51]) as plug-ins into the SUMO Toolbox so as to ensure maximum accuracy and interpretability of models for electronic devices and systems.

TABLE 1. Elapsed time to compute 2-D–6-D models for second order nonlinearity (*IIP*2) Of LNA performance parameters.

LNA	2-D	3-D	4-D	5-D	6-D
mins	0.4	1.2	24	61	192

For Further Information

The SUMO Toolbox framework which was used for the tests, including all algorithms and features discussed here, is available under an open source license (AGPLv3) for download at http://www.sumo.intec.ugent.be. The software package contains a collection of demos, data sets, and examples. Additional documentation is also included, and a wiki page with information is available at http://www.sumowiki.intec.ugent.be.

References

[1] A. Forrester, A. Sobester, and A. Keane, *Engineering Design Via Surrogate Modelling: A Practical Guide*. Hoboken, NJ: Wiley, 2008.

[2] D. P. Solomatine and A. Ostfeld, "Data-driven modelling: Some past experiences and new approaches," *J. Hydroinformat.*, vol. 10, no. 1, pp. 3–22, 2008.

[3] H. Fang, M. Rais-Rohani, Z. Liu, and M. Horstemeyer, "A comparative study of metamodeling methods for multiobjective crashworthiness optimization," *Comput. Struct.*, vol. 83, no. 25–26, pp. 2121–2136, 2005.

[4] A. Chinea, "Passive macromodeling of electrically long interconnects," Ph.D. dissertation, Dept. Elect. Eng., Polytech. Univ. Turin, Italy, 2010.

[5] M. Bakr, J. Bandler, K. Madsen, and J. Sondergaard, "Review of the space-mapping approach to engineering optimization and modeling," *Optim. Eng.*, vol. 1, no. 3, pp. 241–276, 2000.

[6] J. Bandler, R. M. Biernacki, S. H. Chen, P. A. Grobelny, and R. H. Hemmers, "Space mapping technique for electromagnetic optimization," *IEEE Trans. Microwave Theory Tech.*, vol. 42, pp. 2536–2544, Aug. 1994.

[7] J. De Geest, T. Dhaene, N. Faché, and D. De Zutter, "Adaptive CAD-model building algorithm for general planar microwave structures," *IEEE Trans. Microwave Theory Tech.*, vol. 47, pp. 1801–1809, Sept. 1999.

[8] D. Deschrijver and T. Dhaene, "Stability and passivity enforcement of parametric macromodels in time and frequency domain," *IEEE Trans. Microwave Theory Tech.*, vol. 56, pp. 2435–2441, Nov. 2008.

[9] T. Dhaene and D. Deschrijver, "Generalised vector fitting algorithm for macromodelling of passive electronic components," *IEEE Electron. Lett.*, vol. 41, no. 6, pp. 299–300, 2005.

[10] U. D. Annakkage, N. K. C. Nair, Y. F. Liang, A. M. Gole, V. Dinavahi, B. Gustavsen, T. Noda, H. Ghasemi, A. Monti, M. Matar, R. Iravani, and J. A. Martinez, "Dynamic system equivalents: A survey of available techniques," *IEEE Trans. Power Delivery*, vol. 27, no. 1, pp. 411–420, 2012.

[11] J. Sacks, W. J. Welch, T. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Stat. Sci.*, vol. 4, no. 4, pp. 409–435, 1989.

[12] E. S. Siah, M. Sasena, J. L. Volakis, P. Y. Papalambros, and R. W. Wiese, "Fast parameter optimization of large-scale electromagnetic objects using direct with kriging metamodeling," *IEEE Trans. Microwave Theory Tech.*, vol. 52, no. 1, pp. 276–285, 2004.

[13] Q. J. Zhang and K. C. Gupta, *Neural Networks for RF and Microwave Design (Book + Neuromodeler Disk)*. Norwood, MA: Artech House, 2000.

[14] A. H. Zaabab, Q. J. Zhang, and M. Nakhla, "A neural network modeling approach to circuit optimization and statistical design," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1349–1358, June 1995.

[15] J. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.

[16] M. B. Yelten, T. Zhu, S. Koziel, P. D. Franzon, and M. B. Steer, "Demystifying surrogate modeling for circuits and systems," *IEEE Circuits Syst. Mag.*, vol. 12, no. 1, pp. 45–63, 2012.

[17] D. Gorissen, K. Crombecq, I. Couckuyt, P. Demeester, and T. Dhaene, "A surrogate modeling and adaptive sampling toolbox for computer based design," *J. Mach. Learn. Res.*, vol. 11, pp. 2051–2055, 2010.

[18] D. Busby, C. L. Farmer, and A. Iske, "Hierarchical nonlinear approximation for experimental design and statistical data fitting," *SIAM J. Sci. Comput.*, vol. 29, pp. 49–69, Jan. 2007.

[19] C. Cherkassky and F. M. Mulier, *Learning from Data: Concepts, Theory, and Methods*. Hoboken, NJ: Wiley, 2007.

[20] P. Triverio, S. Grivet-Talocia, M. Nakhla, F. Canavero, and R. Achar, "Stability, causality and passivity in electrical interconnect models," *IEEE Trans. Adv. Packag.*, vol. 30, pp. 795–808, Nov. 2007.

[21] T. Dhaene and D. Deschrijver, "Efficient algorithm for passivity enforcement of s-parameter based macromodels," *IEEE Trans. Microwave Theory Tech.*, vol. 57, pp. 415–420, Feb. 2009.

[22] D. Deschrijver and T. Dhaene, "Fast passivity enforcement of s-parameter macromodels by pole perturbation," *IEEE Trans. Microwave Theory Tech.*, vol. 57, pp. 620–626, Feb. 2009.

[23] J. P. C. Kleijnen, *DASE: Design and Analysis of Simulation Experiments*. New York: Springer-Verlag, May 2007.

[24] G. Antonini, D. Deschrijver, and T. Dhaene, "Broadband rational macromodeling based on the adaptive frequency sampling algorithm and the partial element equivalent circuit method," *IEEE Trans. Electromagn. Compat.*, vol. 50, no. 1, pp. 128–137, 2008.

[25] M. Ding and R. Vemur, "An active learning scheme using support vector machines for analog circuit feasibility classification," in *Proc. 18th Int. Conf. VLSI Design*, Jan. 2005, pp. 528–534.

[26] T. G. Robertazzi and S. C. Schwartz, "An accelerated sequential algorithm for producing D-optimal designs," *SIAM J. Sci. Comput.*, vol. 10, pp. 341–358, Mar. 1989.

[27] Y. Lin, "An efficient robust concept exploration method and sequential exploratory experimental design," Ph.D. dissertation, Georgia Inst. Technol., 2004.

[28] A. C. Keys and L. P. Rees, "A sequential-design metamodeling strategy for simulation optimization," *Comput. Oper. Res.*, vol. 31, no. 11, pp. 1911–1932, 2004.

[29] L. Zhang, Y. Cao, S. Wan, H. Kabir, and Q.-J. Zhang, "Parallel automatic model generation technique for microwave modeling," in *Proc. IEEE/MTT-S Int. Microwave Symp.*, June 2007, pp. 103–106.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM Algorithm," *J. Roy. Stat. Soc. Series B Methodol.*, vol. 39, no. 1, pp. 1–38, 1977.

[31] D. Gorissen, T. Dhaene, and F. De Turck, "Evolutionary model type selection for global surrogate modeling," *J. Mach. Learn. Res.*, vol. 10, pp. 2039–2078, 2009.

[32] D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene, "Sequential modeling of a low noise amplifier with neural networks and active learning," *Neural Comput. Appl.*, vol. 18, pp. 485–494, June 2009.

[33] D. Gorissen, "Grid-enabled adaptive metamodeling and active learning for computer based design," in *Proc. 22nd Canadian Conf. Artificial Intelligence* (Lecture Notes in Artificial Intelligence, vol. 5549), Kelowna, BC, May 2009, pp. 266–269.

[34] E. R. Harold and W. S. Means, *XML in a Nutshell*. Cambridge, MA: O'Reilly Media, 2004.

[35] D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene, "Multiobjective global surrogate modeling, dealing with the 5-percent problem," *Eng. Comput.*, vol. 26, pp. 81–89, Jan. 2010.

[36] D. Gorissen, T. Dhaene, P. Demeester, and J. Broeckhove, "Grid enabled surrogate modeling," in *Handbook of Research on Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Applications*, . IGI Global, May 2009, pp. 249–258.

[37] I. Couckuyt, D. Gorissen, H. Rouhani, E. Laermans, and T. Dhaene, "Evolutionary regression modeling with active learning: An application to rainfall runoff modeling," in *Proc. Int. Conf. Adaptive and Natural Computing Algorithms (LNCS 5495)*, Sept. 2009, pp. 548–558.

[38] T. Zhu and P. D. Franzon, "Application of surrogate modeling to generate compact and PVT-sensitive IBIS models," in *Proc. 18th Conf. Electrical Performance of Electronic Packaging and Systems*, Oct. 2009. pp. 77–80

[39] D. Gorissen, K. Crombecq, I. Couckuyt, and T. Dhaene, "Automatic approximation of expensive functions with active learning," in *Foundations of Computational Intelligence, vol. 1: Learning and Approximation: Theoretical Foundations and Applications* (Series Studies in Computational Intelligence, vol. 201), A.-E. Hassanien, A. Abraham, A. V. Vasilakos, and W. Pedrycz Eds. New York: Springer-Verlag, 2009, pp. 35–62.

[40] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *J. Global Optim.*, vol. 13, pp. 455–492, Nov. 1998.

[41] W. Hendrickx and T. Dhaene, "Sequential design and rational metamodelling," in *Proc. Winter Simulation Conf.*, M. Kuhl, S. N. M., F. B. Armstrong, and J. A. Joines, Eds. Dec. 2005, pp. 290–298.

[42] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Proc. 7th NIPS Conf. Advances in Neural Information Processing Systems*, Denver, CO, Dec. 1994, pp. 231–238.

[43] D. MacKay, "Bayesian model comparison and backprop nets," in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. San Mateo, CA: Morgan Kaufmann, Dec. 1992, pp. 839–846.

[44] F. Foresee and M. Hagan, "Gauss-Newton approximation to Bayesian regularization," in *Proc. Int. Joint Conf. Neural Networks*, June 1997, pp. 1930–1935.

[45] H. Nguyen, I. Couckuyt, L. Knockaert, T. Dhaene, D. Gorissen, and Y. Saeys, "An alternative approach to avoid overfitting for surrogate models," in *Proc. Winter Simulation Conf.*, Dec. 2011, pp. 2760–2771.

[46] K. Crombecq, D. Gorissen, D. Deschrijver, and T. Dhaene, "A novel hybrid sequential design strategy for global surrogate modeling of computer experiments," *SIAM J. Sci. Comput.*, vol. 33, pp. 1948–1974, July 2011.

[47] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall, Apr. 1986.

[48] T. Lee, *The Design of CMOS Radio-Frequency Integrated Circuits*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[49] D. Deschrijver, T. Dhaene, and D. De Zutter, "Robust parametric `macromodeling using multivariate orthonormal vector fitting," *IEEE Trans. Microwave Theory Tech.*, vol. 56, pp. 1661–1667, July 2008.

[50] D. Deschrijver, F. Vanhee, D. Pissoort, and T. Dhaene, "Automated near-field scanning algorithm for the EMC analysis of electronic devices," *IEEE Trans. Electromagnetic Compatibility,* vol. 54, no. 3, pp. 502–510, June 2012.

[51] T. Dhaene and D. Deschrijver, "Stable parametric macromodeling using a recursive implementation of the vector fitting algorithm," *IEEE Microwave and Wireless Components Letters,* Vol. 19, No. 2, pp. 59–61, Feb. 2009.