Novel Neural Network application to nonlinear electronic devices modeling: building a Volterra series model

Georgina Stegmayer[1], Omar Chiotti[2]

[1] Politecnico di Torino, Electronics Department, Cso. Duca degli Abruzzi 24, 10129 Torino, Italia
georgina.stegmayer@polito.it
http://www.eln.polito.it

[2] G.I.D.S.A.T.D., Universidad Tecnológica Nacional, Lavaise 610, 3000 Santa Fe, Argentina
gidsatd@frsf.utn.edu.ar
http://www.frsf.utn.edu.ar/investigacion/grupos/gidsatd/

Abstract. In this work we want to present a novel application of Neural Networks as a Black-Box model, which allows representing the nonlinear behavior of a vast number of RF electronic devices with the Volterra series approximation. We propose a simple approach for the generation of the Volterra model for a device, even in the case of a nonlinearity that depends on more than one variable, which allows obtaining a general model, independent of the physical circuit. In particular, we will show the results obtained for the analysis of a transistor and the generation of its analytical Volterra series model, built using a standard Neural Network model and its parameters.

Keywords: neural networks, nonlinear modeling, electronic devices, Volterra model

1 Introduction

The classical representation of a nonlinear electronic device is, usually, an equivalent circuit, whose current-voltage and charge-voltage relationships have to be determined according to the formalisms of the Electronic Circuit theory (Fig. 1). Currents and voltages obtained are mathematical quantities related to the circuit model. This procedure is based on the known physical behavior of the modeled device that dictates the model topology. For this reason, not only the model must be tailored to the device, but also the extraction of its parameters strongly depends on the device under study.

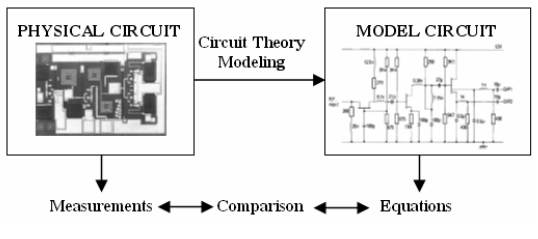

Fig. 1. Classical representation of a nonlinear electronic device

Parameters extraction is the procedure of comparing the currents and voltages obtained from the circuit model, against those measured in the physical circuit. The results are a piece of information on the functioning of the physical circuit. If they do not compare satisfactorily, a more adequate model must be found [1].

A different approach, that claims to represent a general device, is referred to as a *Black-Box* model, which tries to estimate both the functional form of the relations between variables, and the numerical parameters in those functions. Neural Networks are often used for Black-Box models estimation, because, usually, they do not assume almost anything about the incoming data. In fact, the Neural Network approach for electronic device modeling has received increasing attention, especially in the recent years [2][3], since model tailoring to the device under study only needs a training procedure based on experimental data or measurements of the physical circuit.

On the other hand, as an analytical model, the Volterra expansion approach has been proposed since many years for nonlinear electronic devices modeling [4]. The Volterra model is a numerical series representation which has some particular terms named *kernels* that contain the derivative information of the model, of particular interest for the electronics engineer. Due to the difficulties and complexity in identifying the higher order kernels through experimental data, this approach is not effectively used within commercially available circuit simulators.

The objective of our work is to develop a general Black-Box model, independent of the physical circuit modeled, which receiving only simple measurements of the device and using a Neural Network based approach, could generate the Volterra Series Model of the nonlinear electronic device, using the network parameters to calculate the kernels (Fig. 2). We propose a simple approach, which can be applied even in the case of a nonlinearity that depends on more than one variable, and that allows obtaining a general model, independent of the physical circuit under study. In this paper we show the results of the application of this novel Black-Box Neural Network based model to a case of study: the nonlinear current/voltages characteristic in a transistor.

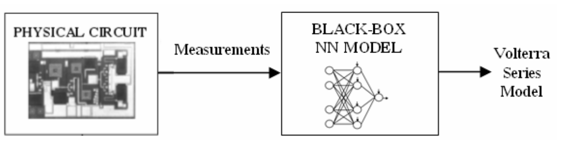

Fig. 2. Novel Black-Box Neural Network based approach proposed for the representation of a nonlinear electronic device through a Volterra model

The organization of the paper is the following: the Volterra Model of a device is shown on Section 2. The fundamentals of Volterra Series are presented on Section 3. Our proposed approach appears in Section 4 and its implementation in Section 5. A Case of Study and some simulation results are presented in Section 6. Finally, the conclusions and the future work appear on Section 7 and 8, respectively.

2 Volterra Model of a Nonlinear Electronic Device

The classic representation of a nonlinear electronic device is an equivalent circuit, with equations that relate its components, such as the current-voltage and charge-voltage relationships in a transistor. These kinds of models, however, only describe the input/output behavior of the device/element, that is to say, they only reflect the output response of the device to determined input variables. An example of this type of model is a generally known and widely accepted Field Effect Transistor (FET) model, the Curtice Model [5]. The equivalent circuit representation for this model can be seen in Fig. 3. One element of this model is the nonlinear behavior of the drain to source current Ids that is function of the intrinsic voltages of the FET, Vgs and Vds. Simulations performed in a microwave circuits simulator show the I/V (current vs. voltage) curves of the model (Fig. 4), for different voltages combinations, using the following parameters values: *Vgs = [-1…0] step 0.2, Vds = [0…5] step 0.5,* $\beta = 0$, $\chi=0.3$.

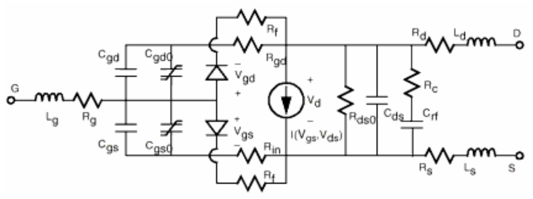

Fig. 3. Curtice FET model equivalent circuit

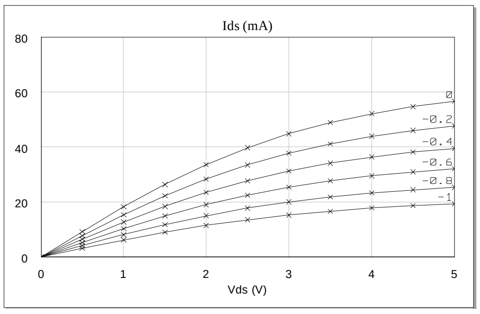

Fig. 4. I/V Curves for the Curtice model

The Curtice model, however, only describes the input/output characteristic of the transistor, but does not allow a deeper analysis about the device behavior that could help an electronic designer. One of the approaches regarding this analysis is the approximation of the nonlinear behavior of the element under consideration with a numerical series such as the Volterra series. This series has some terms named *kernels* that allow a deeper understanding of the device. Its main disadvantage, however, is the analytical expression or calculation of the kernels. Our proposal is to use a Neural Network and its parameters, to help in the building of the Volterra model for a device, and the calculation of its kernels.

The Volterra series analysis is well suited to the simulation of nonlinear microwave devices and circuits, in particular in the weakly nonlinear regime where a few number of kernels are able to capture the nonlinearity in the device behavior. The Volterra kernels allow the analysis of device characteristics of great concern for the microwave designer, such as harmonic generation and inter-modulation phenomena in the case of a FET transistor [4], information that can be inferred from the derivatives of the model. That is to say, it is necessary not only to have a device model that reflects the relationship between its input/output variables, but also its derivatives are important for the device design phase.

The Volterra description for an electronic device is based on Taylor-series expansions of the device nonlinearity around a fixed bias point or around a time-varying signal [6]. Example of the first type of model is shown in Equation (1), the well-known Volterra model for the *Ids* current in a FET transistor (in this example, the series expansion includes only up to third order terms), where *Vgs0* and *Vds0* are the internal bias voltages and *vgs* and *vds* are the incremental intrinsic voltages with respect to the bias voltages.

$$\begin{aligned} Ids = {} & Ids(Vgs0, Vds0) + \\ & Gm1.vgs + Gds.vds + \\ & Gm2.vgs^2 + Gds2.vds^2 + Gmd.vgs.vds + \\ & Gm3.vgs^3 + Gds3.vds^3 + \\ & Gm2d.vgs^2.vds + Gmd2.vgs.vds^2 \end{aligned} \tag{1}$$

The coefficients of the series are the first order (*Gm1* and *Gds*), second order (*Gm2*, *Gds2* and *Gmd*) and third order (*Gm3, Gds3, Gm2d, Gmd2*) derivatives of the current with respect to the voltages. These coefficients happen to be the Volterra kernels of the series. The coefficient *Gm1* is the first derivative of the current with respect to the *vgs* voltage and it is called *transconductance*, a parameter of great concern to the electronics engineer because it is an expression of the performance of the transistor. In general, the larger the transconductance for a device, the greater the gain (amplification) it is capable of delivering, when all other factors are held constant.

The number of terms in the kernels of the series increases exponentially with the order of the kernel. This is the most difficult problem with the Volterra series approach and imposes some restrictions on its application to many practical systems, which are restricted to use second order models because of the difficulty in kernels expression. A common approach for kernels identification is the use of Wiener orthogonal functions, but the problem of number of terms in the functions is still present [7]. Moreover, the use of alternative methods for kernels identification [8], analytical expression [9] or measurement [10] can be a complex and time-consuming task; i.e. in the case of kernel measurements, it is quite difficult to separately detect the individual contributions of each Volterra operator from the global system response. In the next Section the fundaments of the Volterra Series analysis are presented together with some previously proposed approaches for kernels calculation. In Section 4, our proposed approach is explained, based on the use of a Neural Network model.

3 Fundamentals of the Volterra Series Analysis

A single-input, single-output (SISO) non-linear dynamical system, with an output time function, *y(t)*, and an input time function, *x(t)*, can be represented exactly by a converging infinite series of the form

$$y(t) = \sum_{1}^{\infty} y_n(t) \tag{2}$$

$$y_n(t) = \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} h_n(\tau_1 \dots \tau_n) \prod_{j=1}^{n} x(t - \tau_j) d\tau_j \tag{3}$$

This system can be represented to any desired degree of accuracy by a finite series of the form (4). This equation is known as the Volterra series expansion or “power series with memory”, because the actual output of the system depends on the past values of the inputs. The $h_1, h_2, \dots, h_n$ are known as the *Volterra kernels* of the system [11].

$$\begin{aligned} & y(t) = h_0 + \\ & \int_{\tau=0}^{\infty} h_1(\tau) x(t-\tau) d\tau + \\ & \int_{\tau_1=0}^{\infty} \int_{\tau_2=0}^{\infty} h_2(\tau_1, \tau_2) x(t-\tau_1) x(t-\tau_2) d\tau_1 d\tau_2 + \dots + \\ & \int_{\tau_1=0}^{\infty} \dots \int_{\tau_n=0}^{\infty} h_n(\tau_1, \tau_2, \dots, \tau_n) x(t-\tau_1) x(t-\tau_2) \dots x(t-\tau_n) d\tau_1 d\tau_2 \dots d\tau_n + \dots \end{aligned} \tag{4}$$

If the time in (4) is discrete, then the series assumes the form

$$\begin{aligned} & y(k) = h_0 + \\ & \sum_{n=0}^{\infty} h_1(n) x(k-n) + \\ & \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} h_2(n_1, n_2) x(k-n_1) x(k-n_2) + \dots + \\ & \sum_{n_1=0}^{\infty} \dots \sum_{n_n=0}^{\infty} h_n(n_1, n_2, \dots, n_n) x(k-n_1) x(k-n_2) \dots x(k-n_n) + \dots \end{aligned} \tag{5}$$

The Volterra Series and its kernels can be described in the time-domain or in the frequency-domain as well, changing from one representation to the other one with the Fourier Transform. The Fourier Transform of the kernels functions yields the transfer functions of the nonlinear system under study. The transform of h_1 gives the linear transfer function, the transform of h_2 gives the quadratic transfer function and in general, the transforms of the higher order kernels give the higher order transfer functions of the system. A schematic representation of a Volterra system can be seen in Fig. 5. The Volterra model can be easily extended to a function depending on two input variables [12] as it is the case of the previously presented Equation (1), that models the double dependence of the current *Ids* on two intrinsic voltages, *Vds* and *Vgs*.

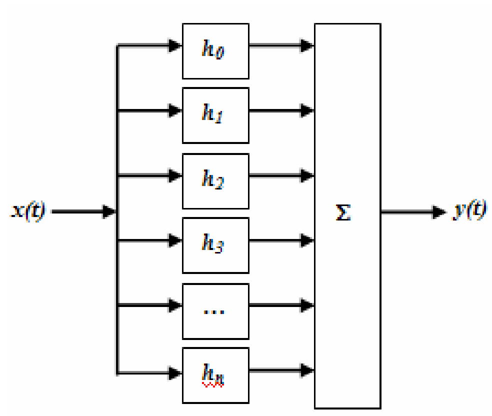

Fig. 5. Schematic representation of a Volterra system

In the Biology field, Wray & Green [13] have outlined a method for extracting the Volterra kernels from the weights and bias values of a time-delayed MLP (Multilayer Perceptron) Neural Network. Based on this idea, there have been several proposals for kernels calculation with different, often non standard, neural networks topologies [14][15][16]. However, all of these approaches use a discrete temporal behavior by means of multi-delayed samples of a unique input variable. In the case of the current/voltages relationships in a transistor, such models are not useful because the multivariable dependence cannot be modeled. Also the fact that they propose non standard neural networks models makes difficult its use. Our proposed model, instead, is more general in the sense that allows representing not only a dependence on one variable, but also a function depending on two or more variables. Another important difference with the mentioned approaches is that we propose the use of a standard MLP Neural Network, which simplifies and speeds its practical implementation.

Concerning the use of laboratory measurements for feeding a Neural Network model of a transistor, in [17] different models and networks topologies are used and compared, to describe the *Ids* nonlinearity using measurements of the current and all its derivatives, which are complex measurements to make and are based, actually, on numerical estimations. Our proposed approach [18] (extended here to a more general case), instead, needs only simple measurements of the input voltages and the output current in the transistor, for the training of the proposed Neural Network model. The current derivatives are found once the training phase has concluded, when the Volterra kernels are calculated using the parameters of the MLP network. The model is explained in the next Section.

4 Novel Neural Network based Volterra model of an electronic device

The topology of the proposed MLP Neural Network model is shown in Fig. 6.

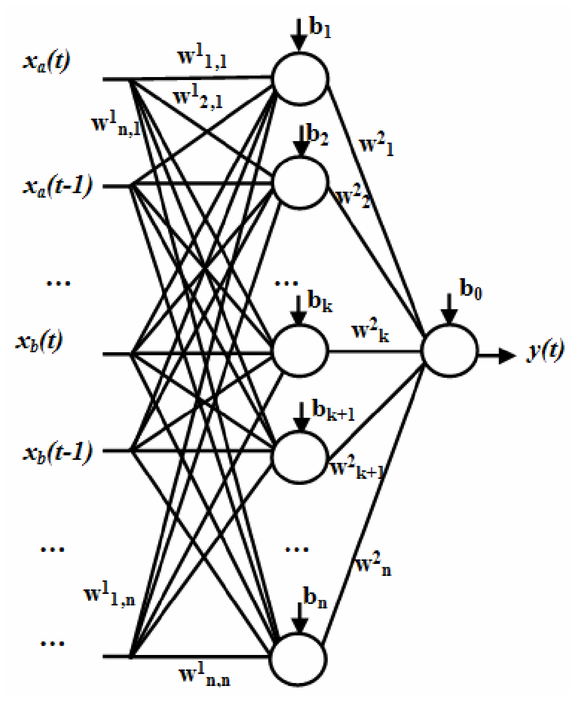

Fig. 6. MLP Neural Network model

The input layer has as inputs the samples of the independent variables, together with their time-delayed values. For the case of study we have chosen , the *Ids(Vgs,Vds)* characteristic in a FET transistor, the model has two input variables, x_a and x_b, which correspond to the two control voltages v*ds* and v*gs*, respectively. In the hidden layer there are, initially, as many hidden neurons as input neurons, with the hyperbolic tangent as activation function. If necessary, the number of neurons in the hidden layer can be changed to improve accuracy. In general, the total number of neurons in each layer can be between *1* and *n*. The hidden neurons receive the sum of the weighted inputs plus a corresponding bias value b_k ($1 \geq k \leq n$) in each neuron.

There is only one output neuron, whose activation function is linear. Therefore, the output of the proposed neural network model is calculated as the sum of the weighted outputs of the hidden neurons plus the output neuron bias, yielding

$$y(t) = b_0 + \sum_{i=1}^{n} w_i^2 \, tanh\left(b_i + \sum_{j=1}^{n} w_{i,j}^1 x_z(t-(j-1)) \right) \quad (6)$$

where, in the case of the neural model of Fig. 6, $z=[a,b]$.

Following the approach in [13], we expand the output of our network model, Equation (6), as a Taylor series around the bias values of the hidden nodes, which yields

$$y(t) = b_0 + \sum_{i=1}^{n} w_i^2 \sum_{m=0}^{\infty} \left[\frac{\left. \frac{\partial^m tanh}{\partial x^m} \right|_{x=b_i}}{m!} \left(\sum_{j=1}^{n} w_{i,j}^1 x_z(t-(j-1)) \right)^m \right] \quad (7)$$

Expanding Equation (7) up to $m = 2$ derivative order and accommodating the terms according to the derivatives, yields Equation (8) which is the final form of the output of our Neural Network model.

$$\begin{aligned} y(t) = {} & b_0 + \sum_{i=1}^{\infty} w_i^2 \, tanh(b_i) + \\ & \left[\sum_{i=1}^{n} w_i^2 w_{i,1}^1 \left(\left. \frac{\partial tanh}{\partial x} \right|_{x=b_i} \right) \right] x_a(t) + \\ & \left[\sum_{i=1}^{n} w_i^2 w_{i,k}^1 \left(\left. \frac{\partial tanh}{\partial x} \right|_{x=b_i} \right) \right] x_b(t) + \\ & \left[\sum_{i=1}^{n} \sum_{j=1}^{n} w_i^2 w_{j,1}^1 w_{j,1}^1 \left(\frac{\left. \frac{\partial^2 tanh}{\partial x^2} \right|_{x=b_i}}{2!} \right) \right] x_a(t)^2 + \\ & \left[\sum_{i=1}^{n} \sum_{j=1}^{n} w_i^2 w_{j,k}^1 w_{j,k}^1 \left(\frac{\left. \frac{\partial^2 tanh}{\partial x^2} \right|_{x=b_i}}{2!} \right) \right] x_b(t)^2 + \\ & \left[\sum_{i=1}^{n} w_i^2 w_{i,1}^1 w_{i,k}^1 \left(\frac{\left. \frac{\partial^2 tanh}{\partial x^2} \right|_{x=b_i}}{2!} \right) \right] x_a(t) x_b(t) + \ldots \end{aligned} \quad (8)$$

Considering in Equation (8) only the terms that contain the variable $x_a(t)$, and supposing that it represents the voltage v*gs*; then supposing also that the variable $x_b(t)$ represents the voltage v*ds*, and that the output of the neural network represents the current *Ids* in a FET, comparing the double Volterra representation of the current in Equation (1) with the output of the Neural Network model, it is easy to recognize the terms between brackets in Equation (8) as the Volterra kernels of a Volterra series expansion for the relationship *Ids(Vgs, Vds).*

Equations (9) to (13) show the general formulas to calculate any order Volterra kernel using the weights and hidden neurons bias values of a neural network model, having into account *n* number of input neurons and *n* number of hidden neurons.

$$h_0 = b_0 + \sum_{i=1}^{n} w_i^2 \, tanh(b_i) \quad (9)$$

$$h_1(k) = \sum_{i=1}^{n} w_i^2 w_{i,k+1}^1 (1 - tanh^2(b_i)) \quad (10)$$

$$h_2(k_1,k_2) = \frac{\sum_{i=1}^{n} w_i^2 w_{i,k_1+1}^1 w_{i,k_2+1}^1 (-2tanh(b_i) + 2tanh^3(b_i))}{2!} \quad (11)$$

$$h_3(k_1,k_2,k_3) = \frac{\sum_{i=1}^{n} w_i^2 w_{i,k_1+1}^1 w_{i,k_2+1}^1 w_{i,k_3+1}^1 (-2 + 8tanh^2(b_i) - 6tanh^4(b_i))}{3!} \quad (12)$$

$$h_n(k_1,k_2,\ldots,k_n) = \frac{\sum_{i=1}^{n} w_i^2 w_{i,k_1+1}^1 w_{i,k_2+1}^1 \ldots w_{i,k_n+1}^1 \left(\left. \frac{\partial^n tanh}{\partial x^n} \right|_{x=b_i} \right)}{n!} \quad (13)$$

Therefore, we have shown here how using only data of the input voltages and output current of a FET transistor to train the proposed Neural Network model, and then operating with its parameters, the Volterra series model for the current and its kernels can be obtained in a very easy and straightforward manner, saving precious time to the electronic designer engineer when modeling a nonlinear device.

In the next Section, the implementation of the above formulas and the creation of a MatLab® function to calculate the Volterra approximation using the Neural Network parameters are explained. Simulation results obtained with the proposed approach are presented in Section 6.

5 Implementation of a function able to calculate the Volterra kernels from the parameters of a MLP model

We have implemented the formulas presented above inside a MatLab® function, that we have named *NN_Volterra*. This function receives as arguments the input/output data that will be used for the training of a standard MLP Neural Network model, and the number of desired neurons for the hidden layer.

The function creates the MLP model according to the parameters passed. It then trains the network until a prefixed accuracy is obtained or a certain number of epochs have passed. Although the Back-Propagation algorithm is generally used for the training of MLP networks, its convergence parameters have to be finely adjusted to get fast convergence. That is why in the simulations in this paper we have used the Levenberg-Marquardt algorithm to train the weights and bias values of the networks, which provides relatively fast training and no adjusting of step and momentum terms are required to obtain convergence. The accuracy desired for the training phase was set to a MSE (Mean Squared Error) equal to $1x10^{-20}$ and the minimum number of training epochs is $1x10^{4}$.

Once the training phase is concluded, the algorithm extracts the weights (WIJ) and bias values (B) matrixes from the network and uses them to calculate the Volterra kernels, which are saved into the matrixes named H1, H2 and H3, that store the first, second and third order kernels, respectively. After that, the Volterra Series model for the nonlinear characteristic of the device under study, including up to the third order kernels, is built. A normalization step performed before the MLP training. The normalization maps the values of the input/output data between the interval [-1;1] which is the domain of the hyperbolic tangent function, used as activation function in the hidden neurons. This allows a better approximation of the nonlinear relationship between the data. After the calculations, a de-normalization step is performed. The source code of the function *NN_Volterra* is presented in the following lines.

```
function [VolterraOut] =
NN_Volterra(InData, OutData, HiddenNeurons)
% NN definition
MLP=newff([InData(:,1)InData(:,size(InData,
2))],[HiddenNeurons,1],{'tansig'
'purelin'});
%-----------------------------------------
% NN TRAINING
% random initial values for the Neural
% Network parameters
MLP.trainParam.epochs = 10000;
MLP.trainParam.goal = 1e-20;
MLP = train(MLP,InData,OutData);
%-----------------------------------------
% Building of the matrix H1 that
% contains the 1st. order kernels
for i = 1:I_N,for j = 1:H_N,
H1(i) = H1(i) + W(j) * WIJ(j,i) * (1-
(tanh(B(j)))^2);end,end;
%-----------------------------------------
% Building of the matrix H2 that
% contains the 2nd. order kernels
for i = 1:I_N,for k = 1:I_N,for j = 1:H_N,
H2(i,k) = H2(i,k) + W(j) * WIJ(j,i) *
WIJ(j,k) * [((-2*tanh(B(j))+
(2*(tanh(B(j))^3)))/2)]; end, end, end;
%-----------------------------------------
% Building of the matrix H3 that
% contains the 3rd. order kernels
for i = 1:I_N,for k = 1:I_N,for j = 1:H_N
if i == k
H3(i,k) = H3(i,k) + W(j) * WIJ(j,i) *
WIJ(j,i) * WIJ(j,k) * [(-
2+8*(tanh(B(j))^2)-6*(tanh(B(j))^4))/6];
else
H3(i,k) = H3(i,k) + 3 * W(j) * WIJ(j,i) *
WIJ(j,i) * WIJ(j,k) * [(-
2+8*(tanh(B(j))^2)-6*(tanh(B(j))^4))/6];
end,end,end,end;
%-----------------------------------------
Volterra_1=h;
for i = 1:I_N,
Volterra_1=Volterra_1+InData(i,:).*H1(i);en
d
%-----------------------------------------
% 1st. order approximation
Volterra_2=Volterra_1;
for i = 1:I_N,for j = 1:I_N,
Volterra_2=Volterra_2+InData(i,:).*InData(j
,:).*H2(i,j);end;end;
%-----------------------------------------
% 2nd. order approximation
Volterra_3=Volterra_2;
for i = 1:I_N,for j = 1:I_N,
Volterra_3=Volterra_3+InData(i,:).*InData(i
,:).*InData(j,:) .* H3(i,j);
end;end;
%-----------------------------------------
% 3rd. order approximation
VolterraOut = Volterra_3;
```

6 Case of Study and Simulation Results

To test our approach and as initial case of study, the data for the training phase was obtained from a cubic Curtice model [5] of a FET transistor, with the following typical parameters values: *A0=0.0625, A1=0.05, A2=0.01, A3=0.001, Vgs = [-1…0] step 0.2, Vds = [0…5] step 0.5,* $\beta = 0$, $\chi=0.3$. To begin the analysis, the data obtained was first used for the training of a simplified version of our proposed model (Fig. 7), with only two input neurons (corresponding to the variables *vgs* and *vds*), two hidden neurons and one output neuron (*Ids*) (MLP_2_2_1).

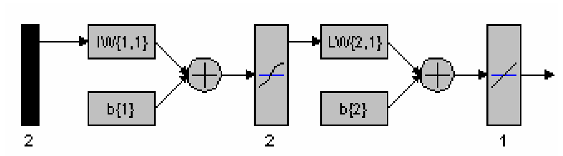

Fig. 7. Initial neural network model MLP_2_2_1 implemented in MatLab®

At the end of the training phase, the Volterra kernels up to the third order were calculated and compared with the values analytically obtained from the Curtice model. The comparison of the results obtained with the Volterra approximation vs. the original data is shown in Fig. 8.

After this initial trial, several possible network topologies were proved, with different number of input neurons and hidden neurons. For example, in Fig. 9 are shown the results obtained with the Neural model MLP_2_10_1, corresponding to 2 input neurons and 10 hidden neurons. As can be seen from the picture, the adding of more neurons to the hidden layer of the model improves accuracy, which can be inferred from the fact that the Mean Square Error is much lower and, graphically, a larger quantity of points is correctly approximated.

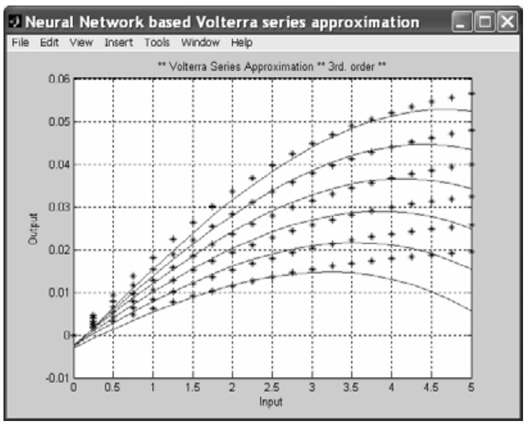

Fig. 8. Current-voltages curves for original data (*) vs. Volterra approximation based on the Neural Network model MLP_2_2_1 (-), MSE = 0.00141235

We have further completed the Neural Network model adding the delayed samples of the input variables, that is to say adding memory to the system. In Fig. 10 we show the results obtained from the model MLP_4_45_1, with 4 inputs and 45 hidden neurons. As can be seen from the results, the adding of memory to this system and of more hidden neurons, further improves the final results obtained, decreasing even more the MSE and approximating almost all the original data points with more precision.

The function we have developed allows this kind of comparison between different models, which can be easily generated in a short period of time, showing the importance of this tool for the analysis of nonlinear electronic devices behavior.

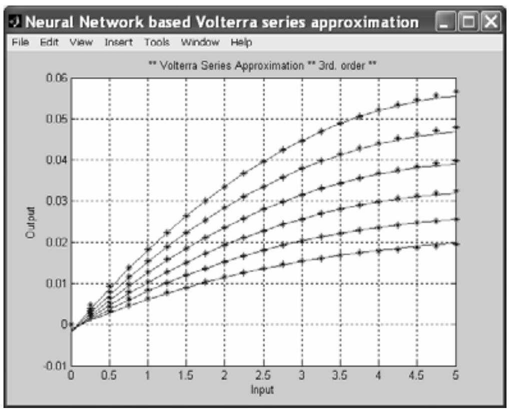

Fig. 9. Current-voltages curves for original data (*) vs. Volterra approximation based on the Neural Network model MLP_2_10_1 (-), MSE = 1.18713x10^{-010}

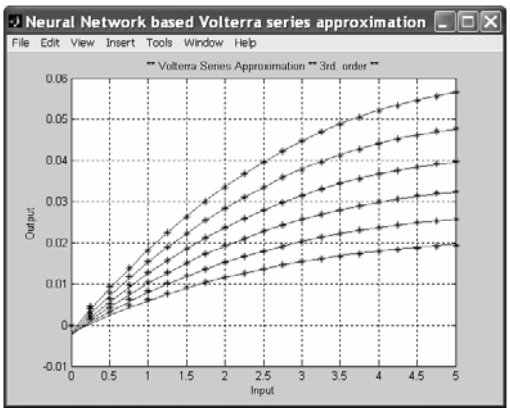

Fig. 10. Current-voltages curves for original data (*) vs. Volterra approximation based on the Neural Network model MLP_4_45_1 (-), MSE = 7.30989x10-^{016}

From the comparison of the presented results we can conclude that the proposed approach is valid and accurate, as an alternative to the classical modeling approach of nonlinearities in electronic devices. Another important point is that the adding of more delayed versions of the inputs variables to the model, and of more hidden neurons, allows obtaining a more accurate approximation of the nonlinearity modeled.

This is a very important piece of information that we will take into account in the next step of our research, which is the automatic Neural Network model topology generation, that is to say, automatically choose of number of hidden neurons and of delayed inputs necessary for an accurate approximation, according to the type of data presented in the training phase.

7 Conclusions

In this work we have presented a Neural Network model that allows the building of a new type of Volterra Black-Box model and the extraction of its Volterra Kernels in the case of a dynamic system with multiple driving voltages, and so able to reproduce a nonlinearity inside an electronic device. We have explained how it is possible to obtain the Volterra series analytical expression for an electronic device nonlinear behavior, which is a difficult task, using parameters of a Neural Network model only trained with input/output measurements, and how to calculate its Volterra Kernels, even in the case of a multivariable function. The model has been validated with a case of study (a FET transistor current-voltages characteristic) and the results are here reported.

Therefore, we can say that the main contribution of our work is the proposal of a novel approach for the generation of the Volterra model for an electronic device, in a simple and straightforward manner, even in the case of a nonlinearity that depends on more than one variable, which allows obtaining a general model, independent of the physical circuit under study.

8 Future Work

Our future work will consist in implementing the proposed Black-Box model inside a commercial electronic circuit simulator for microwave applications, and after that, to try the automatic Neural Network model topology generation, that is to say, automatically choose of number of hidden neurons and of delayed inputs necessary for an accurate approximation, according to the type of data to be modeled.

References

1. Schetzen, M.: The Volterra and Wiener Theories of Nonlinear Systems. Ed. John Wiley & Sons Inc. (1980).
2. Meireles, M.R.G., Almeida, P.E.M., Simoes, M.G.: A comprehensive review for industrial applicability of Artificial Neural Networks. IEEE Transactions on Industrial Electronics, vol. 5, no. 3 (2003) 585-601.
3. Zhang, Q.J., Gupta, K.C., Devabhaktuni, V.K.: Artificial Neural Networks for RF and Microwave Design – From Theory to practice. IEEE Transactions on Microwave Theory and Techniques, vol. 51, no. 4 (2003) 1339-1350.
4. Maas, S.A.: Nonlinear Microwave Circuits. Ed. Artech House (1988).
5. Curtice, W. R., Ettenberg, M.: A Nonlinear GaAs FET Model for Use in the Design of Output Circuits for Power Amplifiers. IEEE Transactions on Microwave Theory and Techniques, vol. 33, no. 12 (1985) 1383-1394.
6. Rough, W.: Nonlinear System Theory. The Volterra/Wiener Approach. Ed. Johns Hopkins University Press (1981) .
7. Weiner, D., Naditch, G.: A scattering variable approach to the Volterra analysis of nonlinear systems, IEEE Transactions on Microwave Theory and Technicques 24 (1976), 422–433.
8. Kashiwagi, H., Harada, H., Rong, L.: Identification of Volterra kernels of nonlinear systems by separating overlapped kernel slices. Conference of the Society of Instrument and Control Engineers of Japan, 2 (2002) 707-712.
9. Bauer, A., Schwarz, W.: Circuit analysis and optimization with automatically derived Volterra kernels. IEEE International Symposium on Circuits and Systems, (2000) I-491 - I-494.
10. Boyd, D., Tang, Y.S., Chua, L.O.: Measuring Volterra Kernels. IEEE Transactions on Circuits and Systems, vol. 30, no. 8 (1983) 571-577.
11. Boyd, S., Chua, L.O, Desder, C.A.: Analytical Foundations of Volterra Series. IMA Journal of Mathematical Control & Information, 1 (1984) 243-282.
12. Pedro, J.C., Madaleno, J.C., Garcia, J.A.: Theoretical Basis for the Extraction of Mildly Nonlinear Behavioral Models. International Journal of RF and Microwave Computer-Aided Engineering, vol. 13, no. 1 (2003) 40-53.
13. Wray, J., Green, G.G.R.: Calculation of the Volterra kernels of non-linear dynamic systems using an artificial neural network. Biological Cybernetics, 71 (1994) 187-195.
14. Soloway, D.I., Bialasiewicz, J.T.: Neural Networks modeling of nonlinear systems based on Volterra series extension of a linear model. IEEE International Symposium on Intelligent Control, 1 (1992) 7-12.
15. Marmarelis, V.Z., Zhao, X.: Volterra models and Three Layers Perceptron. IEEE Transactions on Neural Networks, vol. 8, no. 6 (1997) 1421-1433.
16. Hakim, N.Z., Kaufman, J.J., Cerf, G., Meadows, H.E.: Volterra characterization of

Neural Networks. Signals, Systems and Computers, 2 (1991) 1128-1132.

17. Garcia, J.A., Tazon Puente, A., Mediavilla Sanchez, A., Santamaria, I., Lazaro, M., Pantaleon, C.J., Pedro, J.C.: Modeling MESFETs and HEMTs Intermodulation Distortion Behavior Using a Generalized Radial Basis Function Network. International Journal on RF and Microwave Computer-Aided Design, Special number on Neural Networks, 9 (1999) 261-276.
18. Stegmayer, G., Pirola, M., Orengo, G., Chiotti, O.: Towards a Volterra series representation from a Neural Network model. WSEAS Transactions on Systems, vol. 3, no. 2 (2004) 432-437.