Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 121 – 140
© TÜBİTAK
doi:10.3906/elk-1304-248

Behavior learning of a memristor-based chaotic circuit by extreme learning machines

Ayşegül UÇAR*, **Emrehan YAVŞAN**
Department of Mechatronics Engineering, Faculty of Engineering, Fırat University, Elazığ, Turkey

Received: 28.04.2013 • **Accepted/Published Online:** 11.10.2013 • **Final Version:** 01.01.2016

Abstract: As the behavior of a chaotic Chua's circuit is nonstationary and inherently noisy, it is regarded as one of the most challenging applications. One of the fundamental problems in the prediction of the behavior of a chaotic Chua's circuit is to model the circuit with high accuracy. The current paper presents a novel method based on multiple extreme learning machine (ELM) models to learn the chaotic behavior of the four elements canonical Chua's circuit containing a memristor instead of a nonlinear resistor only by using the state variables as the input. In the proposed method four ELM models are used to estimate the state variables of the circuit. ELMs are first trained by using the data spoilt by noise obtained from MATLAB models of a memristor and Chua's circuit. A multistep-ahead prediction is then carried out by the trained ELMs in the autonomous mode. All attractors of the circuit are finally reconstructed by the outputs of the models. The results of the four ELMs are compared to those of multiple linear regressors (MLRs) and support vector machines (SVMs) in terms of scatter plots, power spectral density, training time, prediction time, and some statistical error measures. Extensive numerical simulations results show that the proposed system exhibits a highly accurate multistep iterated prediction consisting of 1104 steps of the chaotic circuit. Consequently, the proposed model can be considered a promising and powerful tool for modeling and predicting the behavior of Chua's circuit with excellent performance, reducing training time, testing time, and practically realization probability.

Key words: Extreme learning machines, memristor-based chaotic circuit, multistep-ahead prediction

1. Introduction

In 1971 Leon Chua proposed a fourth basic passive element named memristor (the contraction of memory resistor) in addition to the three fundamental passive elements (resistor, capacitor, and inductor) in electronic circuit theory [1]. This element has a functional relation between charge and flux. This function may be either a charge-controlled or a flux-controlled one. The memristor's useful relation was realized 37 years later by Stanley Williams and his group at HP Labs [2]. The memristor was produced on a nanometer scale in solid-state with a two-terminal device form. Following this, MATLAB and SPICE models were presented for simulation [3–5] and more detailed analyses were presented [6,7]. Many more memristor models were then presented by means of available data relating to the dynamic behavior of many different devices to obtain models behaving like them [8,9]. They used a lot of devices, macromodels, and emulators with circuit components or MATLAB and SPICE tools [8–10]. In order to enhance functionality, obtain a more compact structure, and reduce power consumption, a lot of studies have been conducted on memristors for many analogue and digital circuits, including cellular neural networks (CNN) [11], recurrent neural networks (RNN) [12], Schmitt triggers, difference amplifiers [13,14],

*Correspondence: agulucar@firat.edu.tr

ultrawide band receivers [15], adaptive filters [16] and oscillators [17], field programmable nanowire interconnect (FPNI) design [18], field programmable gate array (FPGA) design [19,20], and CMOS design [21,22].

A memristor can lead to chaotic behavior when connected to a power source or when the Chua diode is replaced with a memristor [3,4,23,24]. In [24] several nonlinear oscillators utilized monotone increasing piecewise-linear memristors with passive characteristics, and in [2] a memristor was used instead of the Chua diode in Chua's circuit. In [3,4] high frequency chaotic circuits were proposed based on Chua's canonical circuit with active nonlinearity. In [7] an autonomous circuit that uses only a linear passive inductor, a linear passive capacitor, and a memristor was proposed. The chaotic circuits like in [3,4,7] are used for applications of secure combination and image encryption [25]. Although there are many other variations of a memristor-based Chua's circuit in the literature used to obtain different chaotic behaviors, the current study aims to learn the behavior of the chaotic circuits in [3,4] due to the exhibition of chaotic behavior by a smaller number of components than the others [26]. However, the method can also be extended to the other chaotic circuits.

Chaotic behavior in deterministic dynamical systems is an intrinsically nonlinear phenomenon. Since the behavior is extremely dependent on system inputs and small noises in the input variables will grow exponentially quickly, the learning of chaotic behavior is a highly complicated problem to comprehend. If a chaotic behavior is expressed as a time series, the learning problem of chaotic behavior changes into a multistep-ahead prediction problem of the time series. In order to analyze a time series prediction problem, statistical methods such as linear regression and autoregressive integrated moving average models (ARIMA) are used [27]. They are applied for predicting and modeling the time series of such several soft computing techniques as single-layer feed-forward neural networks (SLFNNs) [28], radial basis functional neural networks (RBFNNs) [29], support vector machines (SVMs) [30], Elman recurrent neural networks (ERNNs) [31], and extreme learning machines (ELMs) [32]. Of all the techniques, the ELM is the most attractive model for the prediction problems in terms of fast learning properties, excellent performance, ease of implementation, and minimal human intervention.

The ELM is a kind of SLFNN. The SLFNNs are capable of approximating a nonlinear function by nonlinear mappings by using input samples [33]. The weights and biases parameters of SLFNNs are usually iteratively adjusted by gradient-based learning algorithms. The SLFNNs are generally very slow due to improper learning steps or may easily converge to local minima; they need a number of iterative learning steps in order to obtain a better learning performance [33]. In order to counter these disadvantages of SLFNNs for the prediction of chaotic time series in the current study, the ELM proposed by Huang et al. [34–36] is considered. In the ELM the weights of hidden nodes and biases are randomly chosen and output weights are analytically determined [37]. The ELM reaches a good generalization performance extremely fast. The fast training property of ELM makes ELM digitally designable by using FPGA.

A neural state space model and a recurrent least square SVM model behaving like the double scroll attractor in Chua's circuit were presented in [38,39], respectively. To alleviate high structural complexity and slow training speed in the SLFNN model of Chua's circuit in [40] a wavelet decomposition method was employed together with SLFNNs. In [41] a wavelet decomposition method and SLFNNs were proposed to classify multiscroll chaotic attractors of Chua's circuit. A memristor including Chua's circuit was modeled by SVMs to generate a framework for secure communication in [42]. All studies need to choose network inputs such as appropriate embedding dimension, time delay, or wavelet coefficients. In [43] Chua's circuit was modeled by the least square SVMs. The state variables were taken as inputs to SVMs without selecting any input parameter. However, the prediction outputs of these models are not considered as the next input to repeat.

Hence, the multistep-ahead prediction error increases when the method in [43] is used.

In the current study a new multiple ELM model is proposed as an efficient regression method to learn the behavior of the memristor-based chaotic circuit in [3,4] and then to carry out the multistep-ahead prediction of the circuit aiming at improving the efficiency and effectiveness of the prediction accuracy. First, the data in relation to the four variables obtained from the MATLAB model of the memristor-based chaotic circuit in [3,4] are spoilt by noise to obtain a high modeling accuracy. Without selecting the embedding dimension and the time delay, an input vector consisting of only the four state variables is used to train the ELMs in a simple and fast way. Therefore, the one-step-ahead predictions of the chaotic time series in relation to current, voltages, and flux of the memristor are carried out by the multiple ELMs. The long-term multistep-ahead prediction is then applied by feeding back the outputs of the ELMs to inputs. Finally, the chaotic time series are regenerated and all chaotic attractors of the circuit are reconstructed. The obtained results are compared to those of Multiple Linear Regression (MLR) and SVMs. Comparative extensive results including a scatter plot, power spectral density, and some error measures are illustrated to present the effectiveness of the ELM.

The rest of this paper is organized as follows: in Section 2 the basic architecture of the ELM is shortly introduced; the equations of the memristor-based chaotic circuit are presented in Section 3; in Section 4 the learning procedure for prediction is emphasized; the simulation results are illustrated to demonstrate the performance of the ELMs for learning of the behavior of the chaotic circuit in Section 5; Section 6 concludes this paper.

2. Extreme learning machine

The ELM proposed by Huang et al. [34–36] is an SLFNN with H hidden neurons and common activation functions. The architecture of the ELM is shown in Figure 1.

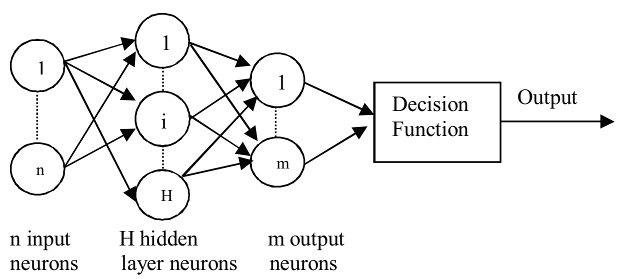

Figure 1. The architecture of ELM.

Given a set of N arbitrary distinct samples (x_i, y_i) where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ is an n-dimensional input vector and $y_i = [y_{i1}, y_{i2}, \ldots, y_{im}]^T \in R^m$ is an m-dimensional output vector, the input–output relation of an ELM with H hidden nodes and activation function s(x) is written in the following form:

$$\sum_{i=1}^{H} F_i s(x_j) = \sum_{i=1}^{H} F_i s\left(w_i.x_j + b_i\right) = \hat{y}_j,\ j = 1, \ldots, N, \tag{1}$$

where $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T \in R^n$ is the weight vector connecting the inputs nodes to an $i-th$ hidden node, $F_i = [F_{i1}, F_{i2}, \ldots, F_{im}]^T \in R^m$ is the weight vector connecting an $i-th$ hidden node to output nodes, and b_i is a threshold relating to an $i-th$ hidden node.

The ELM formulation can approximate N samples with zero error by satisfying

$$\sum_{j=1}^{N} \|\hat{y}_j - y_j\| = 0, \tag{2}$$

or

$$\sum_{i=1}^{H} F_i s\left(w_i.x_j + b_i\right) = y_j, \;\; j \in \{1, 2, \ldots, N\} \tag{3}$$

The output of the ELM can be written as follows:

$$SF = Y, \tag{4}$$

$$S\left(w_1, \ldots, w_H, b_1, \ldots, b_H, x_1, \ldots, x_N\right) = \begin{bmatrix} s\left(w_1.x_1 + b_1\right) & \ldots & s\left(w_H.x_1 + b_H\right) \\ \vdots & \ldots & \vdots \\ s\left(w_1.x_N + b_1\right) & \ldots & s\left(w_H.x_N + b_H\right) \end{bmatrix}_{NxH} \tag{5}$$

$$F_i = \left[F_{i1}, F_{i2}, \ldots, F_{im}\right]^T_{Hxm} \text{ and } Y_i = \left[y_1, y_2, \ldots, y_N\right]^T_{Nxm},$$

where S is the hidden layer output matrix of ELM and s is the infinitely differentiable activation function, the number of hidden nodes is chosen as $H \ll$ N.

Here the learning problem of $\hat{w}_i, \hat{b}_i, \hat{F}\,(i = 1, \ldots, H)$ parameters of an SLFNN is equal to the below equation to solve the primal optimization problem:

$$\left\|S\left(\hat{w}_1, ..., \hat{w}_j, \hat{b}_1, ...\hat{b}_j\right)\hat{F} - Y\right\| = \min_{w_i, b_i, F} \left\|S\left(w_1, ..., w_M, b_1, ..., b_M\right)F - Y\right\|. \tag{6}$$

Generally the objective function for the optimization problem is expressed as

$$E = \sum_{j=1}^{N} \left(\sum_{i=1}^{H} F_i s\left(w_i.x_j + b_i\right) - y_j\right)^2. \tag{7}$$

The parameters are optimized by calculating the negative gradients of the objective function with respect to w_i, b_i, F_i:

$$w_l = w_{l-1} - \tau \frac{\partial E}{\partial w} \tag{8}$$

The accuracy and learning speed of the gradient based method particularly depend on the learning rate τ. While a small learning rate provides very slow convergence, a larger learning rate exhibits the bad local minima effect. The ELMs are rendered free from these limitations by using the minimum on account of the fact that the minimum norm least-square solution is used. The weight and bias values of ELMs are randomly assigned unlike the SLFNNs. Hence, the parameters of ELM represented by the linear system in Eq. (4) are learned solving a least-square formulation in Eq. (6). If S is a nonsquare matrix for $H \ll N$, a norm least-square solution is obtained as $\hat{F} = S^{*}Y = (S^T S)^{-1} S^T Y$, where S^{*} is the Moore–Penrose generalized inverse of matrix S. The smallest training error is achieved by using the following equation:

$$\left\|S\hat{F} - Y\right\| = \|SS^{*}Y - Y\| = \min_F \|SF - Y\|. \tag{9}$$

Therefore, an extremely fast and better generalization performance is obtained than those of traditional SLFNNs for hidden layers with infinitely differentiable activation functions. The final ELM achieves not only the smallest training error but also the smallest generalization error thanks to the smallest norm of output weights similar to SVMs.

3. The memristor-based Chua's circuit

Muthuswamy and Kokote proposed a four-element Chua's chaotic circuit with a memristor that replaces the nonlinear resistor in 2009 [3,4]. Figure 2a shows the four-element chaotic Chua's circuit including the memristor. Figure 2b shows a representation by using only the basic circuit elements of the Chua's circuit. In the equation - k stands for the effect of op-amp A_1. The equations for the circuit are described by a set of ordinary differential equations:

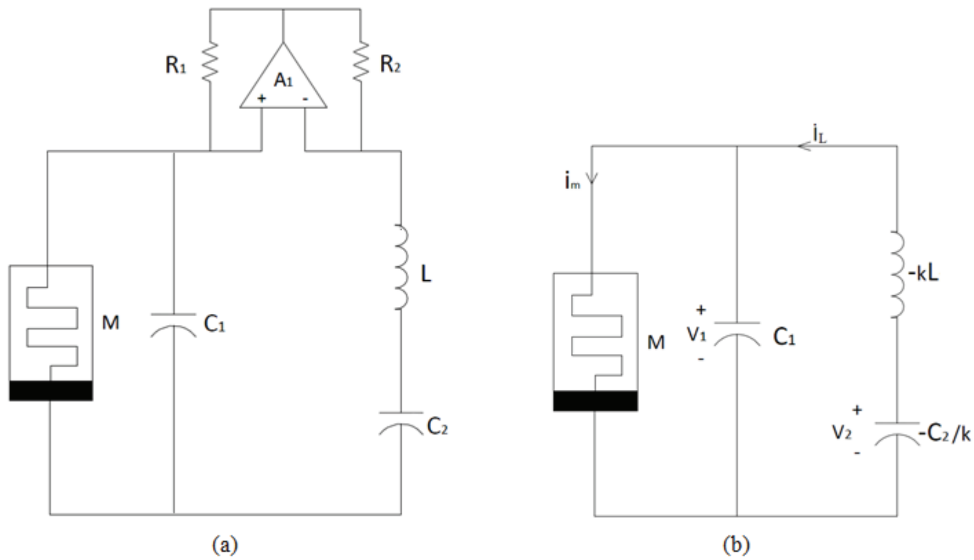

Figure 2. The four-element Chua's circuit: (a) with memristor; (b) with only the basic circuit elements, - k represents the effect of op - amp A_1 in (a) [3].

$$\begin{cases} \frac{dV_1}{dt} = \frac{i_L - W(\emptyset)V_1}{C_1}, \\ \frac{dV_2}{dt} = \frac{k}{C_2} i_L, \\ \frac{di_L}{dt} = \frac{V_1 - V_2}{kL}, \\ \frac{d\emptyset}{dt} = V_1, \end{cases} \tag{10}$$

where i_L is the current flowing in the inductor L and V_1 and V_2 are the voltages across the capacitors C_1 and C_2. The current of the flux-controlled memristor M, i_m, is determined by

$$\begin{cases} i_m(t) = W(\varphi(t))\, V_1(t)\ , \\ W(\varphi) = \frac{dq_m(\varphi)}{d\varphi}, \end{cases} \tag{11}$$

where $\varphi(t)$ is the magnetic flux between the memristor terminals and $W(\varphi(t))$ and it is termed the memductance.

Let $x_1 = V_1, x_2 = V_2, x_3 = i_L, x_4 = \varphi$, then

$$\begin{cases} \dot{x}_1 = \alpha_1 x_3 - \alpha_1 W(x_4)\, x_1, \\ \dot{x}_2 = \alpha_2 x_3, \\ \dot{x}_3 = \alpha_3 x_1 - \alpha_3 x_2, \\ \dot{x}_4 = x_1, \end{cases} \tag{12}$$

where $\alpha_1 = \frac{1}{C_1}, \alpha_2 = \frac{k}{C_2}, \alpha_3 = \frac{1}{kL}$ and $W(x_4)$, the incremental memductance, has the piecewise-linear characteristic in Figure 3.

$$W(x_4) = \begin{cases} 43.25 * 10^{-4} & x_4 \le -15 * 10^{-5} \\ 9.33x_4 - 9.67 * 10^{-4} & -15 * 10^{-5} < x_4 \le -5 * 10^{-5} \\ -5.005 * 10^{-4} & -5 * 10^{-5} < x_4 \le 5 * 10^{-5} \\ -9.33x_4 - 9.67 * 10^{-4} & 5 * 10^{-5} < x_4 \le 15 * 10^{-5} \\ 43.25 * 10^{-4} & \text{the other} \end{cases} \tag{13}$$

The charge is calculated by the indefinite integration

$$Q(x_4) = \int W(x_4)\, dx_4 = \begin{cases} 43.25 * 10^{-4} x_4 & x_4 \le -15 * 10^{-5} \\ -6.888 * 10^{-7} - 9.67*10^{-4} x_4 - 4.665 x_4^2 & -15 * 10^{-5} < x_4 \le -5 * 10^{-5} \\ -6.7717 * 10^{-7} - 5.005 * 10^{-4} x_4 & -5 * 10^{-5} < x_4 \le 5 * 10^{-5} \\ -6.6551 * 10^{-7} - 9.67*10^{-4} x_4 + 4.665 x_4^2 & 5 * 10^{-5} < x_4 \le 15 * 10^{-5} \\ -1.3543 * 10^{-6} + 43.25 * 10^{-4} x_4 & \text{the other} \end{cases} \tag{14}$$

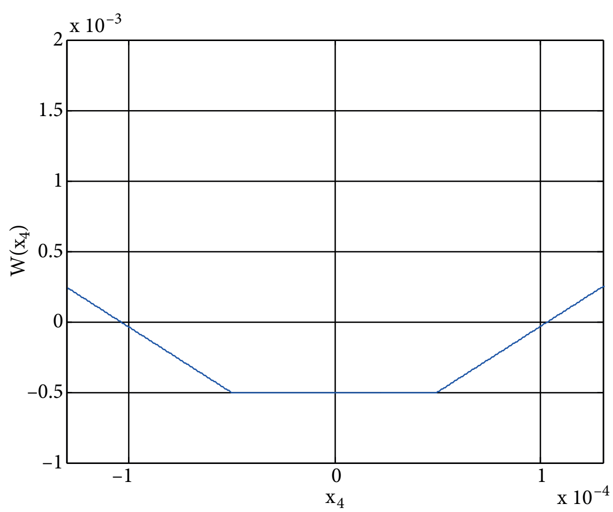

Figure 3. The W(x_4) - x_4 characteristic of the memristor.

In this study the circuit parameters were chosen as $\alpha_1 = \frac{1}{33x10^{-9}}$, $\alpha_2 = \frac{8.33}{100x10^{-9}}$, and $\alpha_3 = \frac{1}{83.3x10^{-3}}$ as proposed in [3,4] in order to exhibit double scroll chaotic behavior. When the value of α_1 is decreased or the value of α_2 is increased, the equilibrium point, period-doubling sequences, and spiral and double scroll chaotic attractor behaviors can be observed. Each of them can be employed for modeling some dynamics in nature [26].

The MATLAB model of memristor in Eq. (13) in this paper is firstly employed as an m-function [3]:

```
function v = W(z)
if z(1) < = – 1.5e – 4
v = 43.25e – 4;
elseif z(1) > – 1.5e – 4 & & z(1) < = – 0.5e – 4
v = – 9.33 * z(1) – 9.67e – 4;
elseif z(1) > – 0.5e – 4 & & z(1) < 0.5e – 4
v = – 5.005e – 4;
elseif z (1) > = 0.5e – 4 & & z(1) < 1.5e – 4
v = 9.33 * z(1) – 9.67e – 4;
else
v = 43.25e – 4;
end
end
```

The memristor-based Chua's circuit is then modeled by MATLAB. The m-file of the circuit is constructed by two m-functions: an m-function defining the state equations and an m-function solving the state equations. Fourth-order Runge-Kutta algorithm as well as the differential equation solver *ode*45 of MATLAB are used. The scalar relative error tolerance value and absolute error tolerance value are chosen as 0.0001. Initial values are $\left[\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array}\right]\Big|_{t=0} = \left[\begin{array}{cccc} 0.1 & 0 & 0 & 0 \end{array}\right]$. The simulation time is chosen as 0.01 second. Some basic program lines are given as follows [3]:

```
state - equation = inline( ′ [p(2) ; (p(3) – W(p) * p(2)) / 33e – 9 ; (p(2) – p(4)) /(8.33 * 10e – 3) ; p(3)
* (8.33 / 100e – 9)] ′ , ′ t ′ , ′ p ′ );
options = odeset ( ′ RelTol ′ ,1e – 4, ′ AbsTol ′ ,1e – 4);
[t, pa] = ode45(state - equation, [0 10e – 3], [0,0.1,0,0],options);
```

4. Multistep-ahead prediction and reconstruction of chaotic attractors

A set of nonlinear ordinary differential equations is defined as follows:

$$\left\{ \begin{array}{l} \mathbf{x}_t = f\left(\mathbf{x}_{t-1}\right), \\ \mathbf{y}_t = g\left(\mathbf{x}_t\right), \end{array} \right. \tag{15}$$

where $\mathbf{x}_t \in R^4$ is the state variables of the circuit and $\mathbf{y}_t \in R^1$ is the time series relating to the state variable to be predicted. The sampling period is expressed as a suitable one so as to provide easiness for presentation.

In the current paper the four state variables are individually estimated by the four ELMs. The ELMs are trained by using only four inputs without selecting the embedding dimension and the time delay similar to [43] and contrary to the conventional time series prediction methods. The output of each ELM is accepted as a next value of state variables different from [43]. A multistep-ahead prediction is carried out in an autonomous mode by feeding back the outputs of ELMs as input:

$$\begin{cases} \mathbf{y}_{i,t+1} = \hat{g}_i\left(\mathbf{x}_t\right), \\ \mathbf{y}_{i,t+2} = \hat{g}_i\left(\mathbf{y}_{t+1}\right), \quad i = 1,2,3,4 \\ \mathbf{y}_{i,t+3} = \hat{g}_i\left(\mathbf{y}_{t+2}\right), \\ \vdots \end{cases} \tag{16}$$

where $\hat{g}_i\left(.\right)$ represents the outputs of evaluated ELMs.

The proposed method, based on multiple ELM models, is iteratively generated in the following steps:

Step 1. Collect the data set in relation to all state variables by using the model of memristor-based Chua's circuit in MATLAB for the time interval between 0 and p_3.

Step 2. Generate the training, testing, and validation sets by using the portions of the data set corresponding to the time intervals of [0; p], [p_1; p_2], and [p_2; p_3], respectively.

Step 3. Train ELM models in relation to each state variable by using the training data spoilt by noise and determine optimal ELM models on the validation set.

Step 4. Apply the optimal ELM models to the testing set feeding back the outputs of ELMs to the inputs.

Step 5. Reobtain the chaotic time series of each state variable and reconstruct the response of the circuit as input and output samples.

The framework of the proposed method is illustrated in Figure 4. The accuracy of the whole system depends on the performance of each ELM model. A higher modeling error for each ELM model causes a higher accumulation error in the multistep iterated prediction. The reason for applying multiple ELM models in this paper is to obtain a more stable and fast prediction performance thanks to the high modeling performances of ELM models trained independently by using the same training data. The method is simple and easy to apply since the embedding dimension and time delay parameter of the chaotic time series in relation to each state variable are not required to be determined.

5. Numerical simulation results and discussion

In this study the four ELM models are used to learn the behavior of a memristor-based chaotic circuit in Figure 3. The ELMs are first trained for estimating x_1, x_2, x_3, and x_4. The future values of the behavior of the chaotic circuit are predicted. This is known as a one-step-ahead prediction. Next, the iterated multistep-ahead prediction is carried out by using the previously trained ELMs with the best network architectures. The outputs of ELM models are given as next inputs of ELM models in this case. Finally, the chaotic time series are reobtained and the chaotic attractors are reconstructed.

All experiments were run by using MATLAB on a personal notebook computer with 2.4-GHz Intel(R) Core(TM) 2 Duo processor, 3GB memory, and Windows 7 operation system. For ELM the implementation method described in [34–36] was employed. Regarding SVM, the LIBSVM implementation method described in [44] was employed. In the simulations a time series consisting of 5521 data was obtained for each state variable. Sixty percent of all data was used for the training set. Half of the remaining 40% was used for testing and validation. The training set corresponds to 3321 data for 0.0059 s. The validation set corresponds to 1104 data for the time interval between 0.0059 s and 0.0080 s. The testing set corresponds to 1104 data for the time

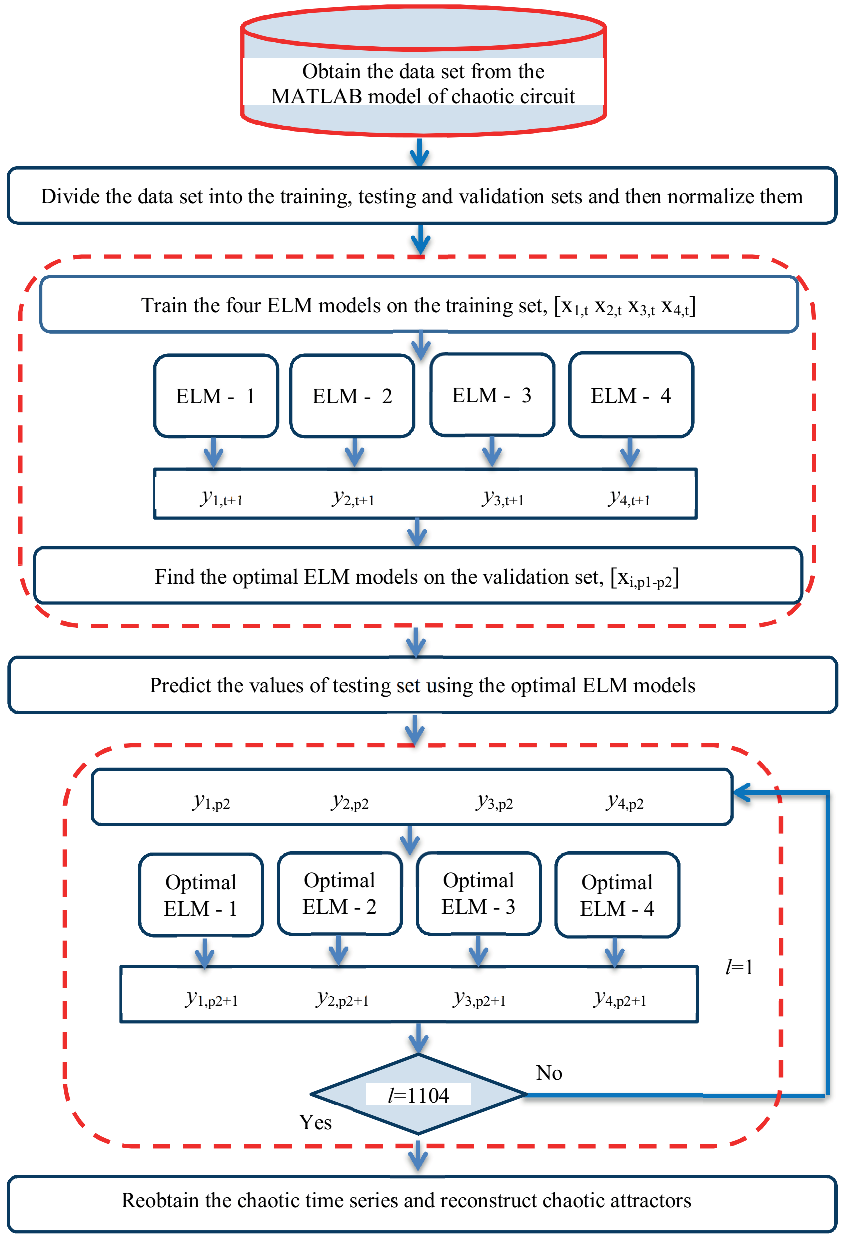

Figure 4. The architecture of the proposed multiple-ELM prediction method.

interval between 0.0080 s and 0.01 s. The training set was spoilt by Gaussian noise with a standard deviation of 0.1. The testing set and validation set were not spoilt by noise.

The performance of ELM models depends only on an appropriate setting of the number of hidden neurons, while the selection of three parameters, the regularization parameter C, the precision parameter ε, and the kernel parameter γ, is important to the performance of SVMs [33,34,44,45]. There are no general rules for parameter setting of the SVM. The selection is usually based on trial and error, also called cross validation method, or user's expertise. However, the trial-and-error method is very time consuming. In this paper the radial basis function kernel was used for SVMs and the width parameter of the kernel was set as 0.6, according to $\gamma^m \sim (0.1, 0.5)$ as in [46] for speeding up the parameter selection stage, where m is the number of input variables. The regularization parameter C and the precision parameter ε were optimized for all combinations of $C = \left[\begin{array}{ccccc} 2^{10} & 2^{9} & 2^{8} & \ldots & 2^{-2} \end{array}\right]$ and $\varepsilon = \left[\begin{array}{ccccc} 2^{-1} & 2^{-2} & 2^{-3} & \ldots & 2^{-7} \end{array}\right]$ [44]. In order to find the best network architecture of ELMs, it was easily changed through increasing gradually the number of hidden neurons from 1 to 50 in an interval of 1 because of the fast training and testing speed of ELM in this paper. All data were normalized in [–1 1] range as proposed in [34–36]. ELMs with sigmoid activation function were used. RMSE values in training, testing, and validation of the four ELMs are given in Figure 4. The optimal number of hidden neurons of ELMs and the optimal parameters $(C,\ \gamma,\ \varepsilon)$ of SVMs were determined by taking their validation performances into consideration. In order to further evaluate the obtained results the following performance statistics such as root mean-squared errors (RMSEs), mean absolute errors (MAE), and correlation coefficient (R) were used:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2} \tag{17}$$

$$MAE = \frac{\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|_i}{N} \tag{18}$$

$$R = \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2}\cdot\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}}, \tag{19}$$

where y_i is the actual value, $\bar{y}$ is the mean actual value, $\hat{y}_i$ is the predicted value, $\bar{\hat{y}}$ is the mean predicted value, and N is the total data number.

MAE is the average of the obsolete difference of the prediction value from the actual values. RMSE is the square root of the mean square error. MAE and RMSE are highly correlated, but significant considerable errors have a large effect on RMSE because of the square of error. In this study we considered both RMSE and MAE in order to gain some insight into the relative distribution of the error. The correlation coefficient measures how well the predicted values are consistent with the actual values. In order to determine the best model, the values converging to unity of R and the small values of RMSE and MAE are searched by the special validation data set.

All simulations were run 10 times. The average results are listed in the tables. Although the ELM usually gives better results for large numbers of hidden neurons [34–36], this was not observed in the prediction of time series in this paper. A considerable number of hidden neurons lead to significant errors for prediction. However, a higher number of neurons provides healthier results on the training and validation sets. The results estimating x_3 and x_4 in relation to ELMs are presented in Tables 1 and 2. The results relating to the other variables can be similarly interpreted and observed from Figure 5. In these tables all of the results that are up to the lowest validation error for a short and comprehensible presentation are given and the others are dismissed. Table 1 lists the first 26 results relating to the ELM estimating x_3. The biggest value of R and the lowest values of RMSE and MAE relating to ELM estimating x_3 are 0.9998, 0.0141, and 0.0112 in the validation stage, respectively. In Table 2, the first 19 results relating to the ELM estimation of x_4 are presented. The biggest value R and the lowest values of RMSE and MAE relating to ELM estimation of x_4 are 0.9999, 0.0132, and 0.0095, respectively. As can be seen from Figure 5, the best number of hidden neurons for the ELM models in relation to x_1, x_2,

x_3, and x_4 are 28, 27, 26, and 19 in the validation stage, respectively. The reconstructed chaotic time series by using the best ELM architectures are illustrated in Figures 6 and 7. The time interval between 0.0080 s and 0.01 s presents the 1104-step prediction results iterating ELM outputs to their inputs (Figures 6 and 7). The others are the results relating to training and validation, respectively. It can be observed from Figures 6 and 7 that the predicted values by the proposed multiple ELM models are very close to the actual values in the MATLAB model of the memristor-based Chua's circuit, including the testing time interval.

Table 1. The RMSE, MAE, and R statistics of the ELM model estimating x_3.

H	Training Stage			Validation Stage			Testing Stage		
	RMSE	MAE	R	RMSE	MAE	R	RMSE	MAE	R
1	0.5649	0.4946	-0.4304	0.5964	0.5321	-0.4345	0.5231	0.4618	-0.4424
2	0.4425	0.3776	0.6315	0.4536	0.3931	0.7048	0.4423	0.3710	0.5696
3	0.1843	0.1469	0.9449	0.2136	0.1755	0.9559	0.2096	0.1715	0.9263
4	0.2743	0.2249	0.8732	0.2901	0.2389	0.8830	0.2706	0.2168	0.8599
5	0.0802	0.0596	0.9898	0.1039	0.0711	0.9851	0.1031	0.0700	0.9816
6	0.0582	0.0456	0.9946	0.0812	0.0561	0.9921	0.0713	0.0460	0.9917
7	0.0535	0.0428	0.9955	0.0468	0.0362	0.9972	0.0427	0.0297	0.9969
8	0.0513	0.0414	0.9958	0.0531	0.0390	0.9974	0.0490	0.0346	0.9962
9	0.0666	0.0525	0.9930	0.0737	0.0595	0.9930	0.0656	0.0506	0.9928
10	0.0478	0.0385	0.9964	0.0251	0.0191	0.9991	0.0242	0.0184	0.9989
11	0.0464	0.0376	0.9966	0.0242	0.0185	0.9992	0.0235	0.0183	0.9990
12	0.0480	0.0387	0.9964	0.0335	0.0248	0.9985	0.0291	0.0229	0.9985
13	0.0439	0.0357	0.9970	0.0211	0.0166	0.9996	0.0183	0.0143	0.9995
14	0.0434	0.0353	0.9970	0.0223	0.0174	0.9993	0.0183	0.0139	0.9994
15	0.0425	0.0346	0.9971	0.0152	0.0116	0.9997	0.0133	0.0112	0.9997
16	0.0429	0.0348	0.9971	0.0170	0.0133	0.9996	0.0138	0.0110	0.9997
17	0.0439	0.0356	0.9970	0.0208	0.0168	0.9994	0.0203	0.0155	0.9993
18	0.0434	0.0353	0.9970	0.0184	0.0151	0.9996	0.0165	0.0138	0.9995
19	0.0429	0.0348	0.9971	0.0178	0.0145	0.9996	0.0172	0.0139	0.9995
20	0.0426	0.0346	0.9971	0.0154	0.0122	0.9997	0.0142	0.0118	0.9996
21	0.0428	0.0348	0.9971	0.0152	0.0120	0.9997	0.0141	0.0118	0.9997
22	0.0429	0.0348	0.9971	0.0158	0.0130	0.9997	0.0153	0.0130	0.9996
23	0.0424	0.0344	0.9972	0.0156	0.0125	0.9997	0.0132	0.0108	0.9997
24	0.0423	0.0344	0.9972	0.0185	0.0143	0.9995	0.0134	0.0107	0.9997
26	0.0422	0.0344	0.9972	0.0141	0.0112	0.9998	0.0129	0.0105	0.9997

The reconstructed attractors by ELMs are perfect since the ELM models are trained and validated by the data in the training and validation stages, respectively. The success of the proposed method should be proven especially for the prediction stage. Hence, Figures 8 and 9 illustrate all of the reconstructed attractors by ELMs by using only the prediction stage and those of the original circuit. It is observed that all reconstructed attractors by the outputs of ELM regressors are similar to the original ones.

Table 2. The RMSE, MAE, and R statistics of the ELM model estimating x_4.

H	Training Stage			Validation Stage			Testing Stage		
	RMSE	MAE	R	RMSE	MAE	R	RMSE	MAE	R
1	0.7479	0.6923	-0.9209	0.8518	0.7824	-0.9575	0.4947	0.4431	-0.9464
2	0.4192	0.3520	0.7853	0.5137	0.4209	0.8318	0.4619	0.4027	0.3440
3	0.1188	0.0954	0.9844	0.1709	0.1271	0.9776	0.0496	0.0412	0.9949
4	0.0888	0.0697	0.9913	0.1065	0.0844	0.9905	0.3230	0.2632	0.7601
5	0.0931	0.0764	0.9904	0.0998	0.0859	0.9908	0.0496	0.0376	0.9953
6	0.0849	0.0651	0.9921	0.0849	0.0665	0.9935	0.0693	0.0531	0.9902
7	0.0480	0.0382	0.9975	0.0374	0.0282	0.9989	0.0460	0.0367	0.9958
8	0.0501	0.0401	0.9972	0.0463	0.0354	0.9982	0.0360	0.0281	0.9974
9	0.0414	0.0335	0.9981	0.0178	0.0127	0.9997	0.0660	0.0508	0.9913
10	0.0415	0.0337	0.9981	0.0203	0.0141	0.9997	0.0245	0.0179	0.9987
11	0.0422	0.0342	0.9980	0.0231	0.0162	0.9996	0.0250	0.0193	0.9987
12	0.0429	0.0348	0.9980	0.0246	0.0190	0.9994	0.0223	0.0179	0.9990
13	0.0429	0.0347	0.9980	0.0262	0.0195	0.9995	0.0222	0.0173	0.9990
14	0.0422	0.0343	0.9980	0.0231	0.0169	0.9996	0.0194	0.0147	0.9992
15	0.0405	0.0328	0.9982	0.0229	0.0171	0.9996	0.0230	0.0175	0.9989
16	0.0421	0.0344	0.9981	0.0184	0.0137	0.9997	0.0197	0.0149	0.9992
17	0.0407	0.0332	0.9982	0.0147	0.0109	0.9998	0.0197	0.0145	0.9992
18	0.0403	0.0329	0.9982	0.0170	0.0123	0.9997	0.0219	0.0163	0.9990
19	0.0401	0.0326	0.9982	0.0132	0.0095	0.9999	0.0209	0.0166	0.9991

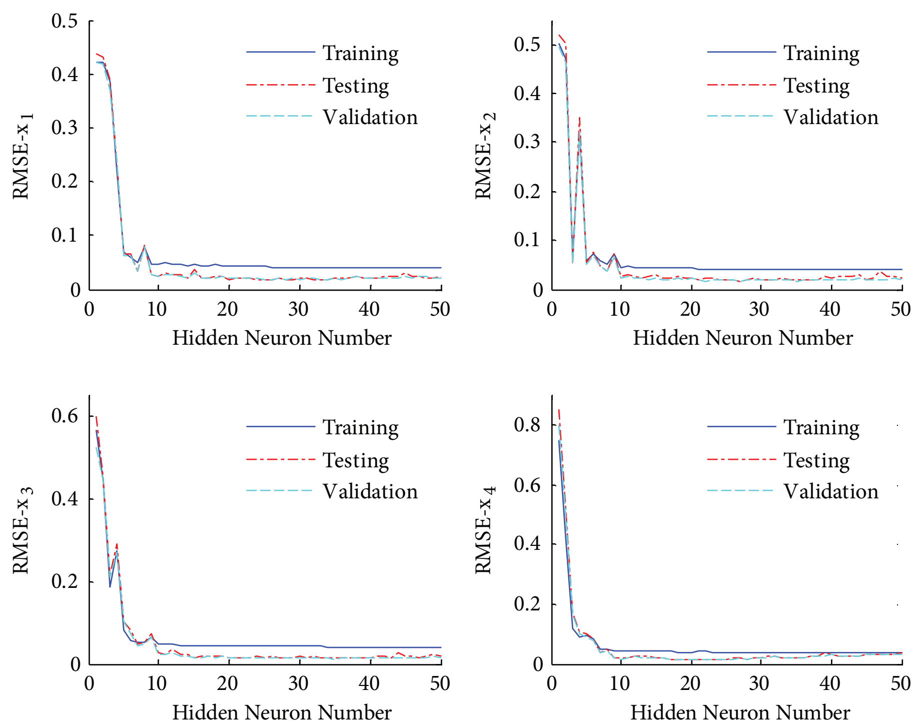

Figure 5. The training, validation, and testing errors with respect to the hidden layer nodes of ELMs models.

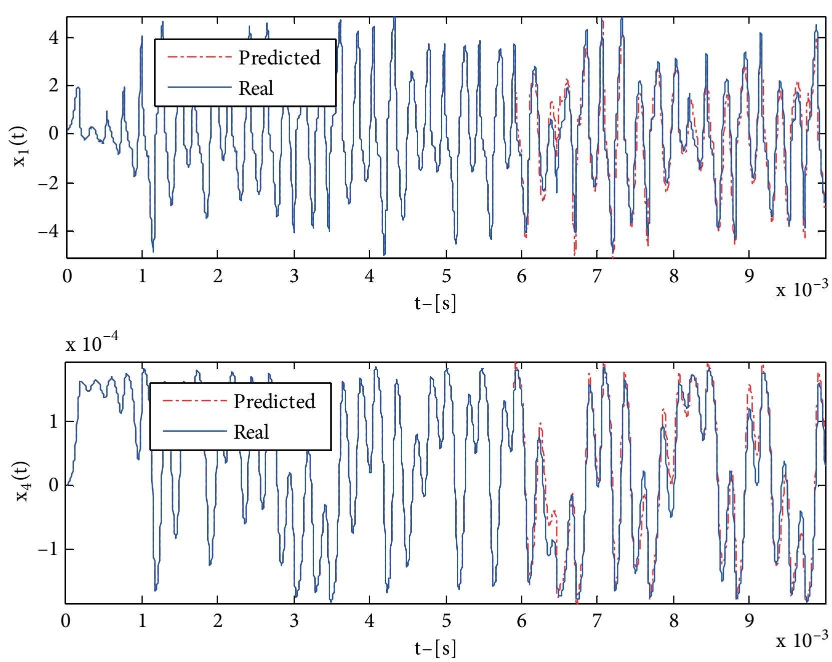

Figure 6. 1104-step iterated-ahead prediction results of ELM models estimating x_1 and x_4.

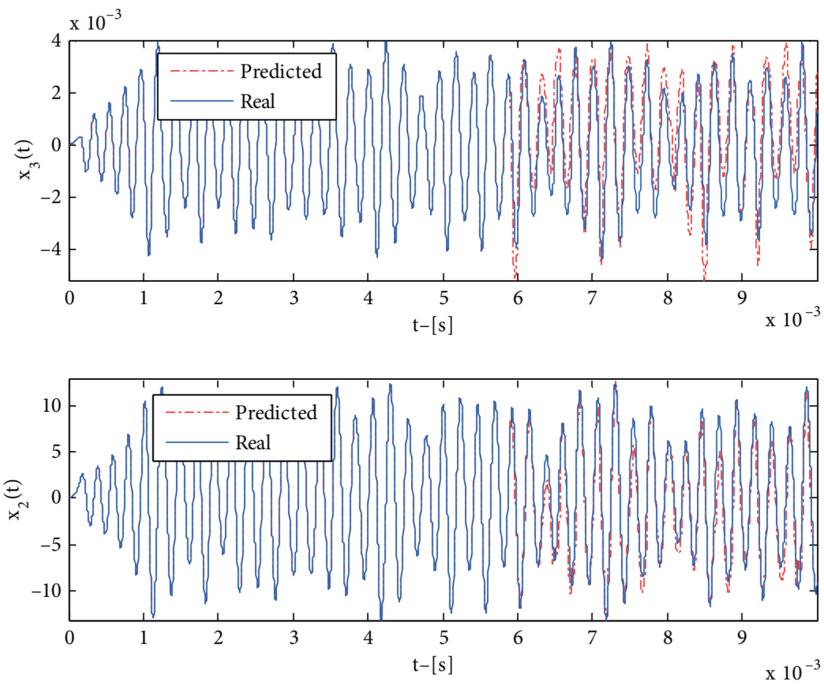

Figure 7. 1104-step iterated-ahead prediction results of ELM models estimating x_2 and x_3.

In addition, the graphs of power spectral density for the reconstructed data and original ones are given in Figure 10 in order to verify the effectiveness of the proposed method. These figures show that the power spectral density of ELMs very closely follows the original ones. This means that the proposed system completely behaves like a memristor-based chaotic Chua's circuit.

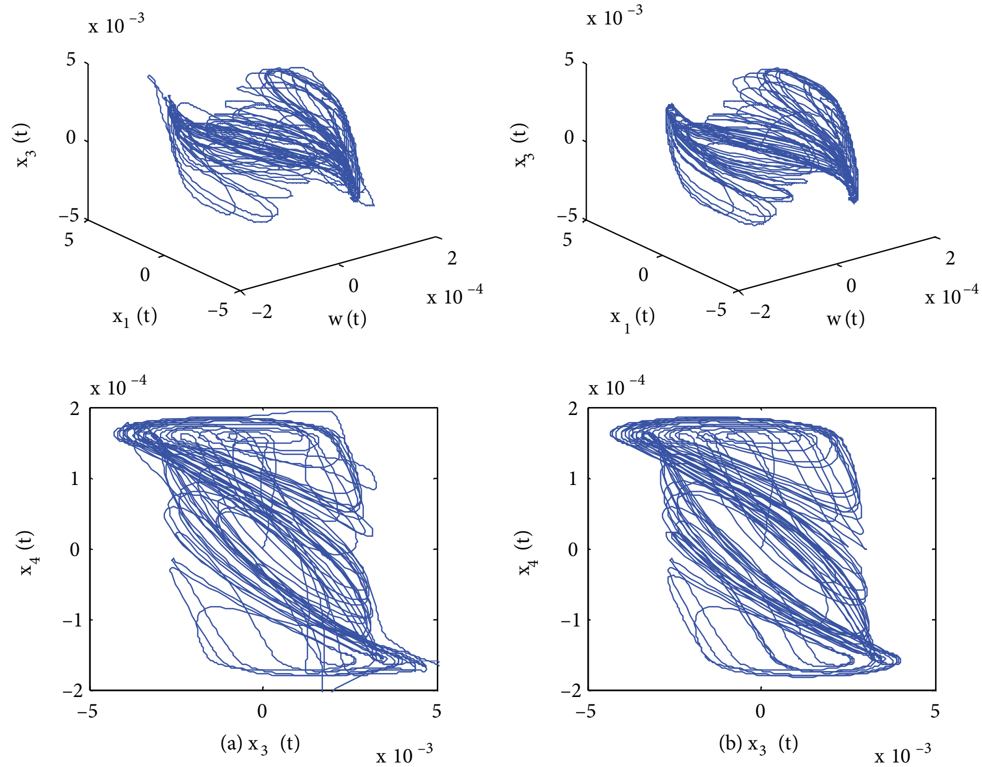

Figure 8. Attractors of: (a) the reconstructed data by ELM models; (b) the MATLAB model of circuit.

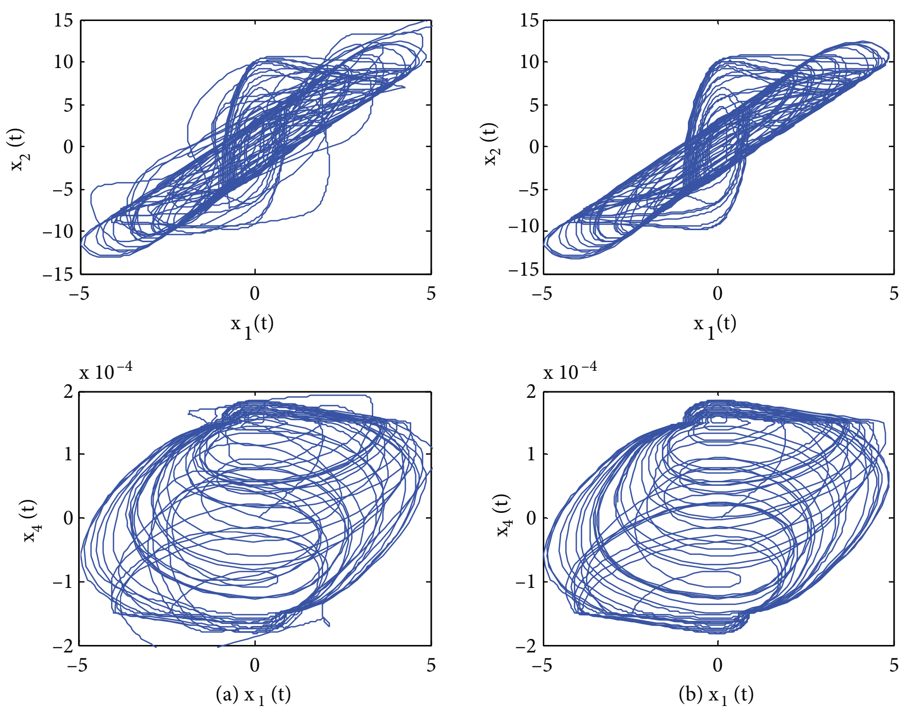

Figure 9. Attractors of: (a) the reconstructed data by ELM models; (b) the MATLAB model of circuit.

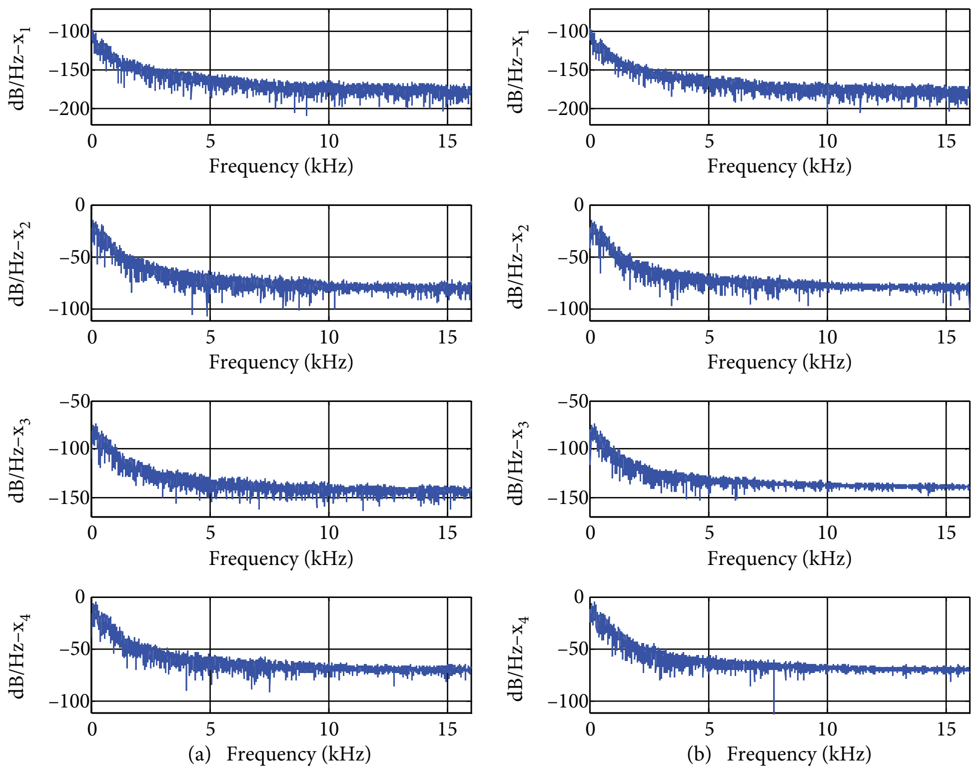

Figure 10. Power spectral density of: (a) the ELM models; (b) the MATLAB model of circuit.

Furthermore, a version using SVMs instead of ELMs in the proposed method and MLRs were used in order to compare the method. Since MLRs are high speed regressors and SVMs have high modeling performance and very low training and testing time [33,45], they were used in comparisons. Since SVMs are superior to the other networks such as SLFNNs and RBFNs in the literature, only the SVMs are evaluated in this study. Figures 11–13 show scatter plots of ELMs, SVMs, and MLRs for the results of only 1104-step. The R values of ELMs are 0.9374, 0.9726, 0.9041, and 0.9917, respectively. The R values of SVMs are 0.8278, 0.9264, 0.9961, and 0.9717, respectively. Figures 11 and 12 show that the ELMs and SVMs can be used to learn the behavior of Chua's circuit. However ELMs have a better ability to learn dynamic behavior than SVMs. In Figure 13 it was observed that the R values of MLRs relating to each variable corresponded to 0.5582, 0.8728, 0.1637, and 0.9333, respectively. The results for x_4 and x_2 are relatively acceptable, but the others are far from reasonable characteristics. It can be said that MLRs cannot be preferred to learn the behavior of Chua's circuit on account of their bad prediction performance.

The multistep-ahead prediction results obtained by using MLRs and the best ELM models and SVM models are given in Table 3 in terms of RMSE, MAE, R, and their best parameter values by taking all time intervals into consideration. It can be observed from Table 3 that the SVM shows a good prediction performance with RMSE values in the range of 1.62e − 04–1.2422. For ELM models, a high prediction performance with RMSE values in the range of 1.22e − 05–1.2087 is obtained.

The training, parameter selection, testing, and prediction times of ELMs, SVMs, and MLRs are presented comparatively in Table 4. It can be observed from Table 4 that MLRs have only an advantage in the training and testing times. On the other hand, the parameter selection time of SVM is much too long. This period is not appropriate for online applications.

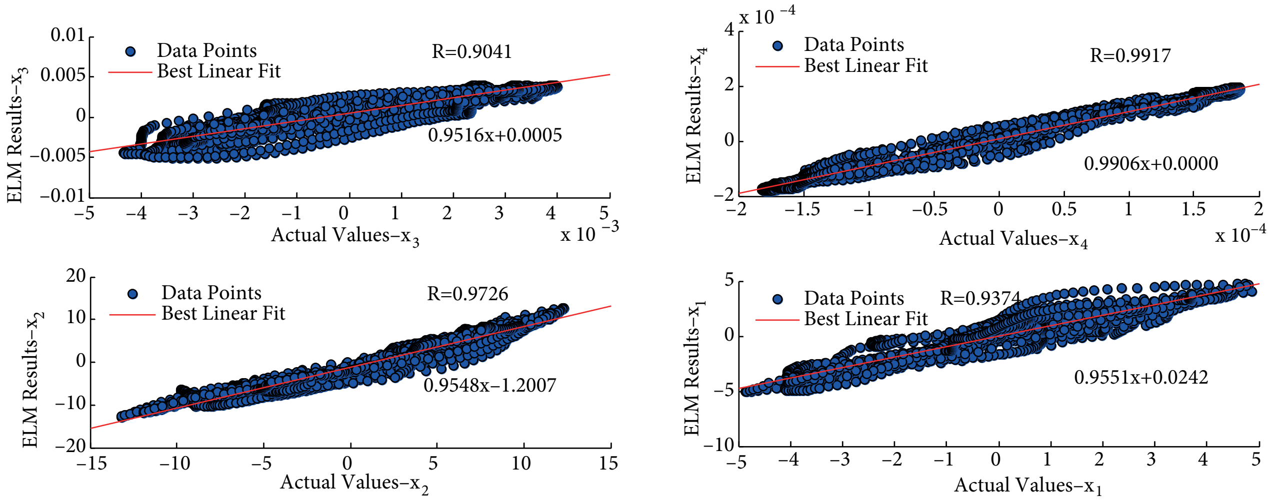

Figure 11. Scatter plots of the ELM models for the testing stage.

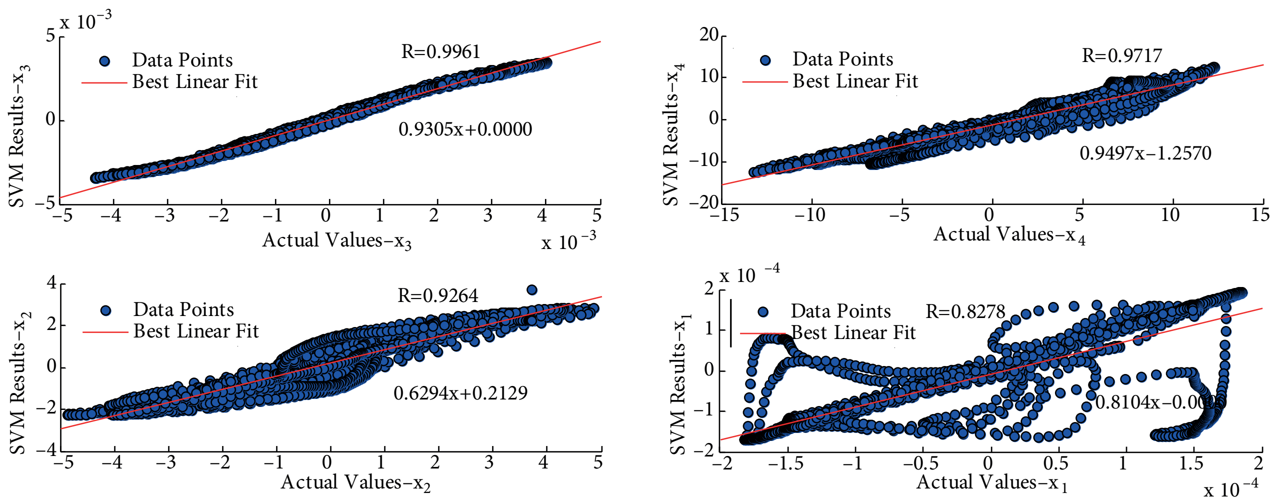

Figure 12. Scatter plots of the SVM models for the testing stage.

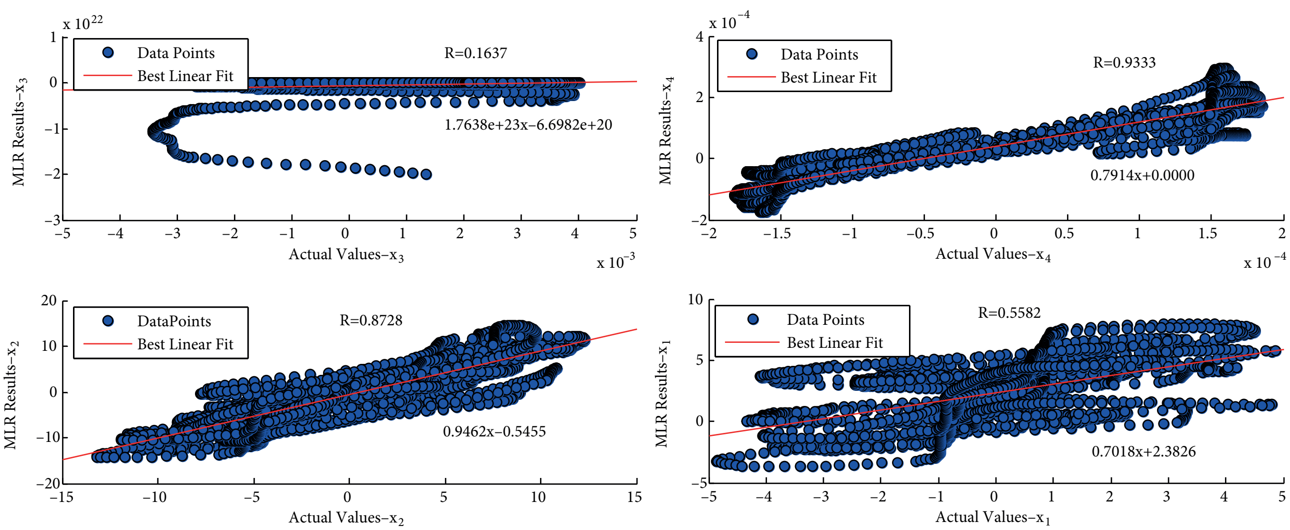

Figure 13. Scatter plots of MLRs for the testing stage.

Table 3. Performance comparison of ELM, SVM, and MLR as to the RMSE, MAE, and R statistics.

Model		ELM	SVM	MLR
RMSE for 1104 steps	x_1	0.4624	0.4256	2.9066
	x_2	1.2087	0.5982	3.4654
	x_3	0.0007	1.6259e –04	1.65e + 21
	x_4	1.22e – 05	1.2422	5.19e – 05
MAE for 1104 steps	x_1	0.2267	0.5672	2.1235
	x_2	0.5861	0.3048	2.4949
	x_3	0.0003	7.8547e – 05	2.71e + 20
	x_4	5.58e – 06	0.5955	3.79e – 05
R for 1104 steps	x_1	0.9750	0.9269	0.4665
	x_2	0.9846	0.9607	0.8663
	x_3	0.9556	0.9979	0.0966
	x_4	0.9960	0.9838	0.9326
Hidden Neuron Number and (C, γ, ε)	x_1	28	(2^{10},0.6, 2^{-5})	-
	x_2	27	(2^{10},0.6, 2^{-1})	-
	x_3	26	(2^{10},0.6, 2^{-6})	-
	x_4	19	(2^{10},0.6, 2^{-6})	-

Table 4. Performance comparison of ELM, SVM, and MLR as to the parameter selection, training, testing, and prediction times.

Model		ELM	SVM	MLR
Training time [s]	x_1	0.0624	0.5221	0.4709
	x_2	0.0312	0.0562	0.4539
	x_3	0.0624	1.0862	0.4755
	x_4	0.0312	1.0731	1.0657
Parameter selection time [s]	x_1	3.7905	709.0343	-
	x_2	3.7905	723.5186	-
	x_3	3.7905	670.7590	-
	x_4	3.7905	709.6261	-
Testing time [s]	x_1	0.0312	0.1018	0.0004
	x_2	0.0312	0.0060	4.14e – 05
	x_3	0.0156	0.4263	3.16e – 05
	x_4	0.0312	0.1402	0.0003
Predicting time [s]	x_1	12.0375	208.8224	1.0449
	x_2	11.6915	35.7368	1.0151
	x_3	11.2808	875.8024	0.8729
	x_4	6.1210	871.8570	2.3756

It is evident from the results presented in Table 4 that the training times of the four ELM models are 0.0624, 0.0312, 0.0624, and 0.0312 in seconds while the training times of the four SVM regressors are 0.5221, 0.0562, 1.0862, and 1.0731 in seconds. The ELM models run faster than the SVM models. Through these analyses it is obvious that a multiple ELM method is an efficient prediction method in comparison with a multiple SVM method. Consequently, ELM models are more appropriate than SVM models in real-time implementations. Furthermore, since ELM models can be designed by using such processors as FPGA, the memristor-based Chua's circuit can be practically realized too. All the analyses show that the proposed model is an accurate and fast method for the prediction of chaotic behavior of a memristor-based Chua' circuit.

To sum up, it can readily be concluded that the ELM model generally outperforms the excellent SVM model with higher approximation power and lower training and testing times for predicting the behavior of a chaotic circuit.

6. Conclusions

In the current paper an efficient and fast method for predicting the multistep-ahead behavior of a memristor-based chaotic Chua's circuit by using multiple ELM models was developed. The main aim of the proposed method was to take on the unparalleled features of ELM models (better generalization performance, fast learning speed, simpler and without employing tedious and time-consuming parameter tuning) to model the chaotic circuits and to predict its multistep-ahead behavior.

In this paper the chaotic time series in relation to the circuit variables spoilt by noise are first used for training the four ELM models. A one-step-ahead prediction of the chaotic behavior of the circuit is estimated by the trained ELMs. Next, a 1104-step-ahead prediction is perfectly carried out by feeding back the outputs of ELMs to their inputs for a time interval not known in advance. The chaotic time series are finally reconstructed by the prediction results and all chaotic attractors are reobtained.

The effectiveness of the proposed prediction method is shown by means of scatter plots, power spectral density, RMSE, MAE, R, and the training, testing, and prediction times. Experimental results demonstrated that the proposed method performed significantly well in a multistep-ahead prediction of the chaotic behavior. It was observed that the method with the four ELM models achieved the highest values of R (0.9750, 0.9846, 0.9556, and 0.9960) and the smaller values of RMSE (0.4624, 1.2087, 0.0007, and $1.22e-05$) and MAE (0.2267, 0.5861, 0.0003, and $5.58e-06$) for 1104 steps.

A comparative study was also conducted on the methods of multiple ELM models, multiple SVM models, and MLRs. The experimental results showed that the multiple ELM method significantly surpassed in performance the multiple SVM method in terms of higher prediction performance, shorter training, parameter learning, and prediction times; it also surpassed MLRs in terms of higher prediction performance.

In addition, it is notable to state that the framework of the multiple ELM method can also be applied to other nonlinear chaotic circuits and the different chaotic time series without determining such parameters as the embedding dimension and the time delay.

FPGAs can be used to obtain the memristor-based chaotic circuit model on account of the fact that an ELM is trained and tested in a short time. Hence, the multiple ELM method is an important step towards modeling real world memristor-based circuits. On the other hand, the other parallel computing systems can be used to speed up the generating process of the circuit model due to the fact that ELM models are trained and tested independently of each other. The next study will be designed in that manner.

Acknowledgment

The authors would like to thank Professor Guang-Bin Huang for his personal help with ELM and his codes.

References

[1] Chua LO. Memristor - the missing circuit element. IEEE T Circuit Theor 1971; 18: 507–519.

[2] Strukov DB, Snider GS, Stewart DR, Williams RS. The missing memristor found. Nature 2008; 453: 80–83.

[3] Muthuswamy B, Kokate PP. Memristor-based chaotic circuits. IETE Tech Rev 2009; 26: 417–429.

[4] Muthuswamy B. Memristor based chaotic circuits. Technical Report. University of California, No: UCB / EECS – 2009 - 6, Berkeley, November 2009.

[5] Biolek Z, Biolek D, Biolkova V. SPICE model of memristor with nonlinear dopant drift. Radioengineering 2009; 18: 210–214.

[6] Joglekar YN, Wolf SJ. The elusive memristor: properties of basic electrical circuits. Eur J of Phys 2009; 30: 661–675.

[7] Muthuswamy B, Chua LO. Simplest chaotic circuits. Int J Bifurcat Chaos 2010; 20: 1567–1580.

[8] Prodromakis T, Peh BP, Papavassiliou C, Toumazou C. A versatile memristor model with nonlinear dopant kinetics. IEEE T Electron Dev 2011; 58: 3099–3105.

[9] Yakopcic C, Taha T, Subramanyam G, Pino R, Rogers S. A memristor device model. IEEE T Electron Dev L 2011; 32: 1436–1438.

[10] Rak A, Cserey G. Macro modeling of the memristor in SPICE. IEEE T Comput Aid D 2010; 29: 632–636.

[11] Itoh, M, Chua LO. Memristor cellular automata and memristor discrete-time cellular neural networks. Int J Bifurcat Chaos 2009; 19: 3605–3656.

[12] Wu A, Zeng Z, Zhu X, Zhang J. Exponential synchronization of memristor-based recurrent neural networks with time delays. Neurocomputing 2011; 74: 3043–3050.

[13] Pershin Y, Ventra MD. Practical approach to programmable analog circuits with memristors. IEEE T Circuits-I 2010; 57:1857–1864.

[14] Wey TA, Jemison WD. An automatic gain control circuit with TiO2 memristor variable gain amplifier. Analog Integr Circ S 2012; 73: 663–672.

[15] Witrisal K. Memristor-based stored-reference receiver—the UWB solution? Electr L 2009; 45: 713–714.

[16] Merrikh-Bayat F, Bagheri-Shouraki S. Mixed analog-digital crossbar-based hardware implementation of sign-sign LMS adaptive filter. Analog Integr Circ S 2011; 66: 41–48.

[17] Talukdar A, Radwan A, Salama K. Generalized model for memristor-based wien family oscillators. Microelectr J 2011; 42: 1032–1038.

[18] Snider GS, Williams RS. Nano/cmos architectures using a field-programmable nanowire interconnect. Nanotechnology 2007; 18: 1–11.

[19] Wang W, Jing T, Butcher B. Fpga based on integration of memristors and cmos devices. In: 2010 IEEE International Symposium on Circuits and Systems; 30 May–2 June 2010; Paris, France. pp. 1963–1966.

[20] Cong J, Xiao B. Mrfpga: a novel fpga architecture with memristor-based reconfiguration. In: 2011 IEEE/ACM International Symposium on Nanoscale Architectures; 8–9 June 2011; San Diego, CA, USA. pp. 1–8.

[21] Xia Q, Robinett W, Cumbie MW, Banerjee N, Cardinali TJ, Yang JJ, Wu W, Li X, Tong WM, Strukov DB. Memristor-cmos hybrid integrated circuits for reconfigurable logic. Nano L 2009; 9: 3640–3645.

[22] Ebong IE. Training memristors for reliable computing. PhD, University of Michigan, USA, 2013.

[23] Driscoll T, Pershin Y, Basov D, Ventra MD. Chaotic memristor. Appl Phys A -Mater 2011; 102: 885–889.

[24] Itoh M, Chua LO. Memristor oscillators. Int J Bifurcat Chaos 2008; 18: 3183–3226.

[25] Lin ZH, Wang HX. Efficient image encryption using a chaos-based PWL memristor. IETE Tech Rev 2010; 27: 318–325.

[26] Koksal M, Hamamci SE. Analysis of four element Chua circuit containing new passive component memristor. In: 3rd Conference on Nonlinear Science and Complexity, Ankara, Turkey, 28–31 July 2010; pp.1–6.

[27] Brillinger DR. Time Series, Data Analysis, and Theory. New York, NY, USA: McGraw-Hill, 2001.

[28] Manabe Y, Chakraborty B. A novel approach for estimation of optimal embedding parameters of nonlinear time series by structural learning of neural network. Neurocomputing, 2007; 70: 1360–1371.

[29] Leung H, Lo T, Wang S. Prediction of noisy chaotic time series using an optimal radial basis function neural network. IEEE T Neural Netw 2001; 12: 1163–1172.

[30] Sapankevych NL, Sankar N. Time series prediction using support vector machines: a survey. IEEE Comput Intell Mag 2009; 4: 24–38.

[31] Farsa MA, Zolfaghari S. Chaotic time series prediction with residual analysis method using hybrid Elman–NARX neural networks. Neurocomputing 2010; 73: 2540–2553.

[32] Pan F, Zhang H, Xia M. A hybrid time-series forecasting model using extreme learning machines. In: Second International Conference on Intelligent Computation Technology and Automation; 10–11 Oct 2009; China. pp. 933–936.

[33] Haykin S. Neural Networks and Learning Machines. 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2008.

[34] Huang GB, Zhu QY, Siew CK. Real-time learning capability of neural networks. IEEE T Neural Networ 2006; 17: 863–878.

[35] Huang GB, Zhu QY, Siew CK. Extreme learning machine: theory and applications. Neurocomputing 2006; 70: 489–501.

[36] Huang GB, Zhou H, Ding X, Zhang R. Extreme learning machine for regression and multiclass classification. IEEE T Syst Man Cy B 2012; 42: 513–529.

[37] Decherchi S, Gastaldo P, Leoncini A, Zunino R. Efficient digital implementation of extreme learning machines for classification. IEEE T Circuits-II 2012; 59: 496–500.

[38] Suykens JAK, Vandewalle J. Learning a simple recurrent neural state space model to behave like Chua's double scroll. IEEE T Circuits-I 1995; 42: 499–502.

[39] Suykens JAK, Vandewalle J. Recurrent least squares support vector machines. IEEE T Circuits-I 2000; 47: 1109–1114.

[40] Hanbay H, Türkoğlu I, Demir Y. An expert system based on wavelet decomposition and neural network for modeling Chua's circuit. Expert Syst Appl 2008; 34: 2278–2283.

[41] Türk M, Oğraş H. Recognition of multi-scroll chaotic attractors using wavelet-based neural network and performance comparison of wavelet families. Expert Systems with Applications 2010; 37: 8667–8672.

[42] Wang XD, Ye MY. Identification of memristor-based chaotic systems using support vector machine regression. In: Luo Q. Editor. Information and Automation. Springer, 2011; 86: 365–371.

[43] Sun JC. Chaotic behavior learning of Chua's circuit. Chin Phys B 2012; 21: 100502-1–100502-7.

[44] Chang CC, Lin CJ. LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2011; 2: 1–27.

[45] Uçar A, Demir Y, Güzeliş C. A penalty function method for designing efficient robust classifiers with input–space optimal separating surfaces. Turk J Elec Eng & Comp Sci 2014; 22: 1664–1685.

[46] Cherkassky C, Ma Y. Practical selection of SVM parameters and noise estimation for SVM regression. Neural Networks 2004; 17: 113–126.