Chaotic System Modelling Using a Neural Network with Optimized Structure

Chapter *in* Studies in Computational Intelligence · March 2017

DOI: 10.1007/978-3-319-50249-6_29

CITATIONS
5

READS
775

4 authors:

Kheireddine Lamamra
Université Larbi Ben Mhidi Oum El Bouaghi
30 PUBLICATIONS **44** CITATIONS
SEE PROFILE

Sundarapandian Vaidyanathan
Vel Tech - Technical University
807 PUBLICATIONS **19,729** CITATIONS
SEE PROFILE

Ahmad Taher Azar
Prince Sultan University
413 PUBLICATIONS **6,323** CITATIONS
SEE PROFILE

Chokri Ben Salah
University of Sousse
27 PUBLICATIONS **399** CITATIONS
SEE PROFILE

Chaotic System Modelling Using a Neural Network with Optimized Structure

Kheireddine Lamamra, Sundarapandian Vaidyanathan, Ahmad Taher Azar and Chokri Ben Salah

Abstract In this work, the Artificial Neural Networks (ANN) are used to model a chaotic system. A method based on the Non-dominated Sorting Genetic Algorithm II (NSGA-II) is used to determine the best parameters of a Multilayer Perceptron (MLP) artificial neural network. Using NSGA-II, the optimal connection weights between the input layer and the hidden layer are obtained. Using NSGA-II, the connection weights between the hidden layer and the output layer are also obtained. This ensures the necessary learning to the neural network. The optimized functions by NSGA-II are the number of neurons in the hidden layer of MLP and the modelling error between the desired output and the output of the neural model. After the construction and training of the neural model, the selected model is used for the prediction of the chaotic system behaviour. This method is applied to model the chaotic system of Mackey-Glass time series prediction problem. Simulation results are presented to illustrate the proposed methodology.

K. Lamamra (✉)
Department of Electrical Engineering, University of Oum El Bouaghi, Oum El Bouaghi, Algeria
e-mail: l_kheir@yahoo.fr

K. Lamamra
Laboratory of Mastering of Renewable Energies, University of Bejaia, Bejaia, Algeria

S. Vaidyanathan
Research and Development Centre, Vel Tech University, Avadi, Chennai, Tamil Nadu, India
e-mail: sundarvtu@gmail.com

A.T. Azar
Faculty of Computer and Information, Benha University, Benha, Egypt
e-mail: ahmad.azar@fci.bu.edu.eg; ahmad_t_azar@ieee.org

A.T. Azar
Nanoelectronics Integrated Systems Center (NISC), Nile University, Cairo, Egypt

C. Ben Salah
Control and Energy Management Laboratory (CEMLab), Department of Electrical Engineering, National School of Engineers of Sfax, Sfax, Tunisia
e-mail: chokribs@yahoo.fr

A.T. Azar et al. (eds.), *Fractional Order Control and Synchronization of Chaotic Systems*, Studies in Computational Intelligence 688,
DOI 10.1007/978-3-319-50249-6_29

Keywords Modelling · Neural networks · Chaotic system · Multilayer perceptron · Mackey-glass time series · Prediction · NSGA II

1 Introduction

Dynamical systems serve as mathematical models for many exciting real-world problems in many fields of science, engineering and economics [1]. Dynamical systems can be classified into two categories: (1) continuous-time and (2) discrete-time dynamical systems.

A continuous-time dynamical system is generally represented as a system of differential equations given by

$$\frac{dx}{dt} = F(x, t) \tag{1}$$

where F is a continuously differentiable function and $x \in R^n$ is the state vector.

A discrete-time dynamical system is generally described by

$$x(k+1) = G(x(k), k) \tag{2}$$

where G is a continuous function and $x \in R^n$ is the state vector.

A dynamical system is called *chaotic* if it is very sensitive to initial conditions, topologically mixing and exhibits dense periodic orbits [2–4, 5, 6].

Chaos modeling with nonlinear dynamical systems is an active area of research [2–4, 7, 8].

Chaos theory has applications in different fields such as medicine [9, 10], chemical reactions [11–14], biology [15], Tokamak systems [16, 17], dynamos [18, 19], population biology systems [20, 21], oscillators [22, 23], etc.

Several methods have been designed to control a chaotic system about its unstable equilibrium such as active control [24]; Pehlivan et al. [25], adaptive control [26–31], backstepping control [32], sliding mode control [33, 34], fuzzy control [35, 36], artificial neural networks [37, 38], etc.

Artificial Neural Network (ANNs) are networks inspired by biological neuron networks. In machine learning and cognitive science, Artificial Neural Networks (ANNs) are used to estimate functions that can depend on a large number of inputs.

Usually, research in the field of Artificial Neural Network (ANN) is focused on architectures by which neurons are combined and the methodologies by which the weight of the interconnections are calculated or adjusted [39, 40].

The use of neural networks is very large because of its advantages such as the ability to adapt to difficult environments and that change its behaviour, etc. [41]. The Artificial Neural Networks (ANN) are used to model and control both linear and nonlinear dynamical systems [42, 43].

The research on artificial neural networks gives great importance on the construction of an appropriate network, how to combine neurons together, how to calculate and adjust the connection weights and how to find other parameters of the Artificial Neural Networks [44, 45].

Currently some biologists, physicists and psychologists are carrying out research to develop a neural model that is able to simulate the behaviour of the brain by improving the accuracy and the precision. Some engineers are interested in the neural network structure and how to improve their powerful computing capabilities [46, 47].

Recently, a Multi-Layer Perceptron neural network with a fast learning algorithms [48] was developed for the prediction of the concentration of the post-dialysis blood urea. Eight different learning algorithms are used to study their capabilities and compared their performances. Artificial neural networks are used for breast cancer classification [49]. Artificial Neural Networks (ANNs) are used to model a photovoltaic power generation system, by applying the Levenberg–Marquardt algorithm adopted into back propagation learning algorithm for training a feed-forward neural network [50].

In this chapter, the ANNs are used to model a nonlinear chaotic system of Mackey Glass and the structure of the neural model is optimized by Non-dominated Sorting Genetic Algorithm II (NSGA-II), which ensures the obtaining of an optimal neural structure and its learning. We use this neuronal model of Mackey Glass chaotic system to predict the time series of this system and also to test the effectiveness of this technique for the construction of optimal structures for chaotic systems. The resulting model can be also used for the control and synchronization of chaotic systems.

This chapter is organized as follows. Section 2 details the related work in modelling chaotic systems with neural networks. Section 3 details the chaotic systems and the prediction in chaotic systems. Section 4 outlines the modelling of systems using neural networks. Section 5 describes the operating principle of the NSGA-II multi-objective genetic algorithm. Section 6 details the construction and the learning of the MLP neural model using NSGA-II algorithm. As an application, Sect. 7 discusses the Mackey–Glass chaotic time series prediction. Section 8 details the simulation results. Section 9 contains a discussion of the main results. Section 10 contains the conclusions of this work and suggests future research directions.

2 Related Work

Neural networks have many applications in chaos theory. Many researchers have used neural networks in their work for modelling or control of chaotic systems.

Recently, Cellular Neural Networks (CNN) have been applied for the modelling and control of chaotic systems [51–55]. Cellular Neural Networks are similar to

neural networks with the difference that communication is allowed between neighbouring units only.

Chen [56] presented a work in which he proposed the application of neural-network based fuzzy logic control to a nonlinear time-delay chaotic system.

Zhang and Shen [57] have studied memristor-based chaotic neural networks with both time-varying delays and general activation functions. Pham et al. [58] proposed a novel simple neural network having a memristive synaptic weight.

Wen et al. [59] have studied the problem of exponential lag synchronization control of memristive neural networks via the fuzzy method and applications in pseudorandom number generators.

Kyprianidis and Makri [60] have presented a study of a complex dynamics of a system of two nonlinear neuronal cells, coupled by a gap junction, which is modelled as a linear variable resistor and the coupled cells of the FitzHugh-Nagumo oscillators systems.

He et al. [61] proposed a pinning control method focused on the chaotic neural network, and they demonstrated that the chaos in the chaotic neural network can be controlled with this method and the states of the network can converge in one of its stored patterns if the control strength and the pinning density are chosen in a suitable manner.

Ramesh and Narayanan [62] discussed the chaos control of Bonhoeffer–van der Pol oscillator using neural networks. Ren et al. [63] have proposed a dynamic control method using a neural network for unknown continuous nonlinear systems through an online identification and adaptive control.

Neural networks have been applied for the synchronization of chaotic systems. Many control methods have been used for the synchronization of chaotic systems such as sampled-data control [64], fuzzy neural control [65], adaptive control [66], chaotic time-series method [67], simulated annealing method [68], sliding mode control [69], time-delay control [70], etc.

In this chapter, we use the Multi-Layer Perceptron (MLP) neural networks optimized by multi-objective genetic algorithms of NSGA-II type for the modelling of chaotic systems and making the prediction.

In this approach, the training of the resulting neural network is provided by the NSGA-II algorithm and the structure is considered optimal, since the NSGA-II algorithm evolves over many generations. Each generation of the NSGA-II algorithm is composed of several individuals who are neural network models proposed to model the chaotic system. At the end of the evolution of the last generation, we obtain a set of solutions (called Pareto front) that are a set of models of the chaotic system, and we choose from this set, the model that suits us most. The details of the NSGA-II algorithm are presented in Sect. 5.

This optimization technique of the neural network structure has been used in further work and has proven its efficiency. We cite for example; the use of radial basis function (RBF) neural networks optimized by the NSGA-II multi-objective genetic algorithm for modelling of a nonlinear systems of Box and Jenkins which is a gas-fired boiler. For this model, the input is the gas at the inlet and the output is the concentration of released CO2. In the neural networks model for the gas-fired

boiler, the NSGA-II algorithm is used to find the best number of neurons in the hidden layer of the RBF neural network and provide the best connection weights between neurons in the hidden layer and the output layer, and also find the parameters of the radial function of neurons hidden layer. A Gaussian form of the radial functions is used in all the neurons. The NSGA-II algorithm finds the best centers and the best widths sigma for the Gaussian functions. The chromosome then contains the number of neurons in the hidden layer, Gaussian functions centers and widths of the hidden layer neurons, and the weights of connections between the hidden layer and the output layer [71].

3 Chaotic Systems and Prediction

3.1 Chaotic Systems

Chaotic systems are nonlinear dynamical systems which are very sensitive to initial conditions [38]. Chaotic behaviour is observed in many branches of science and engineering [2–4]. The behaviour of chaotic systems can be studied with chaotic mathematical models [2, 3, 72]. Chaos theory has applications in several areas such as computer science, engineering, physics, biology, meteorology, sociology, economics, etc. [4, 8, 72].

3.2 Prediction in Chaotic Systems

The prediction in chaotic systems has important applications in science and engineering [73, 74]. Chaotic systems like Lorenz system have important applications in weather models [4]. Several errors of weather forecasts are caused by the use of overly simplified models and the lack of accurate measurement of various parameters such as pressure, temperature, wind speed, etc. [74].

The existence of chaos was first introduced by Poincaré at the end of the nineteenth century [4]. Poincaré discovered chaos when he investigated the stability of the three-body model (e.g. Earth-Moon-Sun). Poincaré tested the stability of the three-body model by comparing the trajectories followed by one of the bodies from two very close initial positions. These phase trajectories remain close to each other in the short term and can therefore predict eclipses, but they become completely different in the long term. This is due to the chaotic nature of the three-body problem (e.g. Earth-Moon-Sun).

Thus, there is a need to build a good mathematical model that can make good predictions of the trajectories of the chaotic system. Neural networks serve as a good mathematical model for building a chaotic system and making predictions of

the trajectories of the system. In this work, a Multi-Layer Perceptron (MLP) neural network is used to model a chaotic system and make prediction thereafter starting from this model.

4 Modelling Using a Neural Network

Artificial neural networks have the approximation property which may be stated as follows: Every bounded function sufficiently regular can be approximated with arbitrary precision in a finite area of space of its variables, by a neural network comprising a layer of hidden neurons finite in number, all possessing the same activation function and a linear output neuron [75, 76].

Usually we do not use neural networks to make approximations of known functions. The system identification problem is typically to find a relationship between a set of outputs of a given process, and all the corresponding entries, through the measurements. It is assumed that this relationship exists despite the measures are finite in number, which are often blemished by noise [77]. Also, all the variables that determine the outcome of the process may not be measurable. Thus, neural networks are considered as important models for system identification using available outputs of the system [3, 78–80].

Generally, a neural network allows to make better use of available measures than the conventional linear approximation methods [77, 81]. This advantage is especially important when the process to be modelled depends on several variables such as in the case of shaping processes where it intervenes several types of non-linearity and multiple hardware and technological parameters. Thus, the concept of classification can be conducted to a problem of approximation of nonlinear regression functions. For this reason, neural networks are frequently used as classifiers or discriminators [82].

When the data related to the system are well controlled and with significant number and where the boundaries between each collected data class are not overly complex, neural networks are used to perform sorting and pattern recognition of surfaces, characters, or symbols [83–85].

5 The NSGA-II Multi-objective Genetic Algorithm

The construction and learning of the Multi-Layer Perceptron (MLP) neural model for a chaotic system are performed by the Multi-Objective Genetic Algorithm of NSGA-II type (Non-dominated Sorting Genetic Algorithm). NSGA-II algorithm has important applications in multi-objective optimization problems [86].

NSGA-II algorithm is a popular genetic algorithm in multi-objective function optimization theory [86]. NSGA-II algorithm has been used in many research works in multi-objective optimization problems [87–91].

NSGA-II algorithm is an improved version of NSGA algorithm for multi-objective function optimization [92]. NSGA-II algorithm has helped to solve the problems encountered in the NSGA algorithm such as the complexity, non-elitist and the use of sharing.

NSGA-II algorithm makes the dominance relationships between individuals and provides a fast sorting method of chromosomes [86, 93]. It uses a selection operator based on the measure of crowding around individuals to ensure diversity in the population.

NSGA-II algorithm is an elitist algorithm. In order to manage elitism, NSGA-II algorithm evolves so that at each new generation, the best individuals encountered are retained. The operating principle of NSGA-II algorithm is shown in Fig. 1 [94].

The working principle of NSGA-II algorithm is described as follows: At first, an initial population is randomly created. Then a sorting operation is performed using the non-domination concept. For each solution, we assign a rank equal to the level of non-dominance, viz. the rank 1 for best, 2 for the next level, etc. Then, a tournament of selection of parents is performed during the reproduction process.

Once two individuals of the population are randomly chosen, the tournament is performed on a comparison of the domination with constraints of the two individuals.

For a given generation t, after creating a children population Q_t from the previous population P_t (generated from the parents via the genetic operators, crossover and mutation), a population R_t is created that includes the parents population P_t and the children population Q_t such that $R_t = P_t \cup Q_t$. This ensures the elite nature of the NSGA-II algorithm. Then the population R_t contains twice the size of the previous population, i.e. R_t has $2N$ individuals consisting of N parents and N children).

Next, the concept of non-dominance of Preto is applied to sort the population R_t. Then the individuals of R_t will be grouped in successive fronts $(F_1, F_2, \ldots)$ where F_1 represents individuals of rank 1, F_2 represents individuals of rank 2, etc.

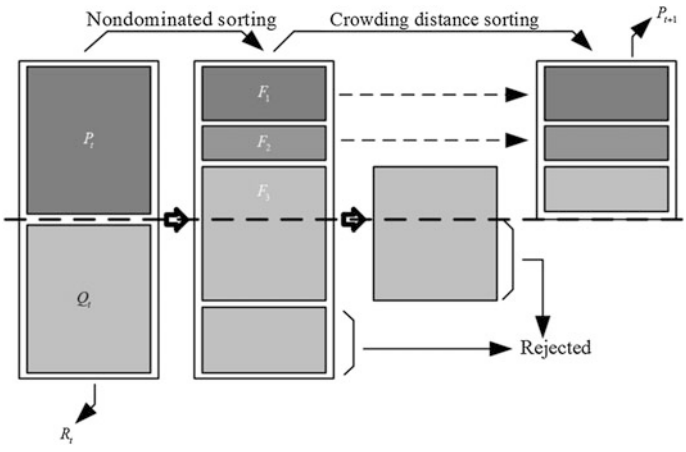

Fig. 1 Operating principle of NSGA-II algorithm

After the sorting, the size of R_t should be reduced to N individuals in order to form the next population P_{t+1}. Thus, N individuals from R_t must be excluded for forming the next population. If the size of the front F_1 is less than N, then all its individuals are preserved and the same procedure is followed for the other fronts while the number of the preserved individuals does not exceed the size N.

In the example illustrated by Fig. 1, both fronts F_1 and F_2 are fully preserved. However, keeping the front F_3 will result in exceeding the size N of the population P_{t+1}. Thus, it is necessary to make a selection to determine which individuals to keep. For this purpose, NSGA-II performs a mechanism for preserving the diversity in the population based on the evaluation of the density of the individuals around each solution across a calculating procedure of the "distance proximity". Thereby, a low value of the proximity distance for an individual is an individual "well surrounded".

NSGA-II algorithm then proceeds with a descending sorting according to this proximity distance to preserve the necessary number of individuals of F_3 front and remove some individuals from the densest areas. In this manner, the population P_{t+1} is made up to N individuals and diversity is ensured. The individuals with extreme values of the criteria are maintained by this selection mechanism, which keeps the external bounds of the Pareto front.

At the end of this phase, the population P_{t+1} is created. Then a new population Q_{t+1} is generated from P_{t+1} by the reproduction operators.

The above procedure is repeated by ensuring elitism and diversity until the stopping criteria defined beforehand is reached.

6 Construction and Learning of the MLP Neural Model by NSGA-II Algorithm

In this work, the learning of the MLP neural model of a chaotic system is performed by using Multi-Objective Genetic Algorithms of NSGA-II type. Therefore this genetic algorithm is used to optimize the structure and parameters of the MLP neural model through the optimization of the following objective functions:

- The first function to be optimized (f_1) is the number of neuron of the hidden layer of the MLP neural model.
- The second function to be optimized (f_2) is the quadratic cumulative error which is the square sum of the difference between the desired output and the output of the MLP neural model.

The NSGA-II algorithm evolves to find the best neurons number of the hidden layer denoted N_{hl} and to provide the best connection weights between neurons of the input layer of the MLP neural model and the hidden layer, and also the best connection weights between the hidden layer and the output layer.

Then the chromosome contains the number of neurons in the hidden layer, and the connection widths connecting the different layers of the MLP neural model.

The MLP neural network used here to model a chaotic system is composed of three layers:

- The Input layer, which is composed of two neurons. The first neuron corresponds to the input data of the chaotic system, whereas the second neuron corresponds to the modelling instantaneous error.
- The hidden layer, which is composed of an undetermined number of neurons (N_{hl}). In fact, the number N_{hl} varies according to the optimized neural network model. The multi-objective genetic algorithm NSGA-II seeks to find the best number that ensures the best possible structure with the smallest model error in order to guarantee good modelling of the chaotic system.
- The output layer, which is composed of one neuron. This neuron corresponds to the output of the neural model of the chaotic system. Our objective is to keep this output the nearest possible to the desired output of the chaotic system to ensure thereafter (after the construction of the neural model) a good prediction of the future states of the chaotic system.

The structure of the neural model used is shown in Fig. 2.

In Fig. 2, x_1 corresponds to the chaotic system input data, x_2 corresponds to the modelling instantaneous error and y corresponds to the output of the neural model of the chaotic system.

W is the matrix of the connection weights of the input layer neurons and the neurons of the hidden layer. In our case, this matrix is composed of W_{11} to W_{1Nhl} which are the x_1 input connection weights with the neurons of the hidden layer N_{hl} and W_{21} to W_{2Nhl} that are the x_2 input connection weights with the neurons of the hidden layer N_{hl}. Therefore, the size of the matrix W is $2 \times N_{hl}$.

Z is a vector composed of Z_1 to Z_{Nhl} which are the connection weights between the neurons of the hidden layer N_{hl} and the neuron of the output layer y. Therefore, the size of the matrix Z is $1 \times N_{hl}$.

The chromosome of the multi-objective algorithm NSGA-II generic therefore takes the following form:

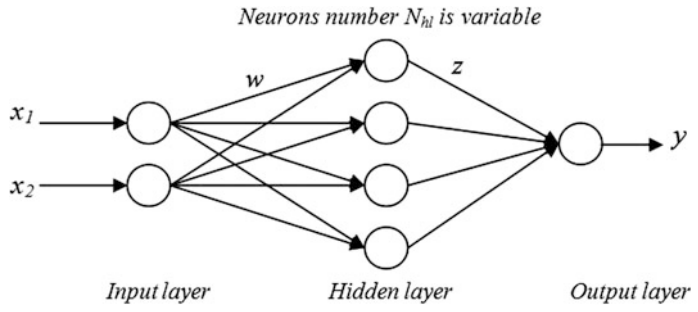

Fig. 2 Structure of the MLP neural model

$$Chromosome = [\,N_{hl} \quad W_{11} \quad \cdots \quad W_{1Nh1} \quad W_{21} \quad \cdots \quad W_{2Nh1} \quad Z_1 \quad \cdots \quad Z_{Nh1}\,]$$

The chromosome length (Lc) depends on the number of neurons in the hidden layer N_{hl} and it is given by the formula:

$$Lc = (N_{hl} \times 3) + 1.$$

For example, for a neural model with 4 neurons in the hidden layer (i.e. $N_{hl} = 4$), the chromosome will have a length equal to 13 (i.e. $Lc = 13$) and it will have the following form:

$$Chromosome = [\,4 \quad W_{11} \quad W_{12} \quad W_{13} \quad W_{14} \quad W_{21} \quad W_{22} \quad W_{23} \quad W_{24} \quad Z_1 \quad Z_2 \quad Z_3 \quad Z_4\,]$$

We have chosen a variation range of the number of neurons in the hidden layer N_{hl} between 2 and 20.

Then the population size matrix (Spm) will have the following size.

$$\text{Size}(Spm) = N \times Lc_{\max} = N \times [(N_{hl\max} \times 3) + 1]$$

where N is the number of individuals in the population, and $N_{hl\max}$ is the maximum number of neurons of the hidden layer of the current population. This number is not necessarily equal to 20, because the number of neurons in the hidden layer of each individual (or neural model) is randomly initialized (such as the initial connection weights) during the creation of the initial population at the beginning of the evolution of the multi-objective genetic algorithm NSGA-II.

Therefore, the size of the population in columns is not fixed and it takes every time a corresponding size to the greatest number of neurons in the hidden layer.

It is noted that this value (which cannot exceed 20 neurons, according to the interval that we have chosen) may change over the generations following the process of crossover and mutation.

Also, since we have to keep the same number of columns of the population matrix for all individuals, we have proceeded to put zeros in columns so that their N_{hl} is less than N_{hlmax}.

For example, we consider two individuals of the same population i, having a different number of neurons in the hidden layer. Suppose that the first individual has a $N_{hl1}(i) = 4$ and the second has a number $N_{hl2}(i) = N_{hlmax}(i) = 16$. In this case, the size of the population is:

$$Spm(i) = N \times Lc_{max}(i) = N \times (N_{hlmax}(i) \times 3 + 1) = N \times (16 \times 3 + 1) = N \times 49;$$

The chromosome of the second individual of this population (i) will have the following structure:

$$Chromosome_2(i) = [16\, W_{11}\, W_{12}\, \ldots\, W_{116}\, W_{21}\, W_{22}\, \ldots\, W_{216}\, Z_1\, Z_2\, \ldots\, Z_{N16}]$$

whereas the chromosome of the first individual of this population will have the following structure:

$$Chromosome_1(i) = [4\, W_{11}\, W_{12}\, \ldots\, W_{14}\, W_{21}\, W_{22}\, \ldots\, W_{24}\, Z_1\, Z_2\, \ldots\, Z_{N4}\, 0\, 0\, \ldots\, 0]$$

Thus, the lengths of the vectors of these two individuals are equal and they have the value of $L_{c1}(i) = L_{c2}(i) = N_{hlmax}(i) \times 3 + 1 = 49$. This is the length of all individuals within the population i. However, the chromosome of the first individual will include 36 zeros to reach the same size as its counterparts individuals within the population i.

It is quite obvious that the evolution process of the multi-objective genetic algorithm takes much time to provide in the end the non-dominated solutions (often they are the optimal solutions), but it does not hamper our technique. that This is because the NSGA-II algorithm is used to evolve off-line and at the end we choose the model which gives the smallest modelling error among those of the Pareto front to make the prediction without the intervention of NSGA-II algorithm.

7 Application to the Chaotic System: Mackey–Glass Chaotic Time Series Prediction

Our method is applied to model the chaotic system of MacKey–Glass Time Series with 1000 data for predicting the time series [95].

A time series is a sequence of observations on a variable measured at successive points in time or over successive periods of time. The measurements may be taken every hour, day, week, month, or year, or at any other regular interval. The pattern of the data is an important factor in understanding how the time series has behaved in the past. If such behaviour can be expected to continue in the future, we can use the past pattern to guide us in selecting an appropriate forecasting method.

To identify the underlying pattern in the data, a useful first step is to construct a time series plot. A time series plot is a graphical presentation of the relationship between time and the time series variable; time is on the horizontal axis and the time series values are shown on the vertical axis. Let us review some of the common types of data patterns that can be identified when examining a time series plot [96].

The chaotic system of MacKey–Glass Time Series is the first delay chaos discovered in 1977 from a physiological model. It is generated by the following differential equation:

$$\dot{x}(t) = \beta x(t) + \frac{\alpha x(t-\tau)}{1 + x^{10}(t-\tau)}$$

where $\alpha = 0.2, \beta = -0.1$ and $\tau = 17$. It is well-known that the MacKey-Glass system is chaotic with the fractal dimension 2.1 for these parameters [97].

For this chaotic system, when $x(0) = 1.2$ and $\tau = 17$, there is a non-periodic series and non-convergent time series which are highly sensitive to initial conditions.

The training phase of the neural model is done on 200 data and the validation on the 800 remaining data. The NSGA-II algorithm is applied to minimize simultaneously the following fitness functions:

1. The number of neurons in the hidden layer (N_{hl})
2. The quadratic cumulative error (Ec) that is the difference between the reference (or desired output) and neural model output.

$$Ec(W) = \sum_{i=1}^{m} (y_r(i) - y_d(i))^2$$

where:

- W is the matrix of connection weights.
- m is the number of *data of* input vector (equal to the number of data vector of desired outputs y_d); m is equal to 1000; (200 *data* for the training phase of the neural model; and 800 data for the validation phase*).*
- $y_r(i)$ is the *i*th output value of the MLP neural model.
- $y_d(i)$ is the *i*th output of the desired data.

8 Results of Simulation

The multi-objective genetic algorithm NSGA-II evolves along the generations with the aim to build, train and provide in the end a neural model with optimal structure and connection weights ensuring a good training (construction and training phase). The parameters of the NSGA-II algorithm used are the following:

- Search intervals:
  - Connexion weights: $W \in [-30, 30]$; $Z \in [-30, 30]$;
  - Neural number in the hidden layer $N_{hl} \in [2, 20]$
- The population size (number of individuals): $N = 100$
- Number of generations = 300
- Crossover Probability = 0.9
- Mutation Probability = 0.08
- Selection type: Stochastic selection

Table 1 Pareto front of the MLP NN Model

N° of individual	Hidden layer neurons number N_{hl}	Training error E_{ct}	Prediction error E_{cp}	Global error E_g
1	2	93.2314	399.7845	493.0159
2	3	15.3047	71.6695	86.9742
3	4	10.4758	35.8125	46.2883
4	5	4.2154	16.8541	21.0695
5	6	1.6245	7.5231	9.1476
6	7	1.1687	5.0054	6.1741
7	8	0.8876	3.3251	4.2127
8	9	0.8340	2.4975	3.3315

At the end of the evolution of the latest generation of the NSGA-II algorithm, a set of neural models is provided in the Pareto front (set of non-dominated solutions).

When the construction and training of the neural model are finished (construction and training phase), the selected model (among the Pareto front) is placed separately (without the intervention of the NSGA-II) for the prediction of the future of the Mackey Glass chaotic system.

The neural models of the Pareto front of the latest generation are given in the Table 1.

The training phase of the MLP neural model is carried out on 200 data, and the prediction phase is done on the 800 data.

E_{ct} is the training quadratic cumulative error; E_{cp} is the prediction quadratic cumulative error and E_g is the global quadratic cumulative error.

All the individuals (MLP neural models) of the Table 1 are non-dominated solutions of the Pareto front. Nonetheless, the individual N° 6 is chosen as the best solution because it provides the smallest error and also there is a great difference between this value and that of other individuals.

The choice of this individual is done through the first fitness function (which is the error) which has a higher importance than the second fitness function which is the number of neurons in the hidden layer of the neural model. This individual is a neural model, with nine neurons in the hidden layer $(N_{hl} = 9)$ and offers a global modelling error equal to 3.3315.

Figure 3 shows the Mackey Glass chaotic time series considered as the desired output y_d. Figs. 4 and 5 show the results of the training phase. Figures 5 and 6 show the results of the prediction phase. Figures 8 and 9 depict the representation of the overall data of the two phases together. Figure 10 shows the non-dominated individuals of the Pareto front of the latest generation of NSGA-II algorithm.

The Mackey Galss time series

See Fig. 3.

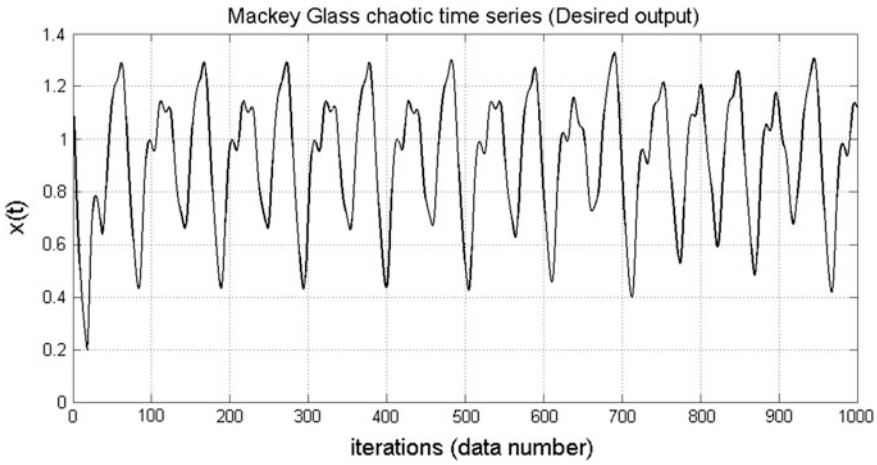

Fig. 3 The Mackey glass chaotic system time series

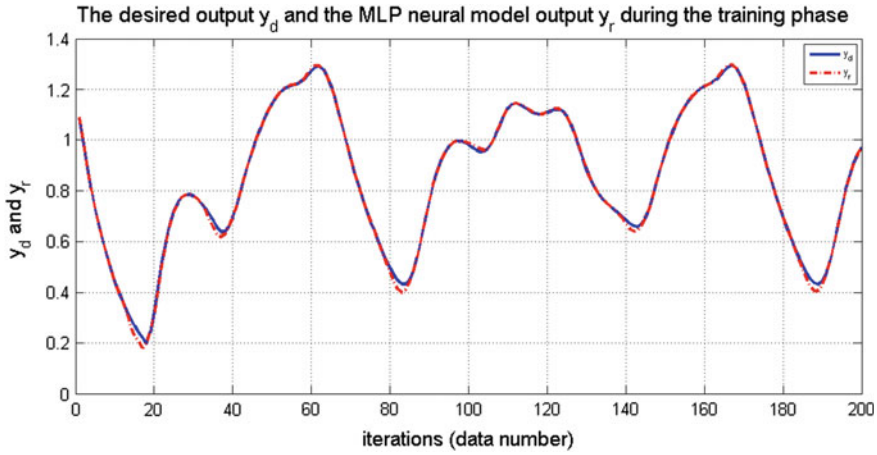

Fig. 4 The Mackey glass chaotic system time series and the output of MLP neural model during the training phase

The training phase:

The Mackey Glass chaotic time series data which are the desired output y_d and the output of MLP neural model y_r of the training phase are shown in Fig. 4 and the global instantaneous quadratic training error is represented in the Fig. 5.

The prediction phase:

The desired output y_d and the output of MLP neural model y_r of the prediction phase are shown in Fig. 6 and the global instantaneous quadratic prediction error is represented in Fig. 7.

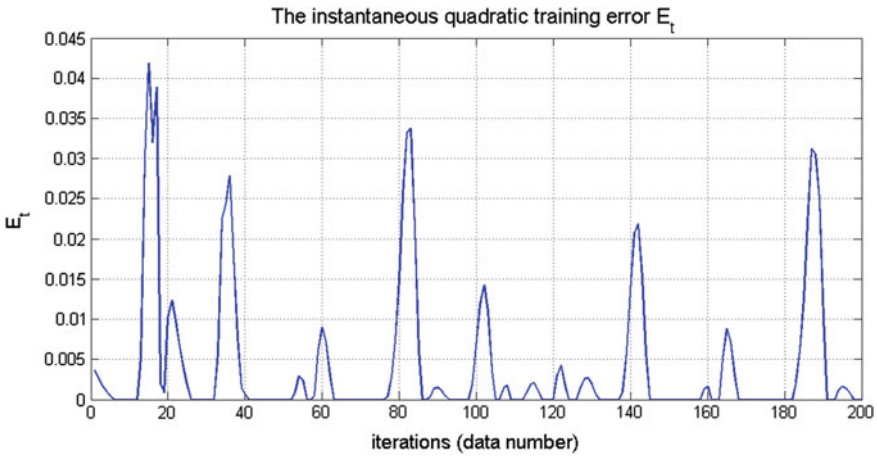

Fig. 5 The instantaneous quadratic training error

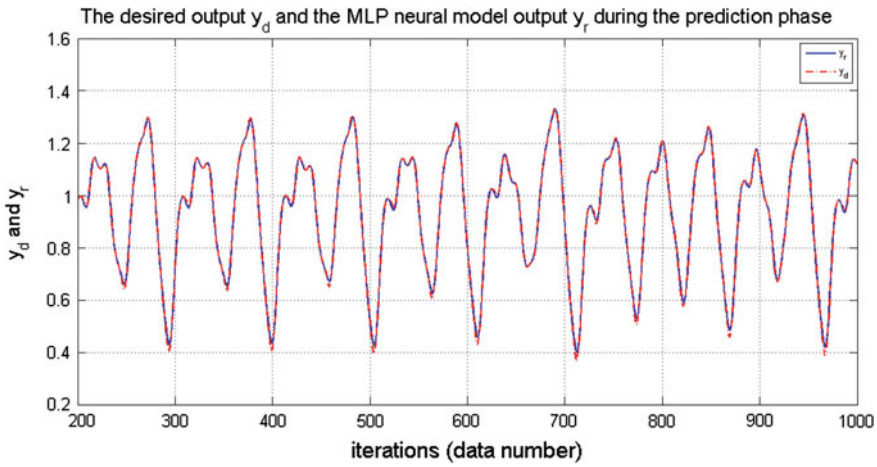

Fig. 6 The Mackey glass chaotic system time series and the output of MLP neural model during the prediction phase

The global data:

The global data of the desired output y_d and the output of MLP neural model y_r are shown in Fig. 8 and the global instantaneous quadratic error is represented in the Fig. 9.

The Pareto front:

Pareto front which is composed of non-dominated individuals (that are the MLP neural models) of the last generation of the multi-objective genetic algorithm NSGA-II is shown in Fig. 10.

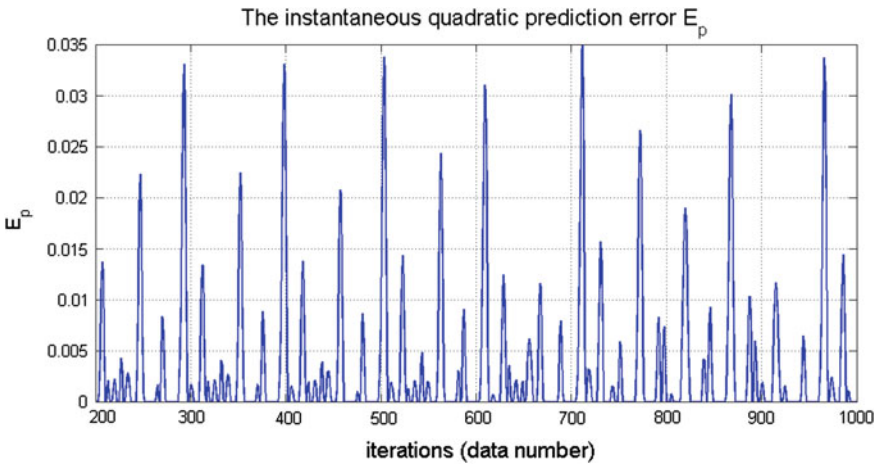

Fig. 7 The instantaneous quadratic prediction error

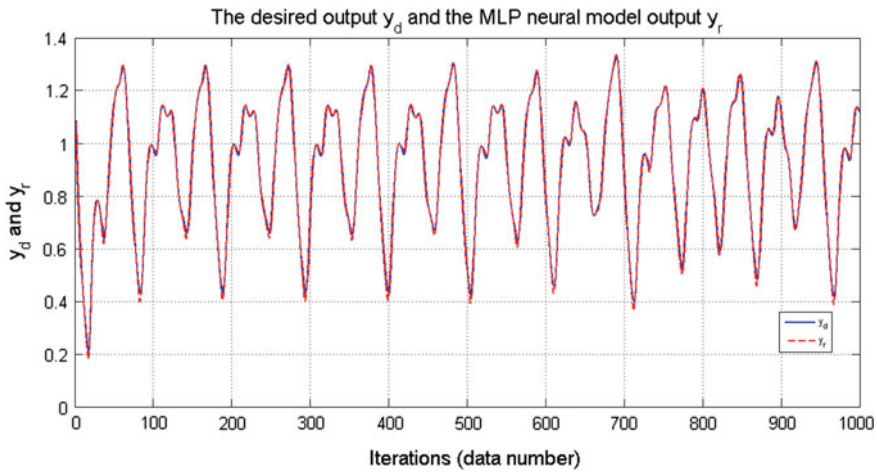

Fig. 8 The Mackey Glass chaotic system time series and the MLP neural model output

9 Discussion of Results

The figures shown in Sect. 8 represent the results of the individual which gives a global quadratic error equal to $E_g = 3.3315$ for 1000 data. This gives rise to an average modelling error $E_{gm} = 3.3315 \times 10^{-3}$. The training quadratic cumulative error is $E_{ct} = 0.8340$ for 200 data and the predictive quadratic cumulative error is $E_{cp} = 2.4975$ for 800 data. Thus, the average training error is $E_{ctm} = 4.17 \times 10^{-3}$

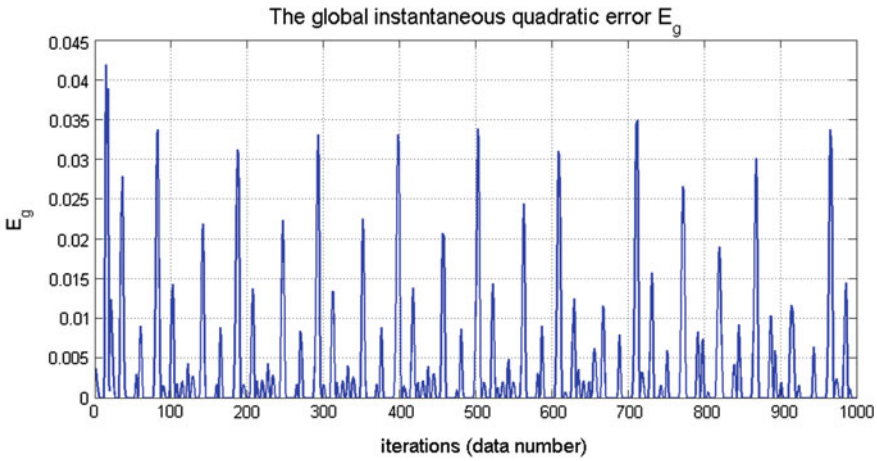

Fig. 9 The global instantaneous quadratic error

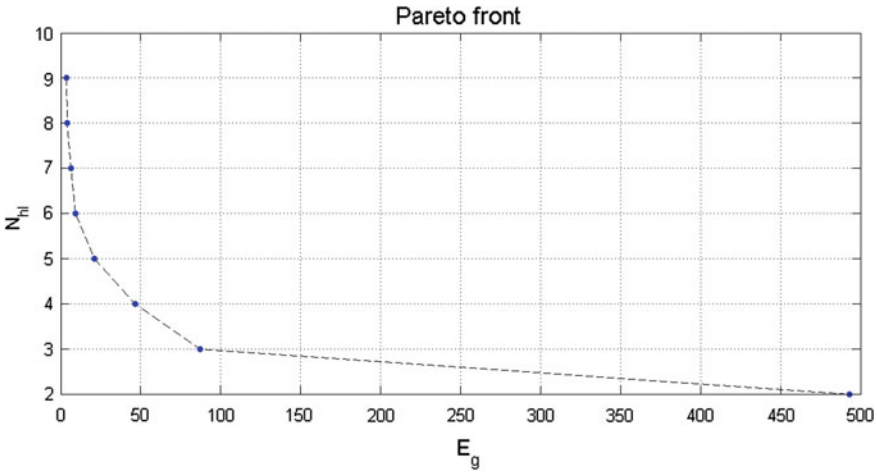

Fig. 10 The Pareto front

and the average prediction error is $E_{cpm} = 3.1218 \times 10^{-3}$. These errors are very small for a chaotic system modelling.

It can be noted that the average training error is bigger than the average prediction error. This is caused from the fact that during the training phase, the neural network needs to adapt to the changes coming to the chaotic system. This is a very important property of a neural network (it has the ability to adapt to the conditions imposed by any environment). During the training phase, the biggest peak in the training error (Fig. 4) appears at 15th iteration with a training error equal to 0.04192, and this error is the greatest instantaneous error for all data of the two

phases. During the prediction phase, the most significant peak in the prediction error is at the 712th iteration with a predictive error of 0.03486.

Through these results, we can observe that the output of MLP neural model has perfectly followed the Mackey Glass time series (the desired output). In Fig. 7, it is difficult to differentiate between these two signals (the desired output and the MLP neural model). According to these results, it is clear that the multi-objective genetic algorithm NSGA-II provided a good structure of the MLP neural network model for the Mackey Glass chaotic system, with a good number of neurons in the hidden layer and good connection weights linking the three layers that constitute this model, which brings us to note that the modelling process using this technique is very efficient.

10 Conclusion and Future Directions

Chaos theory has wide applications in several fields of physics, biology, chemistry, robotics, communication theory, networks, medicine, economics, etc.

Chaotic systems are very sensitive to initial conditions. Thus, the evolution of phase trajectories of chaotic systems follows a very complex pattern and it is a very difficult task to predict long-term behaviour of chaotic systems

In this work we presented the modelling of chaotic systems by Multi-Layer Perceptron (MLP) neural networks. Since the construction of an optimal structure of neural networks is usually difficult, we proceeded to optimize their structure by multi-objective genetic algorithm of NSGA-II type (Non-dominated Sorting Genetic Algorithm).

The proposed neural model is composed of three layers: an input layer, the output layer and the hidden lawyer. The input layer has two inputs which are the input data of the chaotic system and the modelling error. The output layer is composed of a single neuron that represents the output the neural model of the chaotic system. Also, the hidden layer consists of a variable number of neurons. The NSGA-II algorithm determines the optimum number of neurons in the hidden layer. Also, the NSGA-II algorithm is intended to provide the optimal connection weights between the three layers of the neural model of the chaotic system and thereby ensuring its training. Thus it operates to minimize the following two fitness functions: the number of neurons in the hidden layer and the modelling error. This method is applied in two phases. The first phase is the construction and training phase of the neural model with the NSGA-II algorithm. The second phase is the application of the neural model for predicting the future of the modelled chaotic system. Thus, once the construction and training phase of the neural model are finished, the model chosen (among the Pareto front of the latest generation of the NSGA-II) is used alone (without the NSGA-II) in the prediction phase.

This technique is applied to model the Mackey Glass chaotic system. The objective is to make from this neuronal model, the predicting of the Mackey Glass time series and test whether this technique is effective for the construction of

optimal structures for chaotic systems. The resulting model can be also used for the study, control and synchronization of the chaotic systems.

The obtained results show that the two signals of the desired data and of the neural model data are almost completely identical whether during the training phase or into the prediction phase. These results are very satisfying and encouraging and show that this method provides a good model of the chaotic system.

In future work, we plan to improve this technique for the neural networks type of Multi-Layer Perceptron (MLP) and for those of Radial basis function type, by the application of NSGA-II algorithm and even the use of others algorithms such that the Particle Swarm Optimization (PSO). These techniques will also be applied in different areas, such as the modelling and the control in the field of renewable energy mainly for the maximum power point tracking for dynamics photovoltaic systems, and also for the control of highly complex nonlinear systems.

References

1. Sundarapandian, V. (2012). *Ordinary and partial differential equations*. New Delhi, India: McGraw Hill Education.
2. Azar, A. T., & Vaidyanathan, S. (2015). *Chaos modeling and control systems design, Studies in computational intelligence* (Vol. 581). Germany: Springer. ISBN 978-3-319-13131-3.
3. Azar, A. T., & Vaidyanathan, S. (2015). *Computational intelligence applications in modeling and control. Studies in computational intelligence* (Vol. 575). Germany: Springer. ISBN 978-3-319-11016-5.
4. Azar, A. T., & Vaidyanathan, S. (2016). *Advances in chaos theory and intelligent control. Studies in fuzziness and soft computing* (Vol. 337). Germany: Springer. ISBN 978-3-319-30338-3.
5. Marek, M., & Schreiber, I. (1995). *Chaotic behaviour of deterministic dissipative systems* (Vol. 1). Cambridge University Press. ISBN: 0-521-43830-6.
6. Schuster, H. G., & Just, W. (2006). *Deterministic chaos: An introduction.* Wiley. ISBN 978-3-527-40415-5.
7. Iooss, G., Helleman, R. H., & Stora, R. (1983). *Chaotic behaviour of deterministic systems* (Vol. 16). Les Houches. ISBN 0-444-86542-X.
8. Thompson, J. M. T., & Stewart, H. B. (2002). *Nonlinear dynamics and chaos.* Wiley. ISBN 0-471-87645-3.
9. Vaidyanathan, S. (2016). Global chaos control of the FitzHugh-Nagumo chaotic neuron model via integral sliding mode control. *International Journal of PharmTech Research, 9*(4), 413–425.
10. Vaidyanathan, S. (2016). Anti-synchronization of enzymes-substrates biological systems via adaptive backstepping control. *International Journal of PharmTech Research, 9*(2), 193–205.
11. Vaidyanathan, S. (2015). Anti-synchronization of chemical chaotic reactors via adaptive control method. *International Journal of ChemTech Research, 8*(8), 73–85.
12. Vaidyanathan, S. (2015). Global chaos synchronization of chemical chaotic reactors via novel sliding mode control method. *International Journal of ChemTech Research, 8*(7), 209–221.
13. Vaidyanathan, S. (2015). A novel chemical chaotic reactor system and its output regulation via integral sliding mode control. *International Journal of ChemTech Research, 8*(11), 669–683.
14. Vaidyanathan, S. (2015). Integral sliding mode control design for the global chaos synchronization of identical novel chemical chaotic reactor systems. *International Journal of ChemTech Research, 8*(11), 684–699.

15. Meng, X., Xiang, W., Jia, L., & Xu, J. (2015). Train flow chaos analysis based on an improved automata model. *Chaos, Solitons and Fractals, 81*, 43–51.
16. Vaidyanathan, S. (2015). Dynamics and control of Tokamak system with symmetric and magnetically confined plasma. *International Journal of ChemTech Research, 8*(6), 795–803.
17. Vaidyanathan, S. (2015). Synchronization of Tokamak systems with symmetric and magnetically confined plasma via adaptive control. *International Journal of ChemTech Research, 8*(6), 818–827.
18. Vaidyanathan, S. (2015). Adaptive synchronization of Rikitake two-disk dynamo chaotic systems. *International Journal of ChemTech Research, 8*(8), 100–111.
19. Vaidyanathan, S. (2015). Hybrid chaos synchronization of Rikitake two-disk dynamo chaotic systems via adaptive control method. *International Journal of ChemTech Research, 8*(11), 12–25.
20. Vaidyanathan, S. (2015). Active control design for the hybrid chaos synchronization of Lotka-Volterra biological systems with four competitive species. *International Journal of PharmTech Research, 8*(8), 30–42.
21. Vaidyanathan, S. (2015). Adaptive biological control of generalized Lotka-Volterra three-species biological system. *International Journal of PharmTech Research, 8*(4), 622–631.
22. Vaidyanathan, S. (2015). Sliding controller design for the global chaos synchronization of forced Van der Pol chaotic oscillators. *International Journal of PharmTech Research, 8*(7), 100–111.
23. Vaidyanathan, S. (2015). Output regulation of the forced Van der Pol chaotic oscillator via adaptive control method. *International Journal of PharmTech Research, 8*(6), 106–116.
24. Karthikeyan, R., & Sundarapandian, V. (2014). Hybrid chaos synchronization of four-scroll systems via active control. *Journal of Electrical Engineering, 65*(2), 97–103.
25. Pehlivan, I., Moroz, I. M., & Vaidyanathan, S. (2014). Analysis, synchronization and circuit design of a novel butterfly attractor. *Journal of Sound and Vibration, 333*(20), 5077–5096.
26. Akgul, A., Moroz, I., Pehlivan, I., & Vaidyanathan, S. (2016). A new four-scroll chaotic attractor and its engineering applications. *Optik, 127*(13), 5491–5499.
27. Vaidyanathan, S., & Azar, A. T. (2016). A novel 4-D four-wing chaotic system with four quadratic nonlinearities and its synchronization via adaptive control method. In *Advances in chaos theory and intelligent control. Studies in fuzziness and soft computing* (Vol. 337, pp. 203–224). Berlin, Germany: Springer.
28. Vaidyanathan, S., & Azar, A. T. (2016). Adaptive control and synchronization of Halvorsen circulant chaotic systems. In *Advances in chaos theory and intelligent control. Studies in fuzziness and soft computing* (Vol. 337, pp. 225–247). Berlin, Germany: Springer.
29. Vaidyanathan, S., & Azar, A. T. (2016). Dynamic analysis, adaptive feedback control and synchronization of an eight-term 3-D novel chaotic system with three quadratic nonlinearities. In *Advances in chaos theory and intelligent control. Studies in fuzziness and soft computing* (Vol. 337, pp. 155–178). Berlin, Germany: Springer.
30. Vaidyanathan, S., & Azar, A. T. (2016). Generalized projective synchronization of a novel hyperchaotic four-wing system via adaptive control method. In *Advances in chaos theory and intelligent control. Studies in fuzziness and soft computing* (Vol. 337, pp. 275–296). Berlin, Germany: Springer.
31. Vaidyanathan, S., & Azar, A. T. (2016). Qualitative study and adaptive control of a novel 4-D hyperchaotic system with three quadratic nonlinearities. In *Advances in chaos theory and intelligent control. Studies* in *fuzziness and soft computing* (Vol. 337, pp. 179–202). Berlin, Germany: Springer.
32. Vaidyanathan, S., & Azar, A. T. (2016). Adaptive backstepping control and synchronization of a novel 3-D jerk system with an exponential nonlinearity. In: *Advances in chaos theory and intelligent control. Studies in fuzziness and soft computing* (Vol. 337, pp. 249–274). Berlin, Germany: Springer.

33. Sampath, S., & Vaidyanathan, S. (2016). Hybrid synchronization of identical chaotic systems via novel sliding control method with application to Sampath four-scroll chaotic system. *International Journal of Control Theory and Applications, 9*(1), 221–235.
34. Vaidyanathan, S., Sampath, S., & Azar, A. T. (2015). Global chaos synchronisation of identical chaotic systems via novel sliding mode control method. *International Journal of Modelling, Identification and Control, 23*(1), 92–100.
35. Boulkroune, A., Bouzeriba, A., Bouden, T., & Azar, A. T. (2016). *fuzzy adaptive synchronization of uncertain fractional-order chaotic systems, advances in chaos theory and intelligent control, studies in fuzziness and soft computing* (Vol. 337, pp. 681–697). Germany: Springer.
36. Vaidyanathan, S., & Azar, A. T. (2016). Takagi-Sugeno fuzzy logic controller for Liu-Chen four-scroll chaotic system. *International Journal of Intelligent Engineering Informatics, 4*(2), 135–150.
37. Adachi, M., & Aihara, K. (1997). Associative dynamics in a chaotic neural network. *Neural Networks, 10*(1), 83–98.
38. Hoppensteadt, F. C. (2013). *Analysis and simulation of chaotic systems, applied mathematical sciences* (Vol. 94). Berlin, Germany: Springer.
39. Hajihassani, M., Armaghani, D. J., Marto, A., & Mohamad, E. T. (2015). Ground vibration prediction in quarry blasting through an artificial neural network optimized by imperialist competitive algorithm. *Bulletin of Engineering Geology and the Environment, 74*(3), 873–886.
40. Wasserman, P. D. (1993). *Advanced methods in neural computing*. New Jersey, USA: Wiley.
41. Amiri, A., Niaki, S. T. A., & Moghadam, A. T. (2015). A probabilistic artificial neural network-based procedure for variance change point estimation. *Soft Computing, 19*(3), 691–700.
42. Maind, S. B., & Wankar, P. (2014). Research paper on basic of artificial neural network. *International Journal on Recent and Innovation Trends in Computing and Communication, 2* (1), 96–100.
43. Mekki, H., Mellit, A., & Salhi, H. (2016). Artificial neural network-based modelling and fault detection of partial shaded photovoltaic modules. *Simulation Modelling Practice and Theory, 67*, 1–13.
44. Dou, J., Yamagishi, H., Pourghasemi, H. R., Yunus, A. P., Song, X., Xu, Y., et al. (2015). An integrated artificial neural network model for the landslide susceptibility assessment of Osado Island. *Japan Natural Hazards, 78*(3), 1749–1776.
45. Wang, T., Gao, H., & Qiu, J. (2016). A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. *IEEE Transactions on Neural Networks and Learning Systems, 27*(2), 416–425.
46. Grézl, F., Karafiát, M., & Veselỳ, K. (2014). Adaptation of multilingual stacked bottle-neck neural network structure for new language. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 7654–7658). IEEE.
47. Nickl-Jockschat, T., Rottschy, C., Thommes, J., Schneider, F., Laird, A. R., Fox, P. T., et al. (2015). Neural networks related to dysfunctional face processing in autism spectrum disorder. *Brain Structure and Function, 220*(4), 2355–2371.
48. Azar, A. T. (2013). Fast neural network learning algorithms for medical applications. *Neural Computing and Applications, 23*(3–4), 1019–1034.
49. Azar, A. T., & El-Said, S. A. (2013). Probabilistic neural network for breast cancer classification. *Neural Computing and Applications, 23*(6), 1737–1751.
50. Hsu, C.-T., Korimara, R., Tsai, L.-J., & Cheng, T.-J. (2016). Photovoltaic power generation system modeling using an artificial neural network. In J. Juang (Ed.), *Proceedings of the 3rd International Conference on Intelligent Technologies and Engineering Systems (ICITES2014)* (pp. 365–371). Cham: Springer.
51. Huang, X., Zhao, Z., Wang, Z., & Li, Y. (2012). Chaos and hyperchaos in fractional-order cellular neural networks. *Neurocomputing, 94*, 13–21.
52. Vaidyanathan, S. (2015). 3-cells cellular neural network (CNN) attractor and its adaptive biological control. *International Journal of PharmTech Research, 8*(4), 632–640.

53. Vaidyanathan, S. (2015). Synchronization of 3-cells cellular neural network (CNN) attractors via adaptive control method. *International Journal of PharmTech Research, 8*(5), 946–955.
54. Vaidyanathan, S. (2015). Anti-synchronization of 3-cells cellular neural network attractors via adaptive control method. *International Journal of PharmTech Research, 8*(7), 26–38.
55. Vaidyanathan, S. (2016). Anti-synchronization of 3-cells cellular neural network attractors via integral sliding mode control. *International Journal of PharmTech Research, 9*(1), 193–205.
56. Chen, C.-W. (2014). Applications of neural-network-based fuzzy logic control to a nonlinear time-delay chaotic system. *Journal of Vibration and Control, 20*(4), 589–605.
57. Zhang, G., & Shen, Y. (2014). Exponential synchronization of delayed memristor-based chaotic neural networks via periodically intermittent control. *Neural Networks, 55*, 1–10.
58. Pham, V. T., Volos, C., Jafari, S., Wang, X., & Vaidyanathan, S. (2014). Hidden hyperchaotic attractor in a novel simple memristive neural network. *Optoelectronics and Advanced Materials, Rapid Communications, 8*(11–12), 1157–1163.
59. Wen, S., Zeng, Z., Huang, T., & Zhang, Y. (2014). Exponential adaptive lag synchronization of memristive neural networks via fuzzy method and applications in pseudorandom number generators. *IEEE Transactions on Fuzzy Systems, 22*(6), 1704–1713.
60. Kyprianidis, I. M., & Makri, A. T. (2013). Complex dynamics of FitzHugh-Nagumo type neurons coupled with gap junction under external voltage stimulation. *Journal of Engineering Science and Technology Review, 6*(4), 104–114.
61. He, G., Cao, Z., Zhu, P., & Ogura, H. (2003). Controlling chaos in a chaotic neural network. *Neural Networks, 16*(8), 1195–1200.
62. Ramesh, M., & Narayanan, S. (2001). Chaos control of Bonhoeffer–van der Pol oscillator using neural networks. *Chaos, Solitons and Fractals, 12*(13), 2395–2405.
63. Ren, X. M., Rad, A. B., Chan, P. T., & Lo, W. L. (2003). Identification and control of continuous-time nonlinear systems via dynamic neural networks. *IEEE Transactions on Industrial Electronics, 50*(3), 478–486.
64. Gan, Q., & Liang, Y. (2012). Synchronization of chaotic neural networks with time delay in the leakage term and parametric uncertainties based on sampled-data control. *Journal of the Franklin Institute, 349*(6), 1955–1971.
65. Lin, D., & Wang, X. (2011). Self-organizing adaptive fuzzy neural control for the synchronization of uncertain chaotic systems with random-varying parameters. *Neurocomputing, 74*(12), 2241–2249.
66. Zhang, H., Ma, T., Huang, G.-B., & Wang, Z. (2010). Robust global exponential synchronization of uncertain chaotic delayed neural networks via dual-stage impulsive control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *40*(3), 831–844.
67. Ardalani-Farsa, M., & Zolfaghari, S. (2010). Chaotic time series prediction with residual analysis method using hybrid Elman–NARX neural networks. *Neurocomputing, 73*(13), 2540–2553.
68. Chen, L., & Aihara, K. (1995). Chaotic simulated annealing by a neural network model with transient chaos. *Neural Networks, 8*(6), 915–930.
69. Mou, C., Jiang, C., Bin, J., & Wu, Q. (2009). Sliding mode synchronization controller design with neural network for uncertain chaotic systems. *Chaos, Solitons and Fractals, 39*(4), 1856–1863.
70. Chen, M., & Chen, W. (2009). Robust adaptive neural network synchronization controller design for a class of time delay uncertain chaotic systems. *Chaos, Solitons and Fractals, 41*(5), 2716–2724.
71. Lamamra, K., Belarbi, K., & Boukhtini, S. (2015). Box and Jenkins nonlinear system modelling using RBF neural networks designed by NSGAII. In *Computational Intelligence Applications in Modeling and Control* (pp. 229–254). Springer.
72. Sellnow, T. L., Seeger, M. W., & Ulmer, R. R. (2002). Chaos theory, informational needs, and natural disasters. *Journal of Applied Communication Research, 30*(4), 269–292.
73. Ata, N. A. A., & Schmandt, R. (2016). Systemic and systematic Risk. *The tyranny of uncertainty* (pp. 57–62). Berlin Heidelberg: Springer.

74. Elhadj, Z. (2006). *Etude de quelques types de systemes chaotiques : Generalisation d'un modele issu du modele de chen*. Doctoral thesis, University of Constantine.
75. Hecht-Nielsen, R. (1989). Theory of the backpropagation neural network. In *IEEE International Joint Conference on Neural Networks (IJCNN-1989)* (pp. 593–605). IEEE.
76. Yadav, N., Yadav, A., & Kumar, M. (2015). *An Introduction to Neural Network Methods for Differential Equations*. Berlin, Germany: Springer.
77. Lamamra, K., & Belarbi, K. (2011). Comparison of neural networks and fuzzy logic control designed by multi-objective genetic algorithm. *IJACT: International Journal of Advancements in Computing Technology*, *3*(4), 137–143.
78. Sajikumar, N., & Thandaveswara, B. S. (1999). A non-linear rainfall–runoff model using an artificial neural network. *Journal of Hydrology, 216*(1), 32–55.
79. Suykens, J. A. (2001). Support vector machines: A nonlinear modelling and control perspective. *European Journal of Control, 7*(2), 311–327.
80. Zhu, Q., & Azar A. T. (2015). *Complex system modelling and control through intelligent soft computations*. *Studies in fuzziness and soft computing* (Vol. 319). Berlin, Germany: Springer. ISBN: 978-3-319-12883-2.
81. Graupe, D. (2013). *Principles of artificial neural networks* (Vol. 7). World Scientific. ISBN 978-981-4522-73-1.
82. Dong, L., Wesseloo, J., Potvin, Y., & Li, X. (2016). Discrimination of mine seismic events and blasts using the Fisher classifier, Naive Bayesian classifier and logistic regression. *Rock Mechanics and Rock Engineering, 49*(1), 183–211.
83. Greenhalgh, J., & Mirmehdi, M. (2015). Automatic detection and recognition of symbols and text on the road surface. In *International Conference on Pattern Recognition Applications and Methods* (pp. 124–140). Springer.
84. Greenhalgh, J., & Mirmehdi, M. (2015). Detection and recognition of painted road surface markings. In *Proceedings of 4th International Conference on Pattern Recognition Applications and Methods* (pp. 130–138).
85. Lamamra, K., Belarbi, K., & Mokhtari, F. (2006). Optimization of the structure of a neural networks by multi-objective genetic algorithms. *Proceedings of the ICGST International Journal on Automation, Robotics and Autonomous Systems*, *6*(1), 1–4.
86. Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms* (Vol. 16). Wiley. ISBN 0-471-87339-X.
87. Lamamra, K., Belarbi, K., Belhani, A., & Boukhtini, S. (2014). NSGA2 based of multi-criteria decision analysis for multi-objective optimization of fuzzy logic controller for non linear system. *Journal of Next Generation Information Technology, 5*(1), 57–64.
88. Rao, R. V., Rai, D. P., & Balic, J. (2016). Multi-objective optimization of machining and micro-machining processes using non-dominated sorting teaching–learning-based optimization algorithm. *Journal of Intelligent Manufacturing*, 1–23.
89. Sardou, I. G., & Ameli, M. T. (2016). A fuzzy-based non-dominated sorting genetic algorithm-II for joint energy and reserves market clearing. *Soft Computing, 20*(3), 1161–1177.
90. Wong, J. Y., Sharma, S., & Rangaiah, G. P. (2016). Design of shell-and-tube heat exchangers for multiple objectives using elitist non-dominated sorting genetic algorithm with termination criteria. *Applied Thermal Engineering, 93*, 888–899.
91. Yang, M.-D., Lin, M.-D., Lin, Y.-H., & Tsai, K.-T. (2016). Multiobjective optimization design of green building envelope material using a non-dominated sorting genetic algorithm. *Applied Thermal Engineering.* In Press.
92. Srinvas, N., & Deb, K. (1994). Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation, 2*(3), 221–248.
93. Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International Conference on Parallel Problem Solving From Nature* (pp. 849–858). Springer.
94. Versèle, C., Deblecker, O., & Lobry, J. (2011). Multiobjective optimal design of an inverter fed axial flux permanent magnet in-wheel motor for electric vehicles. *Electric Vehicles-Modelling And Simulations, 287*, 300–321.

95. Mackey, M. C., & Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science, 197*(4300), 287–289.
96. Anderson, D., Sweeney, D., Williams, T., Camm, J., & Martin, R. (2011). *an introduction to management science: Quantitative approaches to decision making*. Revised: Cengage Learning.
97. Martinetz, T. M., Berkovich, S. G., & Schulten, K. J. (1993). Neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks, 4*(4), 558–569.