HYBRID NEURAL NETWORK–PARTICLE SWARM ALGORITHM TO DESCRIBE CHAOTIC TIME SERIES

Juan A. Lazzús, Ignacio Salfate, Sonia Montecinos*

Abstract: An artificial neural network (ANN) based on particle swarm optimization (PSO) was developed for the time series prediction. This hybrid ANN+PSO algorithm was applied on Mackey–Glass series in the short-term prediction $x(t+6)$ and the long-term prediction $x(t+84)$, from the current value $x(t)$ and the past values: $x(t-6)$, $x(t-12)$, $x(t-18)$. Four cases were studied, alternating the time-delay parameter as 17 or 30. Also, the first four largest Lyapunov exponents were obtained for different time-delay. Simulation shows that this ANN+PSO method is a very powerful tool for making prediction of chaotic time series.

Key words: *Chaotic time series, time series prediction, Mackey–Glass series, artificial neural network, particle swarm optimization*

Received: November 30, 2013 **DOI:** 10.14311/NNW.2014.24.034
Revised and accepted: December 5, 2014

1. Introduction

Chaotic time series are an important research and application area. Several models for time series data can have many forms and represent different stochastic processes. The prediction of time series is one of the most important aspects for the practical usage of scientific and engineering knowledge, including physical science. In the last decades many different techniques have been developed for the prediction of time series based on artificial neural network (ANN) models that includes back-propagation algorithm [16], radial basic function [7], recurrent network [26], genetic algorithms [24], fuzzy system application [14], and wavelet approach [6].

Time series contain much information about dynamic systems [12]. These systems are usually modeled by delay-differential equations. Some of them, for example, the Mackey–Glass equation [22], the Ikeda equation [13], and equation for an electronic oscillator with delayed feedback [9], are standard examples of time-delay systems [3].

The principal problem of the time series study consists of predicting the next value of a series known up to a specific time, using the known past values of the

*Juan A. Lazzús – Corresponding Author, Ignacio Salfate, Sonia Montecinos, Departamento de Física, Universidad de La Serena, Casilla 554, La Serena, Chile, Tel.: +56-51-2204128, E-mail: jlazzus@dfuls.cl

©CTU FTS 2014

series. In time series prediction, this is usually first embedded in a state space using delay coordinates:

$$\mathbf{x}(t) = [x(t), x(t+\tau), \cdots, x(t+(d-1)\tau)] \tag{1}$$

where $x(t)$ is the value of the time series at time t, τ a suitable time-delay and d the order of the embedding. This embedded vector is then used to predict the next value of the series $x(t+\tau)$. Therefore, the non-linear dependence of the level of a series on previous data points is of interest, partly because of the possibility of producing a chaotic time series. Also, the potential in short-term prediction in chaos based models have been widely appreciated, and improving prediction accuracy of such models by various techniques [16]. However, the long-term prediction has not been widely studied in the literature.

In this work, chaotic time series data taken from the Mackey–Glass differential equation were used to develop a neural network. In order to obtain still a more effective correlation and prediction, particle swarm algorithm has been introduced to update the weights of all layers of the network. Then, this hybrid algorithm was used in the short-term prediction and long-term prediction.

2. Hybrid algorithm

A feed-forward neural network was used to represent non-linear relationships among variables [16]. This ANN was implemented replacing standard back-propagation algorithm with particle swarm optimization (PSO).

PSO is a population-based optimization tool, where the system is initialized with a population of random particles and the algorithm searches for optima by updating generations [20]. In each iteration, the velocity of each particle j is calculated according to the following formula [21]:

$$v_j^{k+1} = \omega v_j^k + c_1 r_1 \left(\psi_j^k - s_j^k\right) + c_2 r_2 \left(\psi_g^k - s_j^k\right) \tag{2}$$

where s and v denote a particle position and its corresponding velocity in a search space, respectively. k is the current step number, ω is the inertia weight, c_1 and c_2 are the acceleration constants, and r_1, r_2 are elements from two random sequences in the range (0,1). s_j^k is the current position of the particle, ψ_j^k is the best one of the solutions that this particle has reached, and ψ_g is the best solutions that all the particles have reached. In general, the value of each component in v can be clamped to the range $[-v_{\max}, +v_{\max}]$ control excessive roaming of particles outside the search space [20,21]. After calculating the velocity, the new position of each particle is:

$$s_j^{k+1} = s_j^k + v_j^{k+1} \tag{3}$$

The procedure to calculate the output values, using the input values of the network were as follows [19]:
The input data were normalized using the following equation:

$$p_i = \left(X_i - X_i^{\min}\right) \frac{2}{X_i^{\max} - X_i^{\min}} - 1 \tag{4}$$

where X_i is the input variables i, $X_i^{\min}$ and $X_i^{\max}$ are the smallest and largest value of the data. Next, the net inputs (N) are calculated for the hidden neurons coming from the inputs neurons. For a hidden neuron:

$$N_i^h = \sum_i^n w_{i,j}^h p_i + b_{i,j}^h \tag{5}$$

where p_i is the vector of the inputs of the training, $w_{i,j}^h$ is the weight of the connection among the input neurons with the hidden layer h, and the term $b_{i,j}^h$ corresponds to the bias of the neuron of the hidden layer h, reached in its activation. The PSO algorithm is very different then any of the traditional methods of training [20]. Each neuron contains a position and velocity. The position corresponds to the weight of a neuron $\left(s_i^k \to w_{i,j}^h\right)$. The velocity is used to update the weight $\left(v_i^{k+1} \to w'_{i,j}\right)$. Starting from these inputs, the outputs (y_i) of the hidden neurons are calculated, using a transfer function f^h associated with the neurons of this layer.

$$y_i = f^h\left(\sum_i^n w_{i,j}^h p_i + b_{i,j}^h\right) \tag{6}$$

To minimize the error, the transfer function f it should be differentiable. In the ANN, the hyperbolic tangent function ($tansig$) was used as

$$f(N_i) = \frac{e^{N_i} - e^{-N_i}}{e^{N_i} + e^{-N_i}} \tag{7}$$

All the neurons of the ANN have an associated activation value for a give input pattern, and the algorithm continues finding the error that is presented for each neuron, except those of the input layer. After finding the output values, the weights of all layers of the network are actualized $w_{i,j} \to w'_{i,j}$ by PSO, using eqs. (2 and 3) [21]. The velocity is used to control how much the position is updated. On each step, PSO compares each weight using the data set. The network with the highest fitness is considered the global best. The other weights are updated based on the global best network rather than on their personal error or fitness [20]. In this article, we used the mean square error (MSE) to determine network fitness for the entire training set:

$$\text{MSE} = \frac{\sum_{i=1}^n \left(Y_i^{\text{true}} - Y_i^{\text{calc}}\right)^2}{n} \tag{8}$$

where Y_i^{true} is the real data and Y_i^{calc} is the calculated output value obtained from the normalized output (y_i) of the network. This process was repeated for the total number of patterns in the training set. For a successful process the objective of the algorithm is to modernize all the weights minimizing the total root mean squared error (RMSE):

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n \left(Y_i^{\text{true}} - Y_i^{\text{calc}}\right)^2}{n}} \tag{9}$$

$$\varepsilon = \min(\text{RMSE}) \tag{10}$$

The step-to-step approach of PSO+ANN can be summarized as:

Step 1: Initialize the positions (weights and biases) and velocities of a group of particles randomly. The particles represents the weight vectors of ANN, including biases. The dimension of the search space is therefore the total number of weights and biases.
Step 2: The ANN is trained using the initial particles position in PSO. The learning error produced from ANN network can be treated as particles fitness value according to initial weight and bias. The current best fitness achieved by particle j is set as ψ_j^k. The ψ_j^k with best value is set as ψ_g and this value is stored.
Step 3: Evaluate the desired optimization fitness function (Eq. 10) over a given data set.
Step 4: Compare the evaluated fitness value of each particle (F_j) with its value. If $F_j < \psi_j^k$ then $\psi_j^k = s_j^k$ is the coordinates corresponding to best particle so far.
Step 5: The objective function value is calculated for new positions of each particle. If a better position is achieved by an agent, ψ_j^k value is replaced by the current value. As in Step 1, ψ_g value is selected among ψ_j^k values. If the new ψ_g value is better than previous value, it is replaced by the current ψ_g value and this value is stored. if $F_j < \psi_g$ then $\psi_g = s_j^k$ is the particle having the overall best fitness over all particles in the swarm.
Step 6: The learning error at current epoch will be reduced by changing the particles position, which will update the weight and bias of the network. Change the velocity and location of the particle according to movement equations (Eqs. 2 and 3). The new sets of positions (weights and biases) are produced by adding the calculated velocity value to the current position value. Then, the new sets of positions are used to produce new learning error in ANN.
Step 7: This process is repeated until the stopping conditions either minimum learning error or maximum number of iteration are met, then stop; otherwise Loop to Step 3 until convergence.
Step 8: The optimum weight and biases for ANN model are obtained by PSO. Best training process is obtained for ANN.

Fig. 1 presents a block diagram of the ANN+PSO algorithm developed in this study. In PSO, the inertial weight ω, the constant c_1 and c_2, the number of particles N_{part} and the maximum speed of particle summary the parameters to syntonize for their application in a given problem. An exhaustive trial-and-error procedure was applied for tuning the PSO parameters. Firstly, the effect of ω was analyzed for values of 0.1 to 0.9. Fig. 2*a* shows the values of ω that favored the search of the particles and accelerated the convergence. Next, the effect of N_{part} was analyzed for values of 10 to 100 particles in the swarm. Fig. 2*b* shows that the best population to solve the problem is of 50 particles. Tab. I shows the selected parameters for this hybrid algorithm.

3. Simulations

The data points were generated from the Mackey–Glass time-delay differential equation [22,10] which is defined as follows:

$$\frac{dx}{dt} = \beta x(t) + \frac{\alpha x(t-\tau)}{1 + x(t-\tau)^{10}} \tag{11}$$

where t is a variable, x is a function of t, and τ is the time delay. The initial values of the time series are $\alpha = 0.2$, $\beta = 0.1$, and $x(0) = 1.2$. If $\tau \geq 17$, the time series show the chaotic behaviour [24,10].

The goal of the task is to use known values of the time series up to the point $x = t$ to predict the value at some point in the future $x = t + T$. The standard method for this type of prediction is to create a mapping from d points of the time

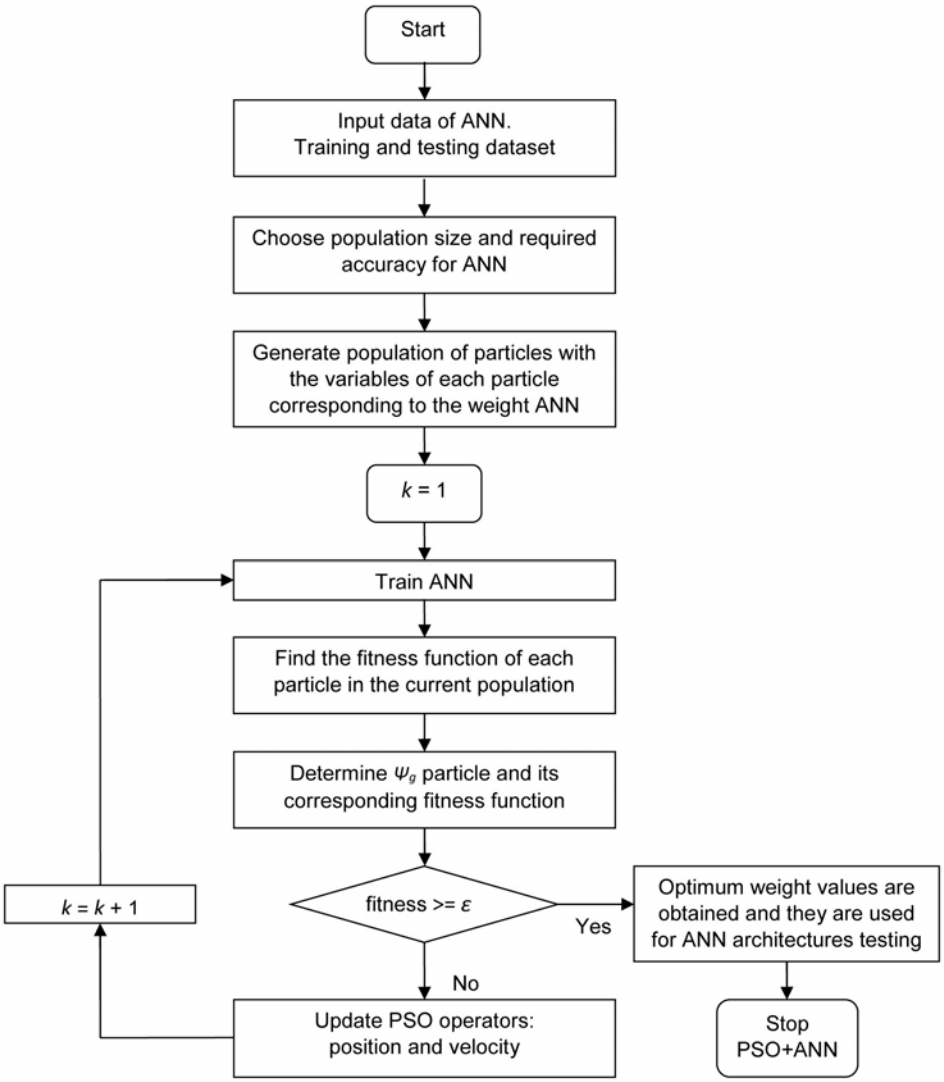

Fig. 1 *Flow diagram for training of the ANN using PSO algorithm.*

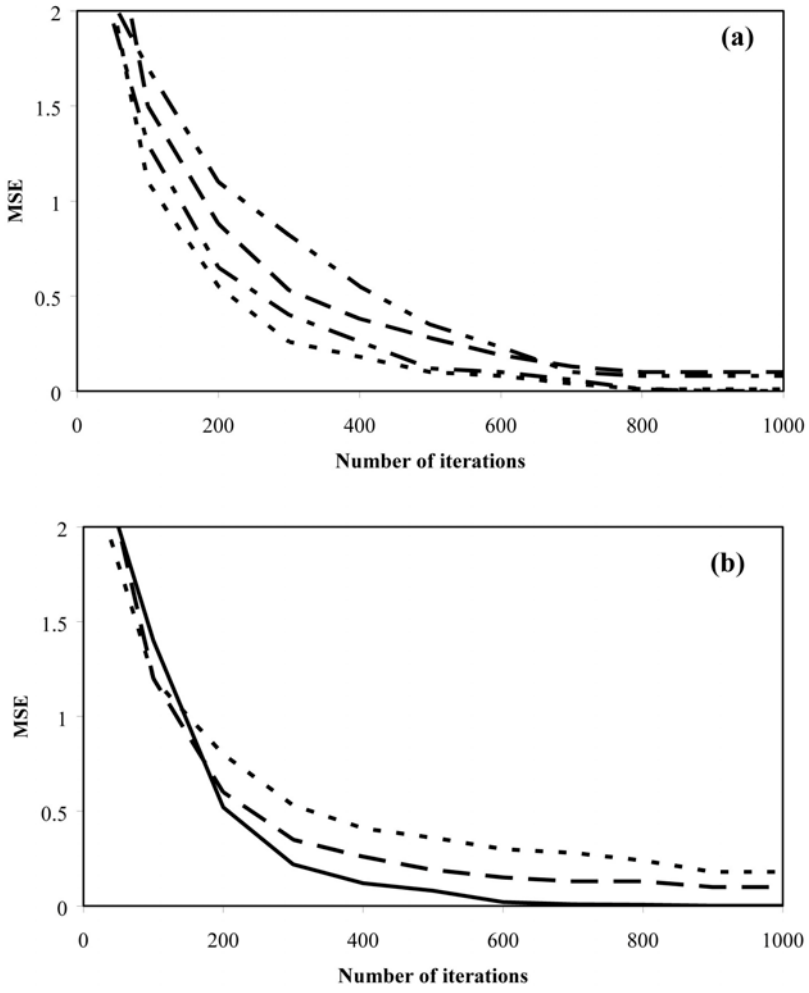

Fig. 2 *Convergence graphics. (a) Determination of the best values for* ω *as: 0.3 (– –), 0.5(- - -), 0.7(– · –), 0.9(– · · –). (b) Effect of* N_{part} *for: 25(- - -), 50(—), 100(– –).*

series spaced Δ apart, that is $(x(t), x(t-\Delta), \ldots, x(t-(d-1)\Delta))$, to a predicted future value $x(t+T)$.

In order to solve the Mackey–Glass equation, the fourth-order Runge–Kutta method was applied to find the numerical solution. The time series was obtained evaluating the solution of eq. (11) at each integer points. Step size of 0.1 was used to generate a time series, and $x(t)$ is thus derived for $0 \leq t \leq 1500$ with $x(t) = 0$ for $t < 0$ in the integration. Four non consecutive points in the time series are given to generate each input vector X_i (where $i = 1, 2, \ldots, n$) of the input matrix $\mathbf{X}$, defined as follows:

$$\mathbf{X} = \begin{bmatrix} X_i \\ \vdots \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} x(t) & x(t-\Delta) & \cdots & x(t-(d-1)\Delta) \\ x(t+1) & x(t) & \cdots & \vdots \\ \vdots & \vdots & & x(t) \\ x(t+n-1) & x(t+n) & \cdots & \vdots \end{bmatrix} \tag{12}$$

A similar criterion was used to create the output matrix, defined as follows:

$$\mathbf{Y} = \begin{bmatrix} x(t+\Delta) \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} Y_i \\ \vdots \\ \vdots \\ Y_n \end{bmatrix} \tag{13}$$

Then, the ANN+PSO method was used in the short-term prediction and long-term prediction. This hybrid algorithm was trained to predict the values $x(t+6)$ and $x(t+84)$ from the current value $x(t)$ and the past values, using the standard form applied in the literature, for $d = 4$ and $\Delta = T = 6$ [7,24].

$$x(t+6) = F\left[x(t), x(t-6), x(t-12), x(t-18)\right] \tag{14}$$

$$x(t+84) = F\left[x(t), x(t-6), x(t-12), x(t-18)\right] \tag{15}$$

One thousand data points of the above format were collected. The first 500 were used for training while the others were used for testing the ANN+PSO method.

Section	Parameter	Value
ANN	NN-type	feed-forward
	Number of hidden layers	1
	Transfer function	tansig (Eq. 7)
	Number of iterations	1500
	Normalization range	[–1, 1]
	Weight range	[–100, 100]
	Bias range	[–10, 10]
	Minimun error	1e–3
PSO	Number of particles in swarm (N_{part})	50
	Number of iterations ($k_{\max}$)	1500
	Cognitive component (c_1)	1.494
	Social component (c_2)	1.494
	Maximum velocity ($v_{\max}$)	12
	Minimum inertia weight ($\omega_{\min}$)	0.5
	Maximum inertia weight ($\omega_{\max}$)	0.7
	Objective function	RMSE (Eq. 10)

Tab. I *Parameters used in the hybrid ANN+PSO algorithm.*

And the following four cases were simulated with $\alpha = 0.2$, $\beta = 0.1$ and $x(0) = 1.2$, as:

- *Case 1*: only short-term prediction $x(t + 6)$ using $\tau = 17$.
- *Case 2*: long-term prediction $x(t + 84)$, with $\tau = 17$.
- *Case 3*: short-term prediction (eq. 14), $\tau = 30$.
- *Case 4*: long-term prediction (eq. 15) using $\tau = 30$.

4. Results and discussion

The most basic architecture normally used for the analysis of chaotic time series involves a neural network consisting of three or four layers [16,12]. The input layer contains one neuron for each input parameter: $x(t)$, $x(t - 6)$, $x(t - 12)$, and $x(t - 16)$. The output layer has one node generating the scaled estimated value of the chaotic time series: $x(t + 6)$ or $x(t + 84)$. The number of hidden neurons needs to be sufficient to ensure that the information contained in the data utilized for training the network is adequately represented [19]. There is no specific approach to determine the number of neurons of the hidden layer, many alternative combinations are possible. The optimum number of neurons was determined by adding neurons in systematic form and evaluating the MSE and RMSE of the sets during the learning process [20]. For the *Case 1* the optimum architecture was 4-6-1. In the *Case 2*, the best architecture was 4-12-1. For the *Case 3* was 4-16-1. And for the *Case 4* was 4-24-1.

The results obtained with the ANN+PSO method were presented using the eqs. (8 and 9). Tab. II show the average errors for all data set used in the proposed model. These results show that the ANN+PSO model can be accurately trained and that the chosen architectures can estimate the short-term $x(t + 6)$ and long-term $x(t + 84)$ with acceptable accuracy.

Simulation	Best architecture	Output	Training set		Prediction set	
			MSE	RMSE	MSE	RMSE
Case 1	4-6-1	$x(t + 6)$	0.000029	0.0054	0.000028	0.0053
Case 2	4-12-1	$x(t + 84)$	0.000063	0.0079	0.000065	0.0080
Case 3	4-16-1	$x(t + 6)$	0.000063	0.0079	0.000064	0.0080
Case 4	4-24-1	$x(t + 84)$	0.000092	0.0096	0.000091	0.0095

Tab. II *Average errors for all data set used in the ANN+PSO method.*

Fig. 3 shows a comparison between recorded and calculated values of the standard configuration applied in the Mackey–Glass time series (*Case 1*). Fig. 3*a* shows the training set for $x(t + 6)$ as a function of the time t. For this training set, the correlation coefficient R^2 was 0.99952. Fig. 3*b* shows the MSE of training as a function of the time t, with a MSE_{max} of 0.00032. Fig. 3*c* shows the prediction set of $x(t + 6)$. Note that for the prediction set, R^2 was 0.99953. Fig. 3*d* shows the MSE obtained for the prediction set with MSE_{max} of 0.00027.

Tab. III shows a comparison between some computational methods found in the literature [5, 6, 11, 14, 17] and the ANN+PSO method proposed in this work.

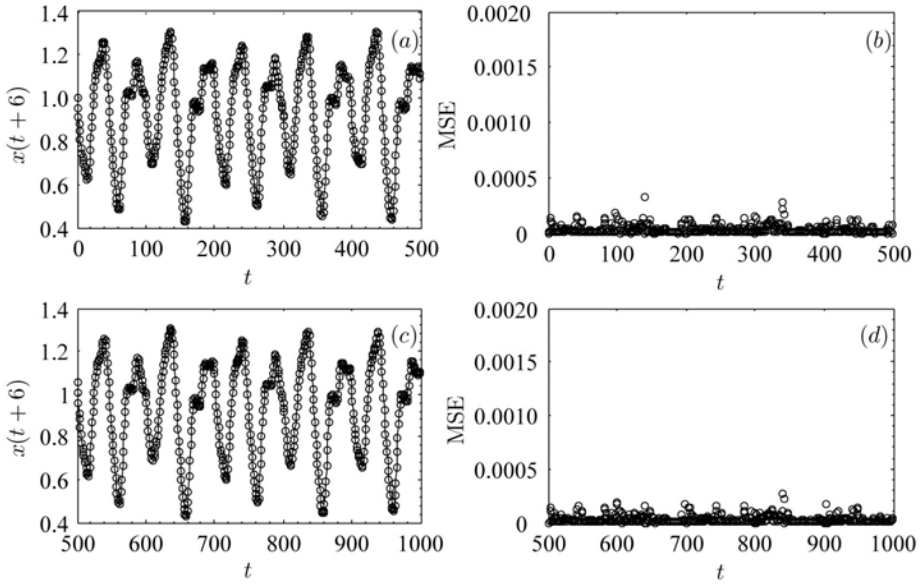

Fig. 3 *Calculated values (circles) obtained for the Mackey–Glass time series – Case 1 (solid line). (a) Training set for x(t+6); (b) MSE for the training set; (c) Prediction set for x(t+6); (d) MSE for the prediction set.*

Method	RMSE
Linear model	0.5503
Auto regressive model	0.1900
Cascade correlation NN	0.0624
Genetic algorithm and fuzzy system [17]	0.0490
Six order polynomial	0.0402
Back-propagation NN	0.0262
RBFNN[a] [8]	0.0114
ANFIS[b] and fuzzy system [14]	0.0084
ARFNN+SVR[c] model 1 [11]	0.0080
ARFNN+SVR[c] model 2 [11]	0.0073
WNNd model 1 [6]	0.0071
Neural tree [5]	0.0069
WNN[d] model 2 [6]	0.0059
This work *Case 1*	0.0053

[a]Radial basis function neural network
[b]Adaptive neuro-fuzzy inference system
[c]Annealing robust fuzzy neural network with support vector regression
[d]Wavelet neural network

Tab. III *Comparison between computational methods found in the literature for the short-term prediction.*

The low errors found with the proposed method (MSE = 0.000028, and RMSE = 0.0053) indicate that it can estimate the Mackey–Glass time series $x(t+6)$ with better accuracy than other methods.

Fig. 4 shows other comparison between recorded and calculated values of the long-term prediction $x(t+84)$. Fig. 4*a* shows the training set for the *Case 2* as a function of the time t. For this figure, R^2 was 0.99877. Fig. 4*b* shows the MSE for training set as a function of the time t, with MSE_{max} of 0.00059. Fig. 4*c* shows the prediction set for the $x(t+84)$, with R^2 of 0.99853. Fig. 4*d* shows the errors for the prediction set with MSE_{max} of 0.00057. Tab. IV shows a comparison between some computational methods found in the literature [1,2,23,25] and the result obtained with the ANN+PSO method. This comparison was made using the normalized root mean squared error (NRMSE), defined as follows:

$$\mathrm{NRMSE} = \sqrt{\frac{\sum_{i=1}^{n}\left(y_i^{\mathrm{true}} - y_i^{\mathrm{calc}}\right)^2}{\sum_{i=1}^{n}\left(y_i^{\mathrm{true}} - \overline{y}_i\right)^2}} \tag{16}$$

The low errors found with the proposed method (MSE = 0.000065, RMSE = 0.0080, and NRMSE = 0.0383) indicate that it can estimate the long-term $x(t+84)$ of the Mackey–Glass time series with better accuracy than other methods.

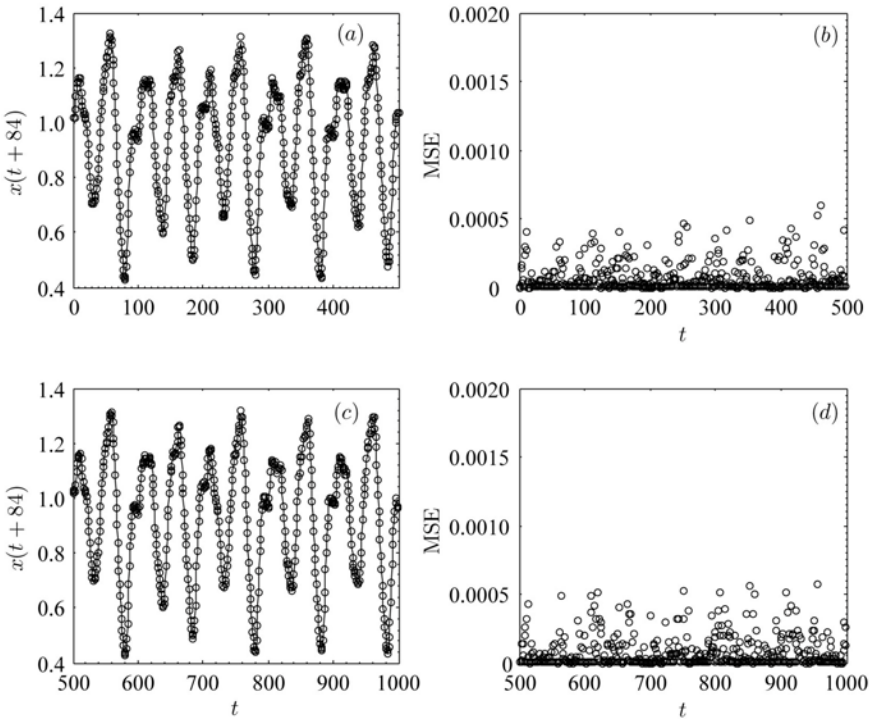

Fig. 4 *Calculated values (circles) obtained for the Mackey–Glass time series – Case 2 (solid line). (a) Training set for x(t+84); (b) MSE for the training set; (c) Prediction set for x(t+84); (d) MSE for the prediction set.*

Method	NRMSE
Linear model	1.504
Cascade correlation NN	0.170
Fuzzy system [2]	0.103[a]
RBFNN[b] [1]	0.097[a]
Six order polynomial	0.085
Back-propagation NN	0.060
Genetic algorithm and RBFNN[b] [25]	0.050[a]
Neural gas [23]	0.050
This work *Case* 2	0.038

[a]long-term: $x(t+85)$
[b]Radial basis function neural network

Tab. IV *Comparison between computational methods found in the literature for the long-term prediction.*

Fig. 5 shows a comparison between recorded and calculated values for the *Case 3*. Fig. 5*a* shows the training set as a function of the time t, with R^2 of 0.99930. Fig. 5*b* shows the MSE for this training set with a MSE_{max} of 0.00061.

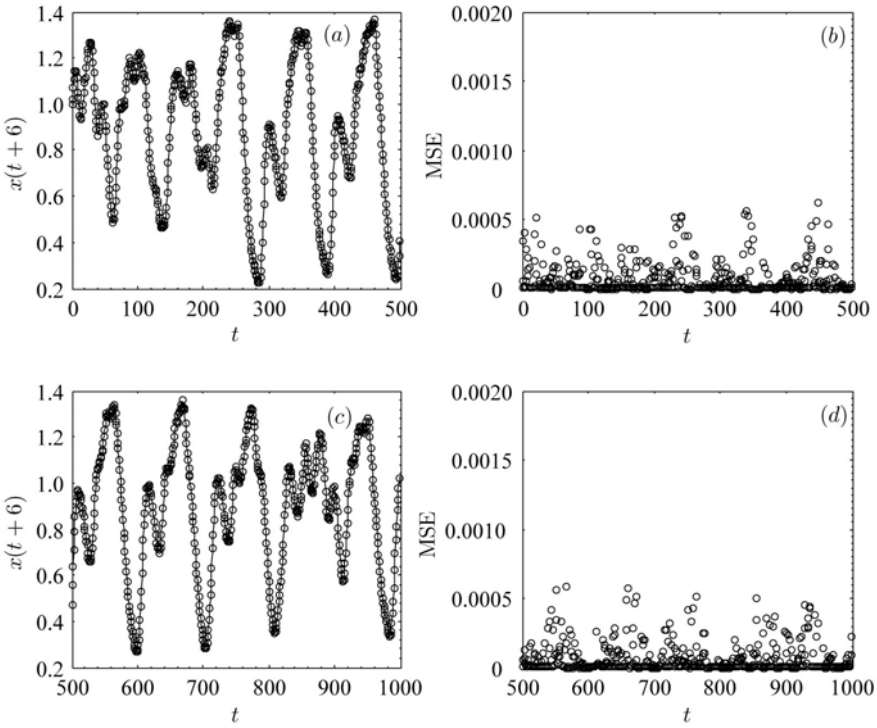

Fig. 5 *Calculated values (circles) obtained for the Mackey–Glass time series – Case 3 (solid line). (a) Training set for x(t+6); (b) MSE for the training set; (c) Prediction set for x(t+6); (d) MSE for the prediction set.*

Fig. 5*c* shows the prediction set with R^2 of 0.99919. Fig. 5*d* shows the MSE obtained for this prediction set with MSE_{max} of 0.00058. Fig. 6 show the results for the *Case 4*. Fig. 6*a* shows the training set with R^2 of 0.99900. Fig. 6*b* shows the MSE of the training set with MSE_{max} of 0.00109. Fig. 6*c* shows the prediction set with R^2 of 0.99877. Fig. 6*d* shows the MSE for this prediction set with MSE_{max} of 0.00078.

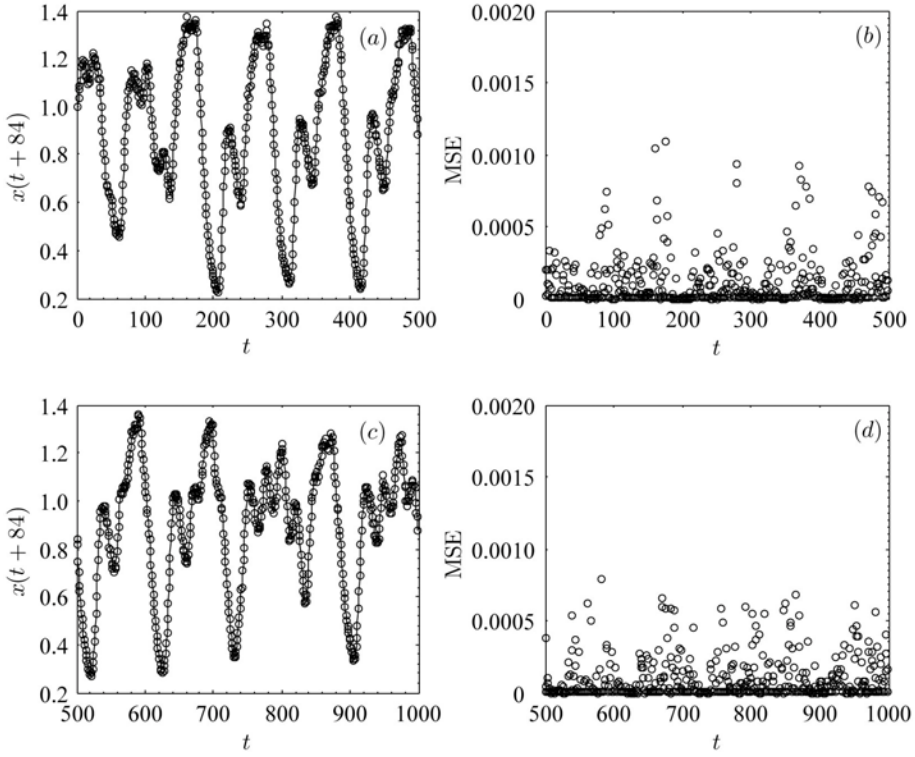

Fig. 6 *Calculated values (circles) obtained for the Mackey–Glass time series – Case 4 (solid line). (a) Training set for x(t+84); (b) MSE for the training set; (c) Prediction set for x(t+84); (d) MSE for the prediction set.*

The time series study consists of predicting the next value of a series using the known past values of the series, because all relevant information about the next event is conveyed by a few recent events contained within a small time window. Thus, short long-term is able to solve many time series tasks unsolvable with fixed size time windows. Tab. V shows some relations to a mean period of short long-term obtained with the proposed method.

Fig. 7 shows a representation of the chaotic attractor for all cases studies. This Figure shows that with the above parameters, the system operates in a high-dimensional regime. The Mackey–Glass system is infinite dimensional (because it is a time-delay equation) and, thus, has an infinite number of Lyapunov exponents (λ_i) [10]. The Lyapunov exponents of dynamical systems are one of a number of invariants that characterize the attractors of the system in a fundamental way [4].

Output	Training set		Prediction set	
	MSE	RMSE	MSE	RMSE
$t+10$	0.000034	0.0058	0.000033	0.0057
$t+20$	0.000040	0.0063	0.000036	0.0060
$t+30$	0.000042	0.0065	0.000040	0.0063
$t+40$	0.000045	0.0067	0.000045	0.0067
$t+50$	0.000052	0.0072	0.000052	0.0072
$t+60$	0.000054	0.0073	0.000055	0.0074
$t+70$	0.000058	0.0076	0.000060	0.0077
$t+80$	0.000061	0.0078	0.000065	0.0081

Tab. V *Some relations to a mean period (Case 1) obtained with the ANN+PSO method.*

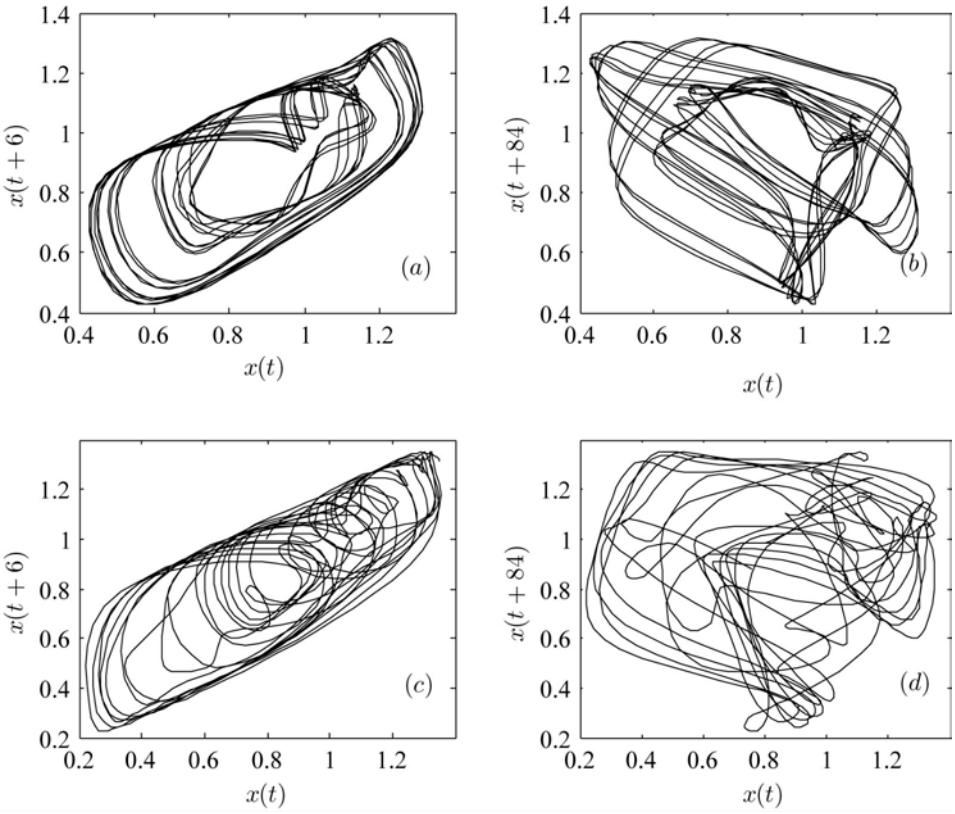

Fig. 7 *Chaotic attractors for the Mackey–Glass time series studies: (a) short-term representation with $\tau = 17$; (b) long-term representation with $\tau = 17$; (c) short-term representation with $\tau = 30$; (d) long-term representation with $\tau = 30$.*

Tab. VI shows a comparison of the first four largest Lyapunov exponents of the Mackey–Glass system reported in ref. [10], with the Lyapunov exponents obtained for the ANN+PSO method for different τ.

τ	λ_i, ref. [10]	λ_i, ANN+PSO
17	0.00860	0.00900
	0.00100	0.00132
	–0.03950	–0.04100
	–0.05050	–0.05000
23	0.00940	0.00938
	0.00000	0.00002
	–0.01200	–0.01400
	–0.03400	–0.04000
30	0.00710	0.00730
	0.00270	0.00300
	0.00000	0.00004
	–0.01670	–0.01900

Tab. VI *Comparison of the first four largest Lyapunov exponents of the Mackey–Glass system reported in ref. [10], with the Lyapunov exponents obtained for the ANN+PSO method.*

An approach to determining an appropriate cutoff value for the number of exponents can be related to the Lyapunov dimension [4]. This idea was originally explored by Kaplan and Yorke [15]. Thus, Kaplan and Yorke conjecture that this dimension (D_{KY}) is equal to the information dimension [18]. Tab. VII present a comparison of the fractal dimension and Lyapunov dimension calculated by the Kaplan–Yorke conjecture [15] for the Mackey–Glass system reported in ref. [10], with the Lyapunov dimension calculated for the Mackey–Glass system generated by the ANN+PSO method. Fig. 8 shows a comparison of the dimension as a function of the τ (from 17 to 30, and at increments of 0.5) between the Lyapunov dimension calculated for the Mackey–Glass system reported in ref. [10], and Lyapunov dimension simulated for the Mackey–Glass system by the ANN+PSO method.

τ	D_{F}, ref. [10]	D_{KY}, ref. [10]	D_{KY}, ANN+PSO
17	2.13	2.10	2.10
23	2.76	2.82	2.80
30	≈3.00	3.58	3.50

Tab. VII *Comparison of the fractal dimension and Lyapunov dimension calculated by the Kaplan–Yorke conjecture for the Mackey–Glass system reported in ref. [10], with the Lyapunov dimension calculated for the Mackey–Glass system generated by the ANN+PSO method.*

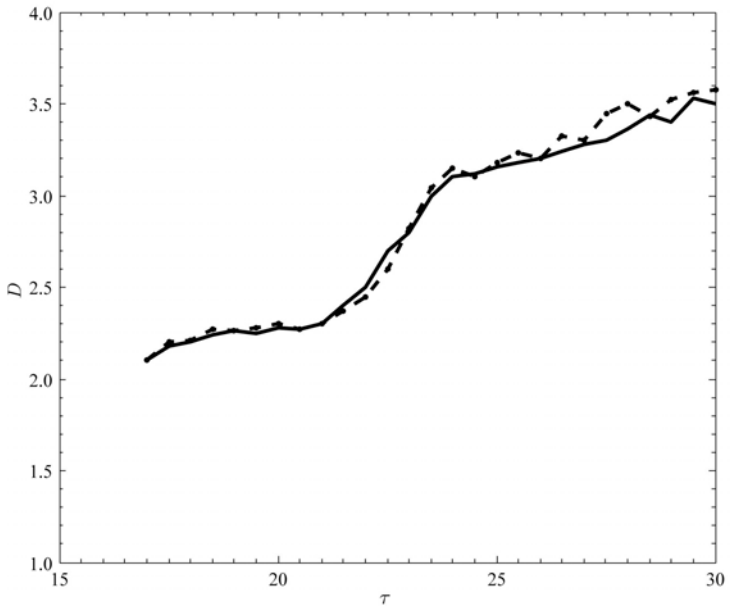

Fig. 8 *The dimension as a function of the delay (τ = 17 to 30), at increments of 0.5. Lyapunov dimension [15], is shown as: dotted line, taken from ref. [10], and solid line, calculated for the ANN+PSO method.*

5. Conclusions

In this work, a hybrid algorithm based on artificial neural network and particle swarm optimization (ANN+PSO) was used in the short-term prediction and long-term prediction, with data taken from the Mackey–Glass time series.

Based on the results and discussion presented in this study, the following main conclusions are obtained: i) ANN+PSO method proposed in this work can be properly trained and used in the short-term prediction and long-term prediction of time series with acceptable accuracy; ii) the current value $x(t)$ and the past values used have influential effects on the good training and predicting capabilities of the chosen network; and iii) simulation shows that this hybrid ANN+PSO algorithm is a very powerful tool for making prediction of chaotic time series, and the low deviations found with the proposed method shows a better accuracy than other methods available in the literature.

Acknowledgement

The authors thank at Direction of Research of the University of La Serena, through the research grant PT13143 and PI13141. Special thanks at Department of Physics of the University of La Serena for the special support that made possible the preparation of this paper.

References

[1] AWAD M., et al. Prediction of time series using RBF neural networks: a new approach of clustering. *International Arab Journal of Information Technology*. 2009, 6(2), pp. 138–143, doi: 10.1.1.167.970.

[2] BERSINI H., DUCHATEAU A., BRADSHAW N. Using incremental learning algorithms in the search for minimal effective fuzzy models. In: *Proceedings of the 6th IEEE International Conference on Fuzzy Systems (FUZZY-1997)*, Barcelona, Spain. Barcelona: 21 IEEE Press, 1997, pp. 1417–1422, doi: 10.1109/FUZZY.1997.619751.

[3] BEZRUCHKO B.P., et al. Reconstruction of time-delay systems from chaotic time series. *Physical Review E*. 2001, 64(5), pp. 056216(1)–056216(6), doi: 10.1103/PhysRevE.64.056216.

[4] BROWN R., BRYANT P., ABARBANEL H.D.I. Computing the Lyapunov spectrum of a dynamical system from an observed time series. *Physical Review A*. 1997, 43(6), pp. 2787–2806, doi: 10.1103/PhysRevA.43.2787.

[5] CHEN Y., YANG B., DONG J. Nonlinear system modelling via optimal design of neural trees. *International Journal of Neural Systems*. 2004, 14(2), pp. 125–137, doi: 10.1142/S0129065704001905.

[6] CHEN Y., YANG B., DONG J. Time-series prediction using a local linear wavelet neural network. *Neurocomputing*. 2006, 69(4-6), pp. 449–465, doi: 10.1016/j.neucom.2005.02.006.

[7] CHNG E.S., CHEN S., MULGREW B. Gradient radial basis function networks for nonlinear and nonstationary time series prediction. *IEEE Transactions on Neural Networks*. 1996, 7(1), pp. 190–194, doi: 10.1109/72.478403.

[8] CHO K.B., WANG B.H. Radial basis function based adaptive fuzzy systems their application to system identification and prediction. *Fuzzy Sets and Systems*. 1996, 83(3), pp. 325–339, doi: 10.1016/0165-0114(95)00322-3.

[9] CHUA L.O., et al. Experimental chaos synchronization in Chua's circuit. *International Journal of Bifurcation and Chaos*. 1992, 2(3), pp. 705–708, doi: 10.1142/S0218127492000811.

[10] FARMER J.D. Chaotic attractors of an infinite-dimensional dynamical system. *Physica D*. 1982, 4(3), pp. 366–393, doi: 10.1016/0167-2789(82)90042-2.

[11] FU Y.Y., et al. ARFNNs with SVR for prediction of chaotic time series with outliers. *Expert Systems with Applications*. 2010, 37(6), pp. 4441–4451, doi: 10.1016/j.eswa.2009.12.067.

[12] HAN M., WANG Y. Analysis and modeling of multivariate chaotic time series based on neural network. *Expert Systems with Applications*. 2009, 36(2), pp. 1280–1290, doi: 10.1016/j.eswa.2007.11.057.

[13] IKEDA K. Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system. *Optics Communications*. 1979, 30(2), pp. 257–261, doi: 10.1016/0030-4018(79)90090-7.

[14] JANG J.S.R., SUN C.T., MIZUTANI E. *Neuro-fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ: Prentice-Hall, 1997.

[15] KAPLAN J., YORKE J. Chaotic behavior of multidimensional difference equations. In: H.O. PEITGEN, H.O. WALTHER, eds. *Functional Differential Equations and Approximation of Fixed Points*. New York: Springer, 1979, pp. 204–227, doi: 10.1007/BFb0064319.

[16] KARUNASINGHE D.S.K., LIONG S.Y. Chaotic time series prediction with a global model: artificial neural network. *Journal of Hydrology*. 2006, 323(1-4), pp. 92–105, doi: 10.1016/j.jhydrol.2005.07.048.

[17] KIM D., KIM C. Forecasting time series with genetic fuzzy predictor ensembles. *IEEE Transactions on Fuzzy Systems*. 1997, 5(4), pp. 523–535, doi: 10.1109/91.649903.

[18] KOSTELICH E.J., SWINNEY H.L. Practical considerations in estimating dimension from time series data. *Physica Scripta*. 1989, 40(3), pp. 436–441, doi: 10.1088/0031-8949/40/3/030.

[19] LAZZÚS J.A. Prediction of solid vapor pressures for organic and inorganic compounds using a neural network. *Thermochimica Acta*. 2009, 489(1-2), pp. 53–62, doi: 10.1016/j.tca.2009.02.001.

[20] LAZZÚS J.A. Estimation of solid vapor pressures of pure compounds at different temperatures using a multilayer network with particle swarm algorithm. *Fluid Phase Equilibria*. 2010, 289(2), pp. 176–184, doi: 10.1016/j.fluid.2009.12.001.

[21] LAZZÚS J.A. Optimization of activity coefficient models to describe vapor-liquid equilibrium of (alcohol + water) mixtures using a particle swarm algorithm. *Computers and Mathematics with Applications*. 2010, 60(8), pp. 2260–2269, doi: 10.1016/j.camwa.2010.08.016.

[22] MACKEY M.C., GLASS L. Oscillation and chaos in physiological control systems. *Science*. 1977, 197(1), pp. 287–289, doi: 10.1126/science.267326.

[23] MARTINETZ T.M., BERKOVICH S.G., SCHULTEN K.J. Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*. 1993, 4(4), pp. 558–569, doi: 10.1109/72.238311.

[24] MIRZAEE H. Linear combination rule in genetic algorithm for optimization of finite impulse response neural network to predict natural chaotic time series. *Chaos Soliton and Fractals*. 2009, 41(5), pp. 2681–2689, doi: 10.1016/j.chaos.2008.09.057.

[25] WHITEHEAD B.A., CHOATE T.D. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transactions on Neural Networks*. 1996, 7(4), pp. 869–880, doi: 10.1109/72.508930.

[26] ZHANG J.S., XIAO X.C. Predicting chaotic time series using recurrent neural network. *Chinese Physics Letters*. 2000, 17(2), pp. 88–90, doi: 10.1088/0256-307X/17/2/004.