

Neural Network Design for Behavioral Model Generation with Shape Preserving Properties

Oleg Mikulchenko and Kartikeya Mayaram

Department of Electrical and Computer Engineering
Oregon State University, Corvallis, OR 97331

Abstract

A practical method for the design of artificial neural networks (ANN) for behavioral modeling of various devices is presented. The approach has been applied to the modeling of micro-flow sensors and MOS transistors. It is shown that the ANN based behavioral models are accurate and computationally efficient.

Keywords— *Behavioral modeling, simulation, device modeling, neural networks, micro-flow sensor modeling, MOS transistor modeling.*

I. INTRODUCTION

Behavioral modeling is important in a top-down design methodology [1]. The hierarchy and abstraction resulting from behavioral models makes it possible to simulate complex systems in an efficient manner. The performance of a behavioral simulation of a complex system is determined by the performance of the behavioral models for the various components and subsystems. Therefore, accurate and computationally efficient behavioral models are required for the nonlinear components of a system.

Conventional approaches to behavioral modeling include use of tables [2], B-splines [3], [4], polynomials [5] and artificial neural networks (ANN) [6], [7], [8]. The most promising of these are the ANN-based behavioral models since ANNs can approximate highly nonlinear characteristics. In addition, ANNs possess highly desirable characteristics for behavioral modeling. They produce reasonable outputs for inputs not encountered during training (learning). Furthermore, ANNs work well with noisy and incomplete data.

However, existing approaches for ANN design do not provide ANNs guaranteed to be free from under-fitting (large errors) or over-fitting (large oscillations for small errors) of data [9]. Moreover, as pure approximators, ANNs can approximate a given data set, but with features not present in the original data. These include a local non-monotonical approximation for a monotonical function, a local negative approximation

for a positive function, and false oscillations. Because of these undesirable features the simulation accuracy is compromised. Traditional ANN based algorithms for model generation [7], [8] do not guarantee shape preserving approximations, which are essential for accurate simulation.

In this paper, an algorithm for creation of ANNs with shape-preserving properties for multivariate approximations is described. The paper is organized as follows. An overview of ANNs and techniques for behavioral model generation are described in Section II. Examples of ANN-based behavioral models are presented in Section III and conclusions are provided in Section IV.

II. ANN-BASED BEHAVIORAL MODEL GENERATION

ANNs are nonlinear mapping schemes the structure of which corresponds to the nervous system. An ANN consists of simple units, called neurons, which can be combined in a network resulting in a complicated structure. The most widely used ANN are the multi-layer perceptrons (MLP), as shown in Fig. 1. These are a subset of the feed-forward ANN [10]. The design of an ANN consists of choosing the connection topology and the connection weights. The information processing by a single neuron is shown in Fig. 2. These neurons have a smooth nonlinear transformation function as shown in the figure. Neurons in the output layer employ a linear transformation function.

MLPs with two hidden layers can approximate any bounded continuous function with an arbitrary accuracy [10], [11], [12]. A MLP with one hidden layer is not as general but can also be used to approximate most functions.

In the design of ANN based behavioral models there are several key considerations. These are:

Definition of the objective function: The optimal ANN must be shape preserving and have small

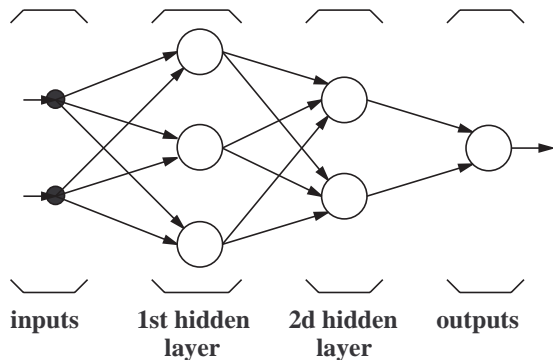


Fig. 1. A multi-layer perceptron. The network consists of artificial neurons with weighted connections. The network has a feed-forward structure, i.e., there are no connection loops. The network implements a static nonlinear mapping $\mathbb{R}^{N_{inputs}} \Rightarrow \mathbb{R}^{N_{outputs}}$.

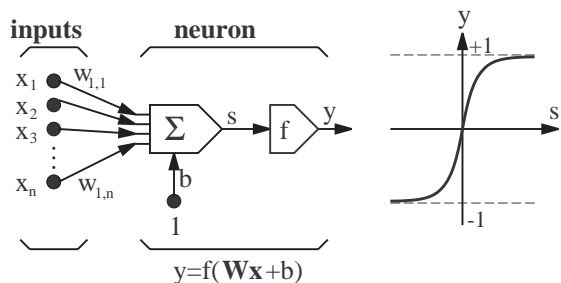


Fig. 2. Information processing by one artificial neuron. A neuron collects weighted input signals and nonlinearly transforms a sum to the output.

errors.

Design of the ANN topology: The optimal ANN structure depends on the number of neurons in the hidden layers.

Training of ANN: A fast and reliable optimization algorithm is required for the adjustment of ANN weights.

Design of experiments: The data set used to train the ANN must provide a reasonable approximation for data not used to train the ANN.

A. Definition of the objective function

The objective function that is minimized is of the following form:

$$f(\mathbf{x}, \mathbf{V}) = f_1(\mathbf{x}, \mathbf{V}) * f_2(\mathbf{x}, \mathbf{V}) \quad (1)$$

where

$$f_1(\mathbf{x}, \mathbf{V}) = \left(\frac{1}{MN_o} \sum_{i=1}^{N_o} \sum_{j=1}^M (f_{NN,i}(\mathbf{x}, \mathbf{v}_j) - y_{j,i})^2 \right)^{1/2} \quad (2)$$

and

$$f_2(\mathbf{x}, \mathbf{V}) = \left(\sum_{k=1}^{N_{inp}} N_{sign,k}(\mathbf{x}, \mathbf{V}) + 1 \right) \quad (3)$$

where N is the number of weights, $\mathbf{x} \in \mathbb{R}^N$ is a vector of ANN weights, N_{inp} is the number of inputs, N_o is the number of outputs, M is the number of points in the data set, $\mathbf{V} \in \mathbb{R}^{N_{inp} \times M}$ is the input data set, $\mathbf{v}_j \in \mathbb{R}^{N_{inp}}$ is the j -th input vector, $\mathbf{f}_{NN} : \mathbb{R}^{N_{inp}} \Rightarrow \mathbb{R}^{N_o}$ is the ANN mapping, $\mathbf{Y} \in \mathbb{R}^{N_{out} \times M}$ is the output data set, and $N_{sign,k}$ is the number of changes in the sign of the partial derivatives $(\partial \mathbf{f}_{NN}(\mathbf{x}, \mathbf{V}) / \partial V_k)$ of the NN output with respect to the inputs \mathbf{V} .

The first factor of Eq. (1) is the standard root mean square error (RMSE) as given in Eq. (2). This is used to minimize the error between the data points and the ANN. The second factor in (1) is a measure of the numbers of oscillations in the function $\mathbf{f}_{NN}(\mathbf{x}, \mathbf{V})$. The minimum of the objective function yields both the shape reconstruction (a small number of the oscillations expressed by $N_{sign,k}$), and good accuracy (expressed by the RMSE). Thus, an optimum for the objective function avoids the over-fitting and under-fitting of data, a problem common to regular ANN [10].

No parameters have to be determined for the objective function described above. This is an important property, because the adjustable parameters in an objective function can affect the optimization performance and the quality of results in unknown ways.

B. Design of ANN topology

The objective function is a non-smooth function of parameters \mathbf{x} . Therefore, conventional optimization techniques cannot be applied for the minimization of the objective function in Eq. (1). Moreover, even for a smooth RMSE, application of traditional optimization methods does not guarantee a global minimum [10].

Genetic algorithms [13] can be applied to minimize Eq. (1). However, our investigations show this approach takes a very long time. A simpler and more efficient algorithm consists of a few repetitions of deterministic optimization with multiple random initializations.

The objective function is directly proportional to the RMSE. Therefore we can use RMSE (Eq. (2)) as

a function to be minimized for the deterministic local optimization. A multiple number of neural networks are generated randomly and optimized based on the training data. During this step the number of neurons are varied. From this collection of neural networks the best neural network is selected which yields ANN with structure and parameters near a global optimum.

For improved optimization performance and to obtain a model valid for a wide parameter range we use nonlinear scaling and normalization.

C. Training of ANN

Back-propagation [14], [15] is the most commonly used method for training multi-layer feed-forward ANN. From an optimization point of view, this method is equivalent to gradient descent optimization. The method is very simple and efficient for ANN programming, but has slow convergence [16]. Other approaches for ANN training are the conjugate gradient methods [17] and the Levenberg-Marquardt algorithm [18], [19]. For a small and moderate ANN, the Levenberg-Marquardt algorithm is preferred, whereas the conjugate gradient methods are preferred for many weights.

In this paper, our focus is on relatively small ANN. Therefore, we use the Levenberg-Marquardt algorithm. The method is applied to the nonlinear least squares problem:

Minimize $f(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T * \mathbf{r}(\mathbf{x}) = \sum_{j=1}^M (r_j(\mathbf{x}))^2$ where $\mathbf{x} \in \mathbb{R}^N$, and $\mathbf{r}(\mathbf{x})$ is the residual vector (the difference between the ANN model and the data). This results in the following iteration:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{J}(\mathbf{x}_k)^T \mathbf{J}(\mathbf{x}_k) + \mu_k \mathbf{I})^{-1} \mathbf{J}(\mathbf{x}_k)^T \mathbf{r}(\mathbf{x}_k), \quad (4)$$

where $\mathbf{J}(\mathbf{x}) \in \mathbb{R}^{N \times M}$, $\mathbf{J}(\mathbf{x}) = \partial \mathbf{r}(\mathbf{x}) / \partial \mathbf{x}$ is the Jacobian matrix and \mathbf{I} is the N -dimensional identity matrix.

Our experience shows that different implementations of the Levenberg-Marquardt method have different performance for different tasks. In general, two approaches are preferred (i) explicit control of μ_k as implemented in the MATLAB neural network toolbox [20] and (ii) trust region algorithms [21].

D. Design of Experiments

Design of experiments [22] is a mature field in process development and process improvement and also in polynomial model generation. The techniques for design of experiments (response surface design, factorial design, I-optimal design) use simple polynomial models and provide a small number of points to determine the coefficients of models. However, such meth-

ods do not yield a good sampling for accurate highly nonlinear model generation. Therefore, the design of simulation experiments for general models continues to be a challenging problem.

The use of uniform grids (one-variable-at-a-time) is a very inefficient approach for multivariate models because of a large number of data points. To avoid a big data set and to obtain a distribution with a good uniformity over each variable and over the multidimensional unit hypercube low-discrepancy sequences have been developed [23]. Low-discrepancy sequences are the basic ingredients of quasi-Monte Carlo methods, e.g., for numerical integration and global optimization with an improved performance. We use the low-discrepancy Sobol's ($\Lambda\Pi_\tau$) quasi-random number generator [24] for data points used in model generation. Our testing shows that this algorithm is superior in comparison with other generators (e.g., Halton sequences, Faur sequences) and with a uniform grid. We use the $\Lambda\Pi_\tau$ generator to obtain different point density distributions via the inversion function method.

III. ANN-BASED BEHAVIORAL MODELS

In this section the ANN-based behavioral modeling is demonstrated on two very different examples. The first example is of MOSFET modeling where the ANN is used to provide an accurate DC model for the transistor. The second example is that of a micro-flow sensor which can only be modeled using a solution of the partial differential equations describing the device. These examples illustrate the generality of the modeling technique whereby this approach can be applied to other type of devices.

A. MOSFET Transistor Modeling

ANN based models speed up the simulation time and can therefore be used instead of accurate (but slow) physics based models. For this reason, we consider the application of ANN to MOSFET modeling. For illustration purposes the circuit simulator SPICE3f5 is used to generate the data for the BSIM3 MOSFET model [25]. In a real application, measured data will be used to train the ANN. The model takes 5 input arguments (channel width W , channel length L , gate-source voltage V_{gs} , drain-source voltage V_{ds} , and the bulk-source voltage V_{bs}) and the output is the drain current I_d . The training data set was obtained by the ($\Lambda\Pi_\tau$) quasi-random number generator with piecewise linear distribution functions for $V_{ds} \in [0, 3]$ V, $V_{gs} \in [0, 3]$ V, and for uniform distribution

functions for $W \in [1, 100] \mu\text{m}$, $L \in [0.5, 100] \mu\text{m}$, and $V_{bs} \in [-3, 0] \text{V}$. The range of I_d is $[1e-9, 1e-2] \text{A}$. Hence, a nonlinear scaling was applied for V_{ds} , V_{gs} , and I_d .

The nonlinear scaling is described by the transform:

$$x' = (\log(x) + a)/b,$$

where a and b are chosen such that $x'_{min} = -1$, $x'_{max} = 1$. This scaling results in a

- better condition number for the Jacobian used in the Levenberg-Marquardt method
- smoothing of the nonlinearity for improved optimization performance
- efficient stochastic search with random initialization of NN weights and biases
- the same optimum solution for both the absolute error of the scaled variable x' and for the relative error of the original variable x
- the same percentage error estimate for each scaled output value

Small-signal analysis is important for analog circuit design. Therefore, good agreement is required not only for I_d , but also for its derivatives $g_{ds} = \partial I_d / \partial V_{ds}$, and $g_m = \partial I_d / \partial V_{gs}$. The results for the ANN based model for different parameter values are shown in Figs. 3 and 4.

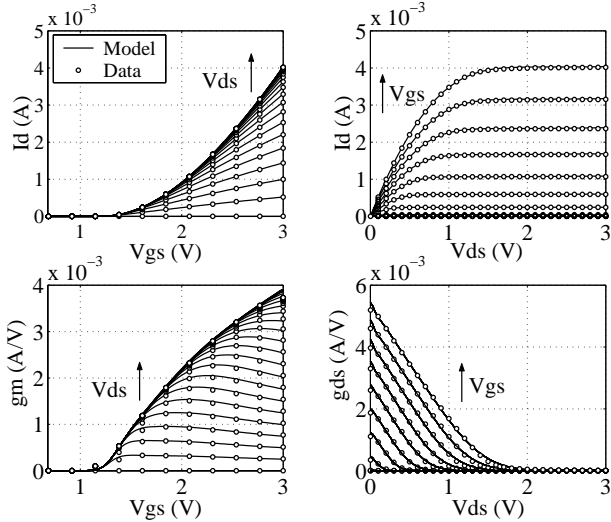


Fig. 3. MOSFET modeling by ANN. $W=45 \mu\text{m}$, $L=0.5 \mu\text{m}$, $V_{bs}=-2 \text{V}$. Comparisons are provided for data not used in the training set.

From these figures the following observations can be made

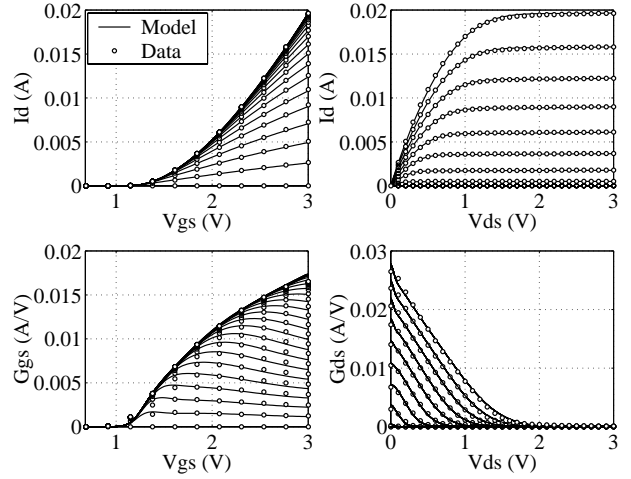


Fig. 4. MOSFET modeling by ANN. $W=100 \mu\text{m}$, $L=1 \mu\text{m}$, $V_{bs}=-2 \text{V}$. Comparisons are provided for data not used in the training set.

- The conductances g_{ds} and g_m are accurately modeled
- Both the high current regimes ($I_d \in [1e-3, 1e-2] \text{A}$), and low currents ($I_d \in [1e-9, 1e-3] \text{A}$) are accurately modeled
- The current I_d is very close to zero for $V_{ds}=0$ ($|I_d| < 1e-12 \text{A}$)

B. Micro-flow Sensor Modeling

The micro-flow sensor is an important component of a microfluidic system. We have applied the ANN to model an anemometer type micro-flow sensor [26] as shown in Fig. 5. The data set for training the ANN was obtained from finite-element solutions of the partial differential equations describing the device operation.

The ANN takes 3 input arguments (flow velocity U , sensor separation d and channel height h) and yields the temperature difference ΔT , the temperature of the heater T_2 , and the temperature of the upstream sensor T_1 .

The sequence of steps used to obtain the ANN model is outlined below:

- A numerical solution is obtained for ΔT , T_1 , and T_2 for different values of U , d , h
- This solution yields discrete data points for ΔT , T_1 , and T_2 as a function of U , d , and h
- The discrete data points are then approximated by neural networks

In our modeling approach, we have used a nonlinear scaling for U , h , and ΔT (because these values vary

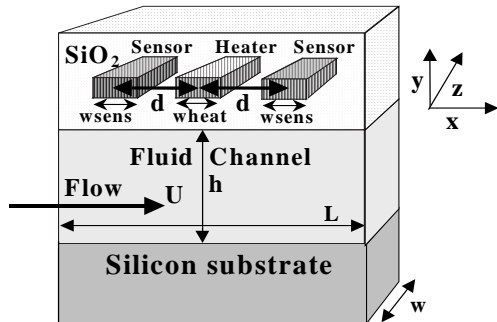


Fig. 5. Structure of an anemometer type flow sensor. This sensor is made up of a heating element and two sensing elements. In this figure, the upstream sensor is placed to the left of the heater. The downstream sensor is placed to the right of the heater. The temperature difference between the downstream and upstream sensors is used to measure the flow.

over a large interval) and a linear scaling for d , T_1 and T_2 .

A good agreement was obtained for a large range of input variables. The ANN model can be used to predict ΔT for parameter values that were not included in the training set for the ANN. An example of such a result is shown in Figure 6 where the model is compared with simulation data for a channel height of $50 \mu\text{m}$. It should be noted that this data was not included in the training set and yet excellent agreement between the model and data is obtained.

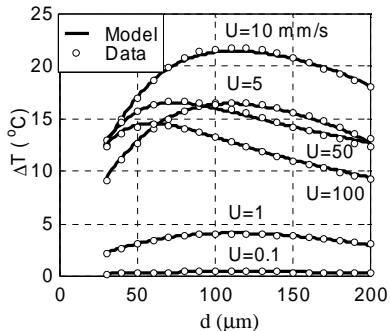


Fig. 6. Predicted ΔT versus separation d for different velocities U with a channel height $h=50 \mu\text{m}$. The channel height of $50 \mu\text{m}$ was not used in the training set.

We have considered a micro-flow sensor with water as the fluid. The solutions of PDEs were used to determine T_1 , T_2 , and ΔT for different velocities in the range $U=[0, 10]$ m/s, separation in the range $d=[30, 200]$ μm and channel height in the range $h=[2, 200]$ μm . The data set consisted of 1100 points.

The optimal ANN-based model (repetitions of ANN training for multiple random initializations) was obtained in two days on a PC with a Pentium-II 400 MHz processor. The time required for evaluation of one data point using the ANN-model is $80 \mu\text{s}$ compared with 300 s for one PDE solution.

IV. CONCLUSION

In this paper we have presented a technique for developing ANN-based behavioral models which possess the correct physical behavior. The ANN models are simple, accurate, have shape preserving properties, and are applicable for a wide range of device parameters. The models can be generated from simulated or measured data. Two examples, a MOS-transistor and a micro-flow sensor are used to demonstrate the application of the ANN-based modeling technique. The performance of ANN-based models is up to 7 orders of magnitude higher compared with PDE solutions. The approach described in this paper is general and can be used to create fast and accurate behavioral models of other components or systems. Furthermore, the models can be incorporated in VHDL-AMS or Verilog-AMS environments for behavioral simulation.

V. ACKNOWLEDGMENTS

This work is supported in part by DARPA under agreement number F30602-98-2-0178 and by NSF grant CCR-9702292.

REFERENCES

- [1] H. Chang, E. Charbon, U. Choudhury, and A. Demir, *A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits*. Kluwer Academic Pub, 1997.
- [2] P. Meijer, "Fast and smooth highly nonlinear table models for device modeling," *IEEE Transactions on Circuits and Systems*, vol. 37, pp. 335–346, March 1990.
- [3] J. Peters and U. Reif, "Analysis of algorithms generalizing B-spline subdivision," vol. 35, pp. 728–748, Apr. 1998.
- [4] C. de Boor and A. Ron, "On multivariate polynomial interpolation," *Constructive Approximation*, vol. 6, pp. 287–302, 1990.
- [5] L. M. Kocic and G. V. Milovanovic, "Shape preserving approximations by polynomials and splines," *Computers and Mathematics with Applications*, vol. 33, pp. 59–97, June 1997.
- [6] Q. J. Zhang and K. C. Gupta, *Neural Networks for Rf and Microwave Design*. Artech House, 2000.

- [7] P. Burrascano and M. Mongiardo, "A review of artificial neural networks applications in microwave CAD," *International Journal on RF and Microwave Computer-Aided Engineering*, vol. 9, no. 3, 1999.
- [8] G. Creech and J. Zurada, "Neural network modeling of semiconductor material and device characteristics," *International Journal on RF and Microwave Computer-Aided Engineering*, vol. 9, no. 3, 1999.
- [9] S. Lawrence, C. L. Giles, and A. Tsoi, "Lessons in neural network training: Overfitting may be harder than expected," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI-97*, pp. 540–545, Menlo Park, California: AAAI Press, 1997.
- [10] R. D. Reed and R. J. Marks, *Neural smithing: supervised learning in feedforward artificial neural networks*. The MIT Press, 1999.
- [11] A. Lapedes and R. Farber, "How neural nets work," in *Neural Information Processing Systems* (D. Z. Anderson, ed.), (New York), pp. 442–456, (Denver 1987), American Institute of Physics, 1988.
- [12] Y. Takahashi, "Generalization and approximation capabilities of multilayer networks," *Neural Computation*, vol. 5, no. 1, pp. 132–139, 1993.
- [13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading: Addison-Wesley, 1989.
- [14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.
- [15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing* (D. E. Rumelhart and J. L. McClelland, eds.), vol. 1, ch. 8, pp. 318–362, Cambridge: MIT Press, 1986.
- [16] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimizations*. Academic Press, 1981.
- [17] D. F. Shanno, "Conjugate gradient methods with inexact searches," *Math. Oper. Res.*, vol. 3, no. 3, pp. 244–256, 1978.
- [18] K. Levenberg, "A method for the solution of certain nonlinear problems in least squares," *Quart. Appl. Math.*, vol. 2, pp. 164–168, 1944.
- [19] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, pp. 431–441, June 1963.
- [20] H. Demuth and M. Beale, "Neural network toolbox." For use with MATLAB. User's guide. Version 3.0.
- [21] J. J. E. Dennis and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, vol. 16 of *Classics in Applied Mathematics*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1996. Corrected reprint of the 1983 original.
- [22] D. Montgomery, *Design and Analysis of Experiments*. John Wiley and Sons, New York, NY, USA, 1991.
- [23] H. Niederreiter and C. Xing, "Quasi-random points and global function fields," in *Finite Fields and Applications* (S. Cohen and H. Niederreiter, eds.), London Math. Soc. LN Series 233, pp. 269–296, 1996.
- [24] P. Bratley and B. L. Fox, "Implementing Sobol's quasirandom sequence generator," *ACM Trans. Math. Softw.*, vol. 14, pp. 88–100, Mar. 1988.
- [25] P. Ko, J. Huang, Z. Liu, and C. Hu, "BSIM3 for analog and digital circuit simulation," in *IEEE Symp. on VLSI Technology CAD*, pp. 400–429, 1993.
- [26] O. Mikulchenko, A. Rasmussen, and K. Mayaram, "A neural network based macromodel for microflow sensors," *Proc. of MSM2000*, pp. 540–543, 2000.