Research Article

Mathematical Methods in the Applied Sciences

Received 2 September 2013

Published online in Wiley Online Library

(wileyonlinelibrary.com) DOI: 10.1002/mma.3064
MOS subject classification: 34D08; 34C28; G.1.7

Estimating 2-cycle fixed points of henon map using backpropagation neural networks

Athanasios Margaris[a*†], Konstantinos Goulianas[b], Konstantinos Diamantaras[b] and Theophilos Papadimitriou[c]

Communicated by T. Monovasilis

The objective of this research is the presentation of a feed-forward neural network capable of estimating the 2-cycle fixed points of Henon map by solving their defining nonlinear algebraic system. The network uses the back propagation algorithm and solves the aforementioned system for a set of values of the parameters α and β of Henon map. Besides the estimation of the fixed points, the paper includes the study of the network convergence and its speed for many different initial conditions. Copyright © 2013 John Wiley & Sons, Ltd.

Keywords: neural networks; Henon map; nonlinear algebraic systems; fixed points

1. Introduction

Artificial neural networks are used among other tasks, to solve algebraic systems of linear and nonlinear equations. Typical references in this field include the work of Mathia and Saeks [1] that used recurrent neural networks to solve nonlinear equations, Mishra and Kalra [2] that used a modified Hopfield network to solve a nonlinear algebraic system of m equations with n unknowns, Luo and Han [3] that used Hopfield networks to solve a nonlinear system transformed to a kind of quadratic optimization, and Li and Zeng [4] that used the gradient descent rule with a variable step size to solve such systems at very rapid convergence and very high accuracy. It has to be mentioned that besides the identification of fixed points of chaotic maps, artificial neural networks are used among other things for modeling ([5] and [6]) as well as the control ([7] and [8]) of chaotic attractors.

In this paper, a back-propagation fully connected feed-forward neural network is used to estimate the 2-cycle fixed points of the Henon map ([9] and [10]), a well-known dynamical system that depending on its parameter values, exhibits stable, periodic, as well as chaotic behavior. This network solves the associated nonlinear algebraic system for different combinations of the values α and β, and its convergence is studied for a set of different initial conditions. The next sections include a short description of the Henon map, a detailed presentation of the associated neural network, and the description of the application of this network for the estimation of the 2-cycle fixed points of that map. The paper is concluded with presentation and analysis of typical experimental results.

2. Henon map

A typical and well-defined class of chaotic systems includes the so-called chaotic maps that exhibit chaotic behavior. In this work, the focus is given to discrete chaotic maps defined in the discrete time domain and described by iterative functions. If such a map is defined in an N dimensional state space, its mathematical description is given by a recursive equation in the form $\boldsymbol{r}_{n+1} = \boldsymbol{F}(\boldsymbol{r}_n, \boldsymbol{\xi})$, where $\boldsymbol{r}_n = \{x_1, x_2, \ldots, x_N\}$ is the state vector of the system and $\boldsymbol{\xi} = \{\alpha, \beta, \gamma, \ldots\}$ is an optional parameter vector that appears in the map's defining equations.

One of the best known and commonly used chaotic maps is the Henon map [9, 11], a two-dimensional map described by the system of recursive equations

$$x_{n+1} = y_n - \alpha x_n^2 + 1$$
$$y_{n+1} = \beta x_n$$

[a] *Department of Applied Informatics, University of Macedonia, Greece (also a member of the University of Macedonia Neural Group)*
[b] *Information Technology Department, TEI of Thessaloniki, 57400, Thessaloniki, Greece*
[c] *Department of Economics, Democritus University of Thrace, Greece*
**Correspondence to: Athanasios Margaris, University of Macedonia, Department of Applied Informatics, Greece.*
[†] *E-mail: amarg@uom.gr*

The Henon map depends on two parameters α and β; these parameters, for the canonical map that exhibits chaotic behavior, have the values $\alpha = 1.4$ and $\beta = 0.3$. On the other hand, for other values of those parameters, this map may be chaotic, intermittent, or converging to a single periodic orbit.

To identify the m-cycle fixed points of the Henon map, one has to solve the equation $\boldsymbol{p} = \boldsymbol{F}^{(m)}(\boldsymbol{p}, \boldsymbol{\xi})$ where $\boldsymbol{F}$ is the mathematical function describing the map and $\boldsymbol{p}$ is a point in the form $\boldsymbol{p} = (x, y)$. It can be easily proven that, for the case $m = 1$, there are two saddle fixed points (x_1, y_1) and (x_2, y_2) that can be estimated by solving the system of equations

$$\alpha x^2 + x - y = 1 \quad \text{and} \quad \beta x - y = 0$$

and their coordinates are given by the expressions

$$\begin{aligned} x_1 &= \frac{1}{2\alpha}\left[\beta - 1 + \sqrt{(1-\beta)^2 + 4\alpha}\right] \\ y_1 &= \frac{\beta}{2\alpha}\left[\beta - 1 + \sqrt{(1-\beta)^2 + 4\alpha}\right] \\ x_2 &= \frac{1}{2\alpha}\left[\beta - 1 - \sqrt{(1-\beta)^2 + 4\alpha}\right] \\ y_2 &= \frac{\beta}{2\alpha}\left[\beta - 1 - \sqrt{(1-\beta)^2 + 4\alpha}\right] \end{aligned}$$

These points appear only for the parameter region defined as $\alpha > (\beta - 1)^2/4$, while for the value $\alpha = (\beta - 1)^2/4$, a saddle node bifurcation occurs, leading to a double root defined by the equation $x = x_1 = x_2 = 2/(1 - \beta)$. On the other hand, the 2-cycle points can be estimated by solving the system of equations

$$\alpha^3 x^4 - 2\alpha^2 x^2 + (1-\beta)^3 x - (1-\beta)^2 + \alpha = 0$$

$$y = \frac{\beta}{1-\beta}(1 - \alpha x^2)$$

This system has four real roots for parameter values in the region $\alpha > 3(\beta - 1)^2/4$. Regarding these roots, two of them, are the 1-cycle fixed points identified in the previous step, while, the remaining two roots can be obtained by solving the equations

$$\alpha^2 x^2 - \alpha(1-\beta)x + (1-\beta)^2 - \alpha = 0 \tag{1}$$

$$y = \frac{\beta}{1-\beta}\left(1 - \alpha x^2\right) \tag{2}$$

The roots of the first equation have the form

$$\begin{aligned} x_1 &= \frac{1}{2\alpha}\left[1 - \beta + \sqrt{4\alpha - 3(1-\beta)^2}\right] \quad \text{and} \\ x_2 &= \frac{1}{2\alpha}\left[1 - \beta - \sqrt{4\alpha - 3(1-\beta)^2}\right] \end{aligned}$$

as it can very easily be identified. Then, by substituting these values to the second equation, the associated y values can be estimated.

For the parameter values $\alpha = 1.4$ and $\beta = 0.3$, the coordinates of the 1-cycle fixed points have the values

$$\begin{aligned} (x_1, y_1) &= (+0.631354, +0.189406) \\ (x_2, y_2) &= (-1.131354, -0.339406) \end{aligned}$$

while the coordinates of the 2-cycle fixed points are estimated as follows

$$\begin{aligned} (x_1, y_1) &= (+0.631354, +0.189406) \\ (x_2, y_2) &= (-1.131354, -0.339406) \\ (x_3, y_3) &= (-0.475800, +0.292739) \\ (x_4, y_4) &= (+0.975800, -0.142739) \end{aligned}$$

3. The neural model

The theoretical foundation for solving 2×2 complete nonlinear algebraic systems in the form $F_1(x,y) = F_2(x,y) = 0$, where

$$F_1(x,y) = \alpha_{11}x^2 + \alpha_{12}y^2 + \alpha_{13}xy + \alpha_{14}x + \alpha_{15}y - \alpha_{16}$$
$$F_2(x,y) = \alpha_{21}x^2 + \alpha_{22}y^2 + \alpha_{23}xy + \alpha_{24}x + \alpha_{25}y - \alpha_{26}$$

can be found in [12], while an improved model of this network can be found in [13]. The main idea behind this approach is the construction of a neural network composed of summation and product units, which simulates exactly the nonlinear system under consideration and find its roots via the classical back propagation approach. In a more detailed description, this network is composed of an input layer with only one neuron with a constant input of unity, a hidden layer with two summation units associated with the first-order terms x and y, a second hidden layer that generates all the linear as well as the nonlinear terms of the system, and an output layer that produces as real outputs, the values of the expressions $F_1(x,y)$ and $F_2(x,y)$. It is clear that if the desired outputs of the network have a zero value, then, the variable weights of the synapses that join the input neuron with the two neurons of the first hidden layer will contain the components of one of the system roots. To estimate all roots of the system (the current version of the simulator supports only real and not complex roots), we have to use different initial conditions, a procedure that allows the experimental estimation of the basin of attraction for each system root. The activation function of the network neurons is the identity function for the input and hidden neurons—an exception to this rule is the top neuron of the second hidden layer that uses the function $f(x) = x^2$—and the hyperbolic tangent function for the output neurons.

It can be very easily seen that the equations (1) and (2) that define the 2-cycle fixed points of the Henon map can be written as a 2×2 nonlinear system in the form

$$F_1(x,y) = \alpha^2x^2 - \alpha(1-\beta)x + (1-\beta)^2 - \alpha = 0 \tag{3}$$

$$F_2(x,y) = \alpha\beta x^2 + (1-\beta)y - \beta = 0 \tag{4}$$

This system is a special case of the 2×2 complete nonlinear algebraic system defined earlier. To solve this system, a new neural network architecture is proposed; this architecture is presented in Figure 1, with the weight matrices $\boldsymbol{W}_{12}$, $\boldsymbol{W}_{23}$, and $\boldsymbol{W}_{34}$ between consecutive layers to have the form

$$\boldsymbol{W}_{12} = \begin{bmatrix} \boldsymbol{W}_{12}^{(1)} & \boldsymbol{W}_{12}^{(2)} \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix}$$
$$\boldsymbol{W}_{23} = \begin{bmatrix} \boldsymbol{W}_{23}^{(11)} & \boldsymbol{W}_{23}^{(12)} & \boldsymbol{W}_{23}^{(13)} \\ \boldsymbol{W}_{23}^{(21)} & \boldsymbol{W}_{23}^{(22)} & \boldsymbol{W}_{23}^{(23)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$\boldsymbol{W}_{34} \begin{bmatrix} \boldsymbol{W}_{34}^{(11)} & \boldsymbol{W}_{34}^{(12)} \\ \boldsymbol{W}_{34}^{(21)} & \boldsymbol{W}_{34}^{(22)} \\ \boldsymbol{W}_{34}^{(31)} & \boldsymbol{W}_{34}^{(32)} \end{bmatrix} = \begin{bmatrix} \alpha^2 & \alpha\beta \\ \alpha(\beta-1) & 0 \\ 0 & 1-\beta \end{bmatrix}$$

By applying the classical back propagation algorithm, we obtain the following results:

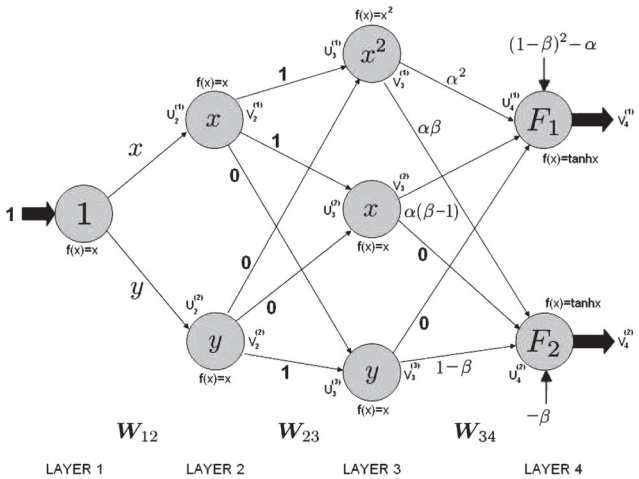

Figure 1. The structure of the neural system that estimates the 2-cycle fixed points of the Henon map.

3.1. *Forward pass*

The inputs to the second hidden layer are defined as

$$\upsilon_2^{(1)} = \boldsymbol{W}_{12}^{(1)} = x$$
$$\upsilon_2^{(2)} = \boldsymbol{W}_{12}^{(2)} = y$$

while the corresponding outputs are given by the equations

$$\nu_2^{(1)} = x \quad \text{and} \quad \nu_2^{(2)} = y$$

In a similar way, the third layer obtains the inputs

$$\upsilon_3^{(1)} = \boldsymbol{W}_{23}^{(11)}\nu_2^{(1)} + \boldsymbol{W}_{23}^{(21)}\nu_2^{(2)} = x$$
$$\upsilon_3^{(2)} = \boldsymbol{W}_{23}^{(12)}\nu_2^{(1)} + \boldsymbol{W}_{23}^{(22)}\nu_2^{(2)} = x$$
$$\upsilon_3^{(3)} = \boldsymbol{W}_{23}^{(13)}\nu_2^{(1)} + \boldsymbol{W}_{23}^{(23)}\nu_2^{(2)} = y$$

and produces the outputs

$$\nu_3^{(1)} = x^2, \ \nu_3^{(2)} = x, \ \nu_3^{(3)} = y$$

Finally, the fourth (output) layer uses the inputs

$$\upsilon_4^{(1)} = \alpha^2x^2 - \alpha(1-\beta)x + (1-\beta)^2 - \alpha = F_1(x,y)$$
$$\upsilon_4^{(2)} = \alpha\beta x^2 + (1-\beta)y - \beta = F_2(x,y)$$

to generate the network output values

$$\nu_4^{(1)} = o_1 = \tanh\left[\alpha^2x^2 - \alpha(1-\beta)x + (1-\beta)^2 - \alpha\right]$$
$$\nu_4^{(2)} = o_2 = \tanh\left[\alpha\beta x^2 + (1-\beta)y - \beta\right]$$

where $f(x) = \tanh(x)$ is the activation function of the output neurons. It can be easily verified that $\nu_4^{(1)} = f[F_1(x,y)]$ and $\nu_4^{(2)} = f[F_2(x,y)]$.

3.2. *Backward pass*

In this stage, the delta values associated with the neurons of the output layer and the two hidden layers are estimated as follows—in the following description for the sake of simplicity, we use the abbreviations $F_1 = F_1(x,y)$, $F_2 = F_2(x,y)$, and $f = f(x) = \tanh(x)$.

The delta values for the two output neurons are estimated as

$$\delta_4^{(1)} = (d_1 - o_1)\tanh'\left(\upsilon_4^{(1)}\right) = -o_1f'([F_1(x,y)]) = -f(F_1)\left[1 - f^2(F_1)\right]$$
$$\delta_4^{(2)} = (d_2 - o_2)\tanh'\left(\upsilon_4^{(2)}\right) = -o_2f'([F_2(x,y)]) = -f(F_2)\left[1 - f^2(F_2)\right]$$

while, in a similar way, the delta values for the neurons of the third layer have the form

$$\delta_3^{(1)} = f'\left[\upsilon_3^{(1)}\right]\sum_{j=1}^{2}\boldsymbol{W}_{34}^{(1j)}\delta_4^{(j)} = -2x\left\{\alpha^2\left[f(F_1)\left\{1 - f^2(F_1)\right\}\right] + \alpha\beta\left[f(F_2)\left\{1 - f^2(F_2)\right\}\right]\right\}$$

$$\delta_3^{(2)} = f'\left[\upsilon_3^{(2)}\right]\sum_{j=1}^{2}\boldsymbol{W}_{34}^{(2j)}\delta_4^{(j)} = \alpha(1-\beta)f(F_1)\left\{1 - f^2(F_1)\right\}$$

$$\delta_3^{(3)} = f'\left[\upsilon_3^{(3)}\right]\sum_{j=1}^{2}\boldsymbol{W}_{34}^{(3j)}\delta_4^{(j)} = (\beta-1)f(F_2)\left\{1 - f^2(F_2)\right\}$$

Finally, the delta values of the two neurons of the second layer are given by the equations

$$\delta_2^{(1)} = f'\left[\upsilon_2^{(1)}\right]\sum_{j=1}^{3} \boldsymbol{W}_{23}^{(1j)}\delta_3^{(j)} = \delta_3^{(1)} + \delta_3^{(2)} =$$
$$= -f(F_1)\left[1-f^2(F_1)\right]\left[2\alpha^2 x + \alpha(\beta-1)\right] - -f(F_2)\left[1-f^2(F_2)\right](2\alpha\beta x)$$
$$\delta_2^{(2)} = f'\left[\upsilon_2^{(2)}\right]\sum_{j=1}^{3} \boldsymbol{W}_{23}^{(2j)}\delta_3^{(j)} = \delta_3^{(3)} = -f(F_2)\left[1-f^2(F_2)\right](1-\beta)$$

3.3. *Correction of the synaptic weights*

In this last stage of the back propagation algorithm, the two variable weights of the synapses between the first and the second layer are updated according to the equations

$$\boldsymbol{W}_{12}^{(1)} = \boldsymbol{W}_{12}^{(1)} + \eta\delta_2^{(1)} \quad \text{and} \quad \boldsymbol{W}_{12}^{(2)} = \boldsymbol{W}_{12}^{(2)} + \eta\delta_2^{(2)}$$

where η is the learning rate and $\delta_2^{(1)}$ and $\delta_2^{(2)}$ are the delta values of the two neurons of the second hidden layer estimated during the backward phase.

3.4. *Convergence analysis*

In this section, we analyze the convergence property of the aforementioned algorithm.

Theorem 1
Let

$$F_1\left(\boldsymbol{W}_{12}^{(1)}, \boldsymbol{W}_{12}^{(2)}\right) = \alpha^2(\boldsymbol{W}_{12}^{(1)})^2 - \alpha(1-\beta)\boldsymbol{W}_{12}^{(1)} + (1-\beta)^2 - \alpha = 0 \tag{5}$$

$$F_2\left(\boldsymbol{W}_{12}^{(1)}, \boldsymbol{W}_{12}^{(2)}\right) = \alpha\beta\left(\boldsymbol{W}_{12}^{(1)}\right)^2 + \alpha(1-\beta)\boldsymbol{W}_{12}^{(2)} - \beta = 0 \tag{6}$$

be the corresponding system that defines the 2-cycle fixed points of the Henon map, as it has been presented in Section 3 [equations (3) and (4)] where (x, y) has been substituted by $\left(\boldsymbol{W}_{12}^{(1)}, \boldsymbol{W}_{12}^{(2)}\right)$. If we apply the aforementioned algorithm and the condition $\alpha > 3(\beta-1)^2/4$ holds, the algorithm converges to the one of the four roots of the system.

Proof
The LMS (Least Mean Square) error of the back propagation algorithm is given by the equation

$$E = \frac{1}{2}\sum_{i=1}^{2}(d_i - o_i)^2 = \frac{1}{2}\sum_{i=1}^{2}\left(-o_i^2\right) = \frac{1}{2}\sum_{i=1}^{2} f^2\left[F_i\left(\boldsymbol{W}_{12}^{(1)}, \boldsymbol{W}_{12}^{(2)}\right)\right] \tag{7}$$

and therefore we have

$$\frac{\partial E}{\partial \boldsymbol{W}_{12}^{(i)}} = \frac{\partial}{\partial \boldsymbol{W}_{12}^{(i)}}\left\{\frac{1}{2}\sum_{i=1}^{2} f^2\left[F_i\left(\boldsymbol{W}_{12}^{(1)}, \boldsymbol{W}_{12}^{(2)}\right)\right]\right\} = \sum_{i=1}^{2} f(F_i)f'(F_i)\left.\frac{\partial F_i}{\partial \boldsymbol{W}_{12}^{(i)}}\right|_{i=1,2} \tag{8}$$

It can be easily proven that the first derivative of the hyperbolic tangent function $f(x) = \tanh(x)$ satisfies the property $f'(x) = 1 - f^2(x)$ and because we have $\boldsymbol{W}_{12}^{(1)} \equiv x$, $\boldsymbol{W}_{12}^{(2)} \equiv y$, and

$$\frac{\partial F_1}{\partial \boldsymbol{W}_{12}^{(1)}} = 2\alpha^2\boldsymbol{W}_{12}^{(1)} + \alpha(\beta-1) \qquad \frac{\partial F_2}{\partial \boldsymbol{W}_{12}^{(1)}} = 2\alpha\beta\boldsymbol{W}_{12}^{(1)} \qquad \frac{\partial F_1}{\partial \boldsymbol{W}_{12}^{(2)}} = 0, \qquad \frac{\partial F_2}{\partial \boldsymbol{W}_{12}^{(2)}} = 1-\beta$$

equation (8) using the aforementioned expressions is expanded as

$$\frac{\partial E}{\partial \boldsymbol{W}_{12}^{(1)}} = f(F_1)\left[1-f^2(F_1)\right]\left[2\alpha^2\boldsymbol{W}_{12}^{(1)} + \alpha(\beta-1)\right] + f(F_2)\left[1-f^2(F_2)\right]\left(2\alpha\beta\boldsymbol{W}_{12}^{(1)}\right) = -\delta_2^{(1)}$$
$$\frac{\partial E}{\partial \boldsymbol{W}_{12}^{(2)}} = f(F_2)\left[1-f^2(F_2)\right](1-\beta) = -\delta_2^{(2)}$$

or in general form

$$\frac{\partial E}{\partial \boldsymbol{W}_{12}^{(i)}} = -\boldsymbol{\delta}_2^{(i)}, \quad i = 1, 2$$

and the weight update equation obtains the form

$$\boldsymbol{W}_{12}^{(i)} = \boldsymbol{W}_{12}^{(i)} + \eta \boldsymbol{\delta}_2^{(i)} = \boldsymbol{W}_{12}^{(i)} - \eta \left. \frac{\partial E}{\partial \boldsymbol{W}_{12}^{(i)}} \right|_{i=1,2}$$

which is the necessary condition for the convergence of the back propagation algorithm that proves the theorem. □

4. Experimental results

The proposed neural network structure was tested for many combinations of Henon map parameters α and β in the region $0.9 \leq \alpha \leq 1.5$ and $-0.5 \leq \beta \leq 0.5$ with a variation step $\Delta\alpha = \Delta\beta = 0.1$. For each pair (α, β), the neural network was trained with a learning rate of $\eta = 0.2$, a maximum number of iterations $N = 10000$, and initial weight values (x, y) in the range $-2 \leq x, y \leq 2$ with variation step $\Delta x = \Delta y = 0.2$. The emphasis was given to classical parameter combinations (α, β) associated with the chaotic region of the Henon map, even though the network can work efficiently in any region (in fact, in this paper, we are not interested in the chaotic features of Henon map but only in the nonlinear algebraic system associated with the 2-cycle points).

Table I contains the experimental results for the parameter values $\alpha = 1.3, 1.4$ and $-0.5 \leq \beta \leq 0.5$. In some cases, the neural network could not converge to some root, which means that the associated pair of initial conditions (x, y) belongs to the basin

Table I. Simulation results for best simulation runs for parameter values $\alpha = 1.3, 1.4$, and $-0.5 \leq \beta \leq 0.5$.

	$\alpha = 1.3$						$\alpha = 1.4$				
b	X_{init}	Y_{init}	X_{root}	Y_{root}	N	b	X_{init}	Y_{init}	X_{root}	Y_{root}	N
−0.5	No	No	No	No	No	−0.5	No	No	No	No	No
	No	No	No	No	No		No	No	No	No	No
−0.4	No	No	No	No	No	−0.4	No	No	No	No	No
	No	No	No	No	No		No	No	No	No	No
−0.3	0.2	−1.0	0.361325	−0.191602	187	−0.3	0.2	−0.2	0.204281	−0.217286	46
	0.6	0.1	0.638675	−0.108397	281		1.4	−0.4	0.724289	−0.061284	77
−0.2	0.1	−0.2	0.100737	−0.164467	44	−0.2	−0.4	−0.2	0.024510	−0.166526	38
	1.6	−0.4	0.822339	−0.020147	40		1.4	−0.2	0.832632	−0.004920	41
−0.1	−0.8	−0.2	−0.058844	−0.090449	49	−0.1	0.4	−0.4	−0.108416	−0.089413	50
	1.0	0.1	0.904998	−0.005844	46		1.0	0.0	0.894131	0.010841	41
0.0	−0.2	0.1	−0.185861	0.0	10	0.0	−0.2	0.0	−0.218732	0.0	4
	1.0	0.0	0.955092	0.0	11		0.8	0.0	0.933018	0.0	5
0.1	−1.0	0.2	−0.293973	0.098628	79	0.1	−0.8	0.2	−0.314446	0.095730	81
	1.4	0.0	0.986281	−0.029397	74		1.2	0.0	0.957303	−0.031444	74
0.2	−0.4	0.2	−0.388875	0.200852	74	0.2	−0.6	0.2	−0.399404	0.194166	77
	1.8	0.2	1.004260	−0.077775	79		1.6	0.4	0.970833	−0.079880	122
0.3	−1.0	0.4	−0.473584	0.303614	124	0.3	0.2	−0.4	−0.475800	0.292739	124
	1.6	0.0	1.012046	−0.142075	138		0.6	−0.2	0.975800	−0.142739	131
0.4	−0.4	0.4	−0.549914	0.404581	158	0.4	−1.4	0.6	−0.545010	0.389432	119
	0.4	−0.4	1.011453	−0.219965	178		0.8	1.2	0.973581	−0.218004	264
0.5	−1.2	0.6	−0.619039	0.501827	259	0.5	No	No	No	No	No
	1.8	−0.2	1.003654	−0.309519	253		No	No	No	No	No

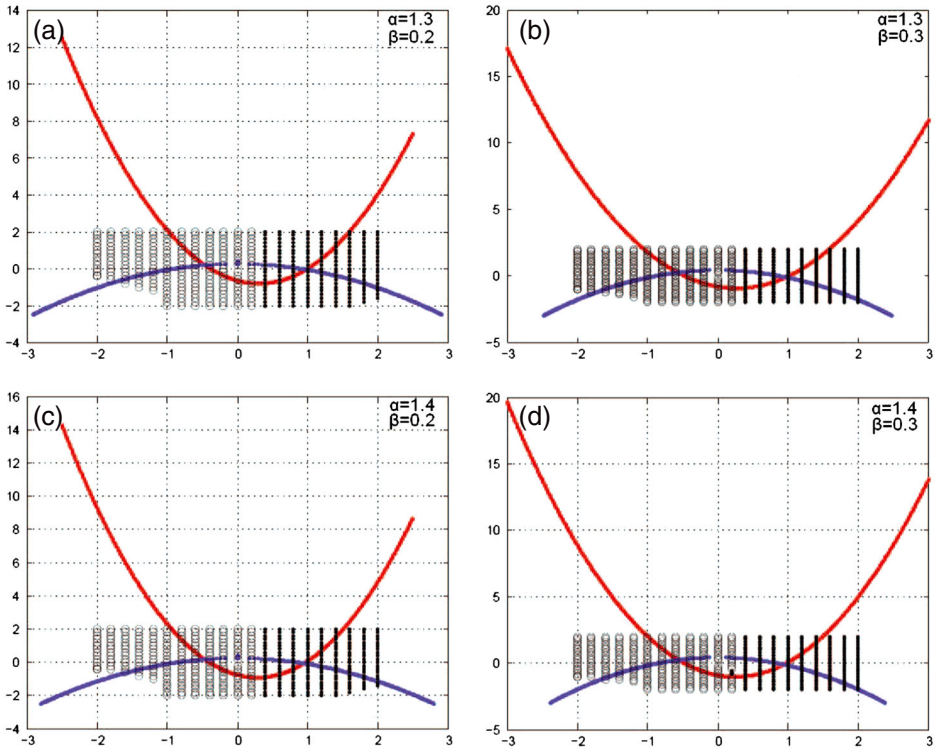

Figure 2. Graphical estimation of system roots and the associated basin of attraction for typical parameter value pairs.

Table II. Simulation results for Henon map and for parameter values $\alpha = 1.4$ and $\beta = 0.4$.

	Average iterations	STDDEV iterations	Average MSE	STDDEV MSE
Root1	484.8283	1116.572	9.39E-16	4.90239E-17
Root2	280.1500	0013.415	9.45E-16	5.03831E-17
	Average Xroot	STDDEV Xroot	Average Yroot	STDDEV Yroot
Root1	−0.54501041	4.93577E-09	+0.389432723	7.2808E-08
Root2	+0.97358184	1.58114E-09	−0.218004091	3.1622E-09

STDDEV stands for Standard Deviation

of divergence. For each parameter combination, the associated row includes the initial conditions $(X_{\text{init}}, Y_{\text{init}})$, the estimated root $(X_{\text{root}}, Y_{\text{root}})$, and the number of iterations associated with this run. In fact, this is the minimum number of iterations required for the estimation of the root $(X_{\text{root}}, Y_{\text{root}})$ for the given parameter value pair (α, β), and for this reason, the results of Table I refer to the best run (with respect to the iteration number). The value of the mean square error (MSE) was practically zero ($\sim 10^{-15}$)—this zero MSE value is a powerful feature of the proposed simulator for all cases—while, the values of $F_1(X_{\text{root}}, Y_{\text{root}})$ and $F_2(X_{\text{root}}, Y_{\text{root}})$ was also too small ($\sim 10^{-8}$) to be presented here. Therefore, the estimated values are identical with the theoretical ones, with the discrepancy between the two values, to occur after the sixth decimal point, or even better. In all cases, the number of different runs for each pair (α, β) and for the specified ranges and variation steps for the variables (x, y) was equal to 441. The graphical representation of roots and the basin of attraction for the cases $(\alpha, \beta) = (1.3, 0.2), (1.3, 0, 3), (1.4, 0.2), (1.4, 0.3)$ are shown in Figure 2(a), 2(b), 2(c), and 2(d), respectively. In this figures, the roots are the intersection points between the red and the blue curves that represent the equations (1) and (2), respectively, and the basin of attraction are denoted with the symbols $\circ$ and $+$ for roots 1 and 2.

Table II contains the experimental results for a typical pair of values, and more specifically, the pair $(\alpha, \beta) = (1.4, 0.4)$; however, all the other pairs are characterized by the same behavior. The system roots for this pair are $(x_1, y_1) = (-0.545010, 0.389432)$ and $(x_2, y_2) = (0.973581, -0.218004)$. For each root, the table includes the average value and the standard deviation of the iteration number, the MSE as well as the estimated root $(X_{\text{root}}, Y_{\text{root}})$. It is interesting to note that the standard deviation of the parameters $(X_{\text{root}}, Y_{\text{root}})$ is practically zero ($\sim 10^{-9}$) and therefore, all the initial conditions lead to the same value with a very high accuracy—up to the sixth decimal point or even better.

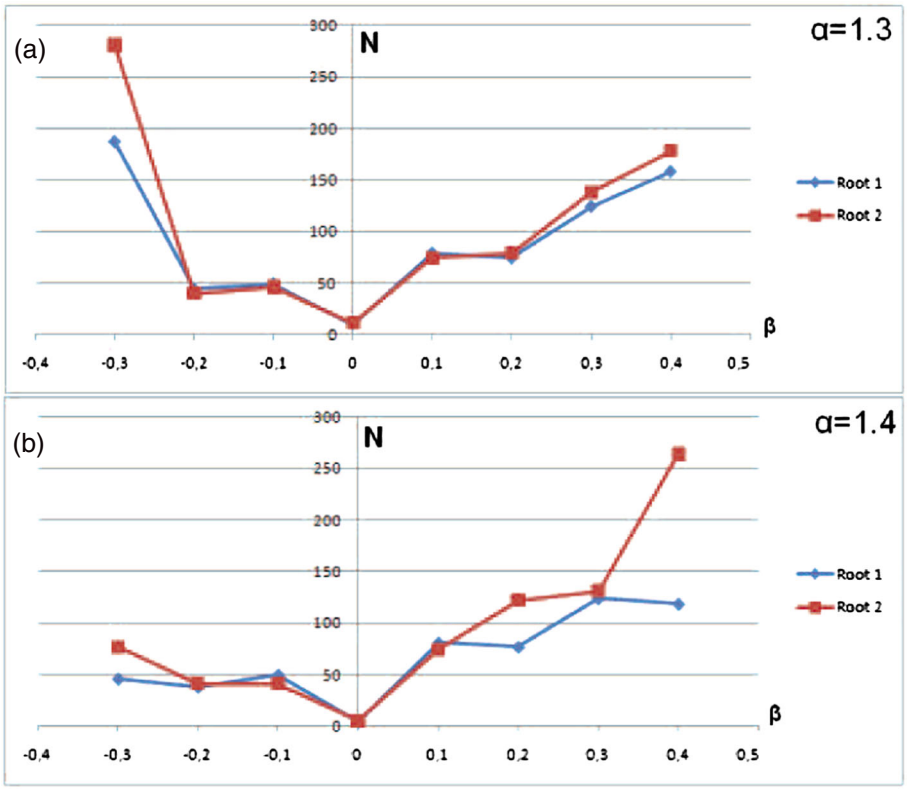

Figure 3. The minimal iteration number (associated with the best run) for the values $\beta = -0.3, -0.2, -0.1, 0.0, 0.1, 0.2, 0.3, 0.4$ and for the case $\alpha = 1.3$ and $\alpha = 1.4$.

Figure 3 shows the variation of the minimal iteration number associated with the best run of the network for the set of values $\beta = -0.3, -0.2, -0.1, 0.0, 0.1, 0.2, 0.3, 0.4$ and for the case $\alpha = 1.3$ [Figure 3(a)] and $\alpha = 1.4$ [Figure 3(b)]. It is clear that the smaller the iteration number, the larger the speed of convergence to the associated root. From this figure, it is clear that the 2-cycle fixed points are identified more easily for small values of the β parameter, with the case $\beta = 0$ to be the best case. This fact is of course expected, because for $\beta = 0$, the equations of the nonlinear system are much simpler and more easily to be solved by the neural system.

Figure 4 represents the percentage of the initial conditions that belong to the basin of attraction for each root, as well as to the basin of divergence. From this figure, it is clear that in all cases, a very large percentage leads to the one of the two available roots; the remaining part of the bar—the black part—is associated with the percentage of points for which the network is not capable of identifying a root; this means that the corresponding initial value pair (x, y) belongs to the basin of infinity.

Figure 5, depicts graphically the sequence of points that eventually converged to some root for the values parameter $\alpha = 1.2, 1.3, 1.4$ and $\beta = 0.2, 0.3, 0.4$, and for the initial conditions $(x, y) = (1.5, -0.5)$ [Figure 5(a)] and $(x, y) = (0.5, 0.5)$ [Figure 5(b)]. From this figure, it is clear that for the same α parameter, the three curves associated with the corresponding values of β are quite similar.

After the presentation of the experimental results associated with the proposed method, let us proceed to a comparison between this method and other well-known numerical methods with respect to their performance. Because the proposed algorithm is an iterative one, we choose for this comparison well-known iterative numerical algorithms such as the trust-region-dogleg [14], the trust-region-reflective [14], and the Levenberg–Marquardt algorithm [15, 16], as well as the multivariate Newton–Raphson method (NR) with partial derivatives. The first three algorithms are supported by the *fsolve* function of the optimization toolbox of the MATLAB programming environment, while the NR MATLAB implementation can be found in the literature—for a theoretical description of this algorithm, see, for example, [17]. These algorithms were used to solve the problem using the initial conditions $(x_{\text{init}}, y_{\text{init}})$ of Table I associated with the best run (to save pages, this comparison is restricted only to the case $\alpha = 1.3$, because its extension to the case $\alpha = 1.4$ is straightforward). We have to note, however, that the NR algorithm could not converge for initial conditions $x_{\text{init}} = 0$ or $y_{\text{init}} = 0$, and for this reason, these initial conditions for this algorithm were set to the value of 0.1.

The simulation results of these comparisons can be found in Table III. For each one of the tested algorithms, the associated row includes the number of iterations required for the convergence to that root (because all algorithms estimated exactly the same roots—at least for the first six decimal points—the coordinates of that roots are included only once). It is not difficult to note that even though the neural solver gave the correct results and with the same accuracy, it requires more iterations than the other methods, because of the small learning rate values that allow the network to converge (it is clear that these results can be improved by testing many different learning rates until to find the best one). Besides this disadvantage of the neural-based approach (which, actually is not a major issue because modern computing systems are characterized by very high speeds), the proposed method is easy in its implementation (because it uses the classical back propagation approach and therefore it computes the function values in the first pass and

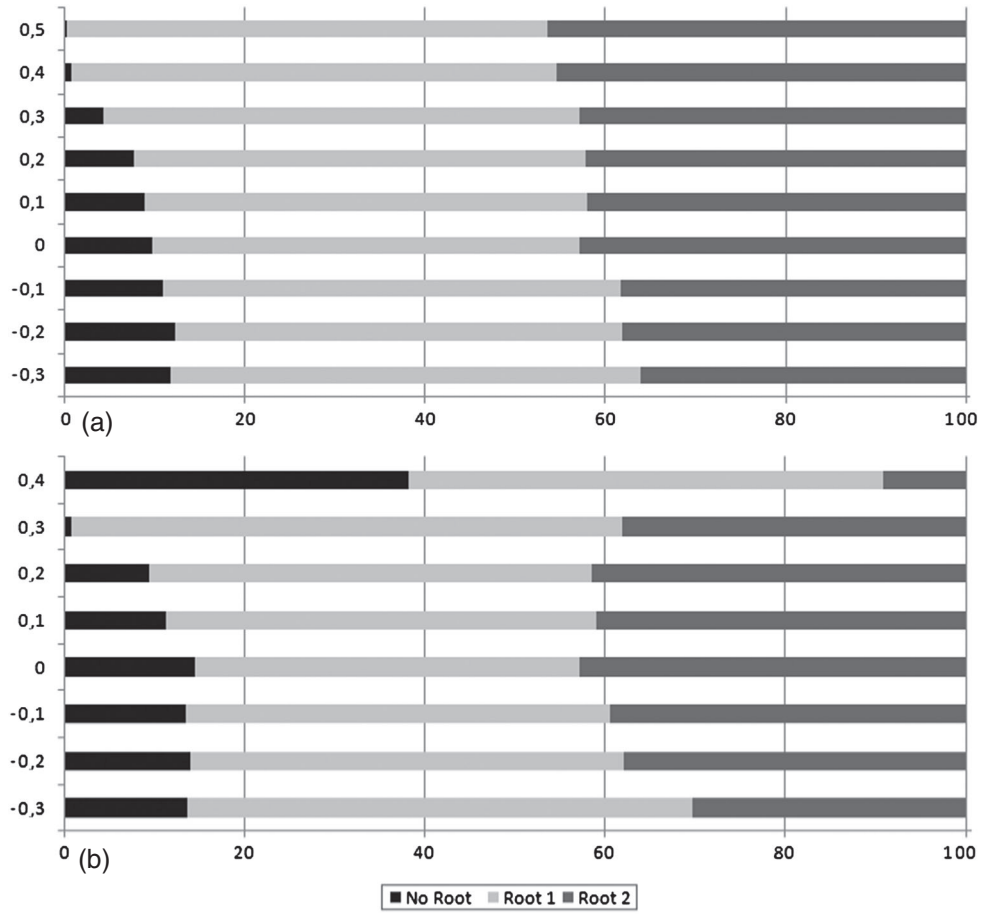

Figure 4. Percentage of initial conditions (in the form of stacked bars) that form the basin of attraction for each root as well as the basin of divergence for the cases $\alpha = 1.3$ [Figure 4(a)] and $\alpha = 1.4$ [Figure 4(b)]. The labels of the vertical axis in both figures represent the associated value of the β parameter.

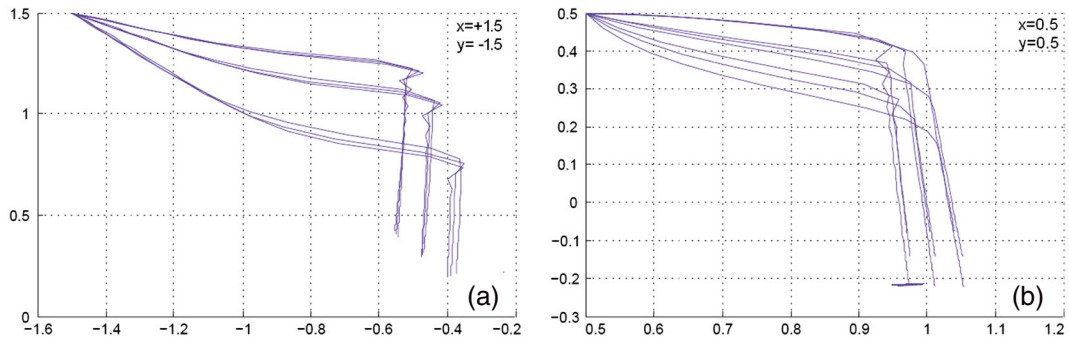

Figure 5. The sequence of consecutive points that finally converged to one of the system roots for parameter values $\alpha = 1.2, 1.3, 1.4$ and $\beta = 0.2, 0.3, 0.4$, and for the initial conditions $(x, y) = (1.5, -0.5)$ [Figure 5(a)] and $(x, y) = (0.5, 0.5)$. [Figure 5(b)].

the values of the partial derivatives via the estimation of δ parameters), in contrast with the other methods that are more difficult to implement because they require a lot of mathematics and complicated operations such as Jacobian evaluations at specified points.

5. Conclusion

In this research, we used a back-propagation neural network to solve the nonlinear algebraic system associated with the 2-cycle points of the Henon chaotic map, by using an algorithm that guarantees convergence. This algorithm was tested with many different values of the parameters α and β, and it was studied with respect to its convergence and the speed of it, for a broad set of initial conditions. The experiments show that the algorithm is characterized by a large speed of convergence to each one of the two available roots, for initial

Table III. Simulation results for the parameter value $\alpha = 1.3$ associated with alternative numerical algorithms used for the identification of the fixed points of the Henon map.

b	x_0	y_0	X_{root}	Y_{root}	TRDL	TRR	LM	NR
−0.3	0.2	−1.0	0.361324	−0.191602	4	4	5	6
−0.3	0.6	0.0	0.638675	−0.108397	4	3	4	5
−0.2	1.6	−0.4	0.822339	−0.020147	5	5	6	3
−0.2	0.0	−0.2	0.100737	−0.164467	4	4	4	6
−0.1	−0.8	−0.2	−0.058844	−0.090499	4	5	5	6
−0.1	1.0	0.0	0.904998	0.005884	5	4	4	4
0.0	−0.2	0.0	−0.185861	0.000000	3	3	3	4
0.0	1.0	0.0	0.955092	0.000000	3	3	4	4
0.1	−1.0	0.2	−0.293973	0.098628	5	5	5	6
0.1	1.4	0.0	0.986281	−0.029397	4	4	5	5
0.2	−0.4	0, 2	−0.388875	0.200852	3	3	3	3
0.2	1.8	0.2	1.004260	−0.077775	7	5	6	5
0.3	−1.0	0.4	−0.473584	0.303613	4	4	5	5
0.3	1.6	0.0	1.012046	−0.142075	5	5	5	5
0.4	−0.4	0.4	−0.549914	0.404581	4	4	4	5
0.4	0.4	−0.4	1.011453	−0.219965	5	6	6	7
0.5	−1.2	0.6	−0.619039	0.501827	4	4	5	5
0.5	1.8	−0.2	1.003654	−0.309519	5	5	5	6

TRDL, trust-region-dogleg; TRR, trust-region-reflective; LM, Levenberg–Marquardt; NR, Newton–Raphson.

conditions that belong to the same interval and for different values of the parameters α and β—for example, $(x, y) = (-0.2 : 0.1, -2 : 2)$ for root 1, and $(x, y) = (0.2 : 2, -2 : 2)$ for the root 2. Furthermore, the algorithm exhibits a similar behavior regarding the estimation of the final values. All these observations are very helpful in the further investigation of the Henon map behavior for the m-cycle points ($m \geq 3$).

Acknowledgements

The research of Athanasios Margaris, Konstantinos Goulianas, Konstantinos Diamantaras and Theophilos Papadimitriou has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF)—research funding program: THALES, investing in knowledge society through the European Social Fund.

References

1. Mathia K, Saeks R. *Solving Nonlinear Equations Using Recurrent Neural Networks*, World Congress on Neural Networks, July 17-21, Vol. 1. Lawrence Erlbaum Associates, Mahwah, NJ: Washington DC, USA, 1995. pp.76-79.
2. Mishra D, Kalva PK. Modified Hopfield neural network approach for solving nonlinear algebraic equations. *Engineering Letters*; **14**(1):135–142.
3. Luo D, Han Z. Solving nonlinear equation systems by neural networks. *Proc. of International Conference on Systems, Man and Cybernetics* October 1995; **1**:858–862.
4. Li G, Zeng Z. A neural network algorithm for solving nonlinear equation systems. *Proc. of 20th International Conference on Computational Intelligence and Security*, Los Alamito, USA, 2008; 20–23.
5. Margaris A, Roumeliotis M, Adamopoulos M. Development of neural models for the logistic equation and study of the neural based trajectories in the convergence, periodic and chaotic regions. *Neural, Parallel & Scientific Computations* 2001; **9**:221–230.
6. Yousefizadeh H, Jonckheere EA. Neural network modeling of discrete time chaotic maps. *Technical Report*, January 2002. Available from the http://newport.eecs.uci.edu/~hyousefi/publ/tnn.pdf.
7. Bakker R, Schouten JC, Takens F, van den Bleek CM. Neural network model to control an experimental chaotic pendulum. *Physical Review E* 1996; **54**(4):3545–3552. DOI: 10.1103/PhysRevE.54.3545.
8. Weeks ER, Burgess JM. Evolving artificial neural networks to control chaotic systems. *Physical Review E* 1997; **56**(2):1531–1540. DOI: 10.1103/PhysRevE.56.1531.
9. Henon M. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics* 1976; **50**:69–77. DOI: 10.1007/BF01608556.
10. Galias Z. Existence and uniqueness of low period cycles and estimation of topological entropy for the henon map. *Proceedings of International Symposium on Nonlinear Theory and Applications, NOLTA'98*, Le Regent, Switzerland, 1998; 187–190.
11. Sonis M. Once more on Henon map: analysis of bifurcations. *Chaos, Solitons & Fractals* 1996; **7**:2215–2234.
12. Margaris A, Adamopoulos M. Solving nonlinear algebraic systems using artificial neural networks. *Proc. of the 10th International Conference on Engineering Applications of Artificial Neural Networks*, Thessaloniki, Greece, August 2007; 107–120.

13. Margaris A, Goulianas K. Finding aLL roots of 2 × 2 non linear algebraic systems using back propagation neural networks. *Neural Computing and Applications* 2012; **21**(5):891–904.
14. Conn NR, Gould NIM, Toint PL. *Trust-Region Methods*, MPS/SIAM Series on Optimization. SIAM and MPS: Philadelphia, USA, 2000.
15. Levenberg K. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics* 1944; **2**:164–168.
16. Marquardt D. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*; **11**(2):431–441. DOI: 10.1137/0.111030.
17. Press W, Teukolsky S, Vetterling W, Flannery B. *Numerical Recipes in C - The Art of Scientific Programming*, 2nd Edition. Cambridge University Press: Cambridge, UK, 1992.