**Open Journal of Applied Sciences, 2015, 5, 259-269**
Published Online June 2015 in SciRes. http://www.scirp.org/journal/ojapps
http://dx.doi.org/10.4236/ojapps.2015.56027

# Quantum-Inspired Neural Network with Sequence Input

**Ziyang Li[1*], Panchi Li[2]**

[1]School of Earth Science, Northeast Petroleum University, Daqing, China
[2]School of Computer and Information Technology, Northeast Petroleum University, Daqing, China
Email: [*]liziyangemail@126.com, lipanchi@vip.sina.com

Received 22 May 2015; accepted 13 June 2015; published 16 June 2015



## Abstract

**To enhance the approximation and generalization ability of artificial neural network (ANN) by employing the principles of quantum rotation gate and controlled-not gate, a quantum-inspired neuron with sequence input is proposed. In the proposed model, the discrete sequence input is represented by the qubits, which, as the control qubits of the controlled-not gate after being rotated by the quantum rotation gates, control the target qubit for reverse. The model output is described by the probability amplitude of state $|1\rangle$ in the target qubit. Then a quantum-inspired neural network with sequence input (QNNSI) is designed by employing the sequence input-based quantum-inspired neurons to the hidden layer and the classical neurons to the output layer, and a learning algorithm is derived by employing the Levenberg-Marquardt algorithm. Simulation results of benchmark problem show that, under a certain condition, the QNNSI is obviously superior to the ANN.**

## Keywords

**Quantum Rotation Gate, Multi-Qubits Controller-Not Gate, Quantum-Inspired Neuron, Quantum-Inspired Neural Network**

## 1. Introduction

Many neuro-physiological experiments indicate that the information processing character of the biological nerve system mainly includes the following eight aspects: the spatial aggregation, the multi-factor aggregation, the temporal cumulative effect, the activation threshold characteristic, self-adaptability, exciting and restraining characteristics, delay characteristics, conduction and output characteristics [1]. From the definition of the M-P

[*]Corresponding author.

**How to cite this paper:** Li, Z.Y. and Li, P.C. (2015) Quantum-Inspired Neural Network with Sequence Input. *Open Journal of Applied Sciences*, **5**, 259-269. http://dx.doi.org/10.4236/ojapps.2015.56027

neuron model, classical ANN preferably simulates voluminous biological neurons' characteristics such as the spatial weight aggregation, self-adaptability, conduction and output, but it does not fully incorporate temporal cumulative effect because the outputs of ANN depend only on the inputs at the moment regardless of the prior moment. In the process of practical information processing, the memory and output of the biological neuron not only depend on the spatial aggregation of input information, but also are related to the temporal cumulative effect. Although the ANN in Refs. [2]-[5] can process temporal sequences and simulate delay characteristics of biological neurons, in these models, the temporal cumulative effect has not been fully reflected. Traditional ANN can only simulate point-to-point mapping between the input space and output space. A single sample can be described as a vector in the input space and output space. However, the temporal cumulative effect denotes that multiple points in the input space are mapped to a point in the output space. A single input sample can be described as a matrix in the input space, and a single output sample is still described as a vector in the output space. In this case, we claim that the network has a sequence input.

Since Kak firstly proposed the concept of quantum-inspired neural computation [6] in 1995, quantum neural network (QNN) has attracted great attention by the international scholars during the past decade, and a large number of novel techniques have been studied for quantum computation and neural network. For example, Ref. [7] proposed the model of quantum neural network with multilevel hidden neurons based on the superposition of quantum states in the quantum theory. In Ref. [8], an attempt was made to reconcile the linear reversible structure of quantum evolution with nonlinear irreversible dynamics of neural network. Ref. [9] presented a novel learning model with qubit neuron according to quantum circuit for XOR problem and described the influence to learning by reducing the number of neurons. In Ref. [10], a new mathematical model of quantum neural network was defined, building on Deutsch's model of quantum computational network, which provides an approach for building scalable parallel computers. Ref. [11] proposed the neural network with the quantum gated nodes, and indicated that such quantum network may contain more advantageous features from the biological systems than the regular electronic devices. Ref. [12] proposed a neural network model with quantum gated nodes and a smart algorithm for it, which shows superior performance in comparison with a standard error back propagation network. Ref. [13] proposed a weightless model based on quantum circuit. It is not only quantum-inspired but also actually a quantum NN. This model is based on Grover's search algorithm, and it can both perform quantum learning and simulate the classical models. However, from all the above QNN models, like M-P neurons, it also does not fully incorporate temporal cumulative effect because a single input sample is either irrelative to time or relative to a moment instead of a period of time.

In this paper, in order to fully simulate biological neuronal information processing mechanisms and to enhance the approximation and generalization ability of ANN, we proposed a quantum-inspired neural network model with sequence input, called QNNSI. It's worth pointing out that an important issue is how to define, configure and optimize artificial neural networks. Refs. [14] [15] make deep research into this question. After repeated experiments, we opt to use a three-layer model with a hidden layer, which employs the Levenberg-Marquardt algorithm for learning. Under the premise of considering approximation ability and computational efficiency, this option is a relatively ideal. The proposed approach is utilized to the time series prediction for Mackey-Glass, and the experimental results indicate that, under a certain condition, the QNNSI is obviously superior to the common ANN.

## 2. Qubit and Quantum Gate

### 2.1. Qubit

In the quantum computers, the "qubit" has been introduced as the counterpart of the "bit" in the conventional computers to describe the states of the circuit of quantum computation. The two quantum physical states labeled as $|0\rangle$ and $|1\rangle$ express 1 bit information, in which $|0\rangle$ corresponds to the bit 0 of classical computers, while $|1\rangle$ bit 1. Notation of "$|\ \rangle$" is called the Dirac notation, which is the standard notation for the states in the quantum mechanics. The difference between bits and qubits is that a qubit can be in a state other than $|0\rangle$ and $|1\rangle$. It is also possible to form the linear combinations of the states, namely superpositions:

$$|\phi\rangle = \cos\theta|0\rangle + \sin\theta|1\rangle \tag{1}$$

An $n$ qubits system has $2^n$ computational basis states. For example, a 2 qubit system has basis $|00\rangle$, $|01\rangle$,

$|10\rangle$, $|11\rangle$. Similar to the case of a single qubit, the *n* qubits system may form the superposition of $2^n$.

$$|\phi\rangle = \sum_{x\in\{0,1\}^n} a_x |x\rangle \tag{2}$$

where $a_x$ is called probability amplitude of the basis states $|x\rangle$, and $\{0,1\}^n$ means the set of strings of length two with each letter being either zero or one. The condition that these probabilities can sum to one is expressed by the normalization condition.

$$\sum_{x\in\{0,1\}^n} |a_x|^2 = 1 \tag{3}$$

## 2.2. Quantum Rotation Gate

In quantum computation, the logic function can be realized by applying a series of unitary transform to the qubit states, which the effect of the unitary transform is equal to that of the logic gate. Therefore, the quantum services with the logic transformations in a certain interval are called the quantum gates, which are the basis of performing quantum computation.

The definition of a single qubit rotation gate is written as

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4}$$

## 2.3. Quantum NOT Gate

The NOT gate is defined by its *truth table*, in which $0 \to 1$ and $1 \to 0$, that is, the 0 and 1 states are interchanged. In fact, the quantum NOT gate acts *linearly*, that is, it takes the state $\alpha|0\rangle + \beta|1\rangle$ to the corresponding state in which the role of $|0\rangle$ and $|1\rangle$ have been interchanged, $\alpha|1\rangle + \beta|0\rangle$. There is a convenient way of representing the quantum NOT gate in matrix form, which follows directly from the linearity of quantum gates. Suppose we define a matrix $X$ to represent the quantum NOT gate as follows

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{5}$$

The notation *X* for the quantum NOT is used for historical reasons. If the quantum state $|0\rangle$ and $|1\rangle$ is written in a vector notation as $[\alpha \ \beta]^{\mathrm{T}}$, then the corresponding output from quantum NOT gate is $X[\alpha \ \beta]^{\mathrm{T}} = [\beta \ \alpha]^{\mathrm{T}}$.

## 2.4. Multi-Qubits Controlled-Not Gate

In a true quantum system, a single qubit state is often affected by a joint control of multi-qubits. A multi-qubits controlled-not gate $C^n(X)$ is a kind of control model. The multi-qubits system is also described by the wave function $|x_1 x_2 \cdots x_n\rangle$. In an (*n* + 1)-bits quantum system, when the target bit is simultaneously controlled by *n* input bits, the dynamic behavior of the system can be described by multi-qubits controlled-not gate in **Figure 1**.

In **Figure 1(a)**, suppose we have *n* + 1 qubits, and then we define the controlled operation $C^n(X)$ as follows

$$C^n(X)|x_1 x_2 \cdots x_n\rangle|\varphi\rangle = |x_1 x_2 \cdots x_n\rangle X^{x_1 x_2 \cdots x_n} |\varphi\rangle \tag{6}$$

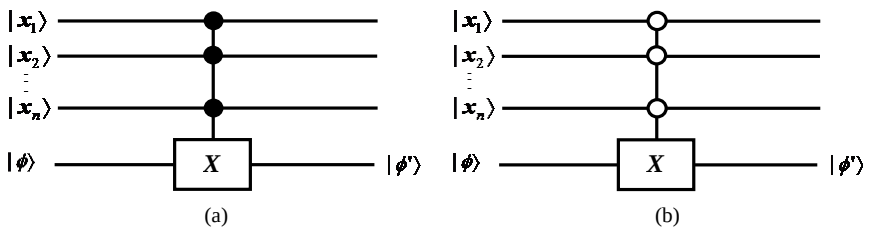


**Figure 1.** Multi-qubits controlled-not gate. (a) Type 1 control; (b) Type 0 control.

Suppose that the $|x_i\rangle = a_i|0\rangle + b_i|1\rangle$ are the control qubits, and the $|\phi\rangle = c|0\rangle + d|1\rangle$ is the target qubit. From Equation (6), the output of $C^n(X)$ is written by equation

$$\begin{aligned} & C^n(X)|x_1x_2\cdots x_n\rangle|\phi\rangle \\ & = |x_1\rangle\otimes|x_2\rangle\otimes\cdots\otimes|x_n\rangle\otimes|\phi\rangle - b_1b_2\cdots b_nc\left|\overbrace{11\cdots}^{n}10\right\rangle + b_1b_2\cdots b_nd\left|\overbrace{11\cdots}^{n}10\right\rangle \\ & + b_1b_2\cdots b_nc\left|\overbrace{11\cdots}^{n}11\right\rangle - b_1b_2\cdots b_nd\left|\overbrace{11\cdots}^{n}11\right\rangle \end{aligned} \tag{7}$$

It is observed from Equation (7) that the output of $C^n(X)$ is in the entangled state of $n+1$ qubits, and the probability of the target qubit state $|\varphi\rangle$, in which $|1\rangle$ is observed, equals to

$$P = (b_1b_2\cdots b_n)^2(c^2 - d^2) + d^2 \tag{8}$$

In **Figure 1(b)**, the operator $X$ is applied to last a qubit if the first $n$ qubits are all equal to zero, and otherwise, nothing is done. The controlled operation $C^n(X)$ can be defined by the equation

$$C^n(X)|x_1x_2\cdots x_n\rangle|\varphi\rangle = |x_1x_2\cdots x_n\rangle X^{\overline{x_1x_2\cdots x_n}}|\phi\rangle \tag{9}$$

The probability of the target qubit state $|\varphi\rangle$, in which $|1\rangle$ is observed, equals to

$$P = (a_1a_2\cdots a_n)^2(c^2 - d^2) + d^2 \tag{8}$$

At this time, after the joint control of the $n$ input bits, the target bit $|\phi'\rangle$ can be defined as follows

$$|\phi'\rangle = \sqrt{1-P}|0\rangle + \sqrt{P}|1\rangle \tag{9}$$

## 3. QNNSI Model

### 3.1. Quantum-Inspired Neuron Model

In this section, we first propose a quantum-inspired neuron model with sequence input, as shown in **Figure 2**. This model consists of quantum rotation gates and multi-qubits controlled-not gate. The $\{|x_i(t_r)\rangle\}$ defined in time domain interval $[0, T]$ denote the input sequences. The output is the probability amplitude of the target state in $|1\rangle$. The control parameters are the rotation angles $\bar{\theta}_i(t_r)$.

Let $|x_i(t_r)\rangle = \cos\theta_i(t_r)|0\rangle + \sin\theta_i(t_r)|1\rangle$, $|\phi(t_1)\rangle = |0\rangle$. In this paper, we define the output of the quantum neuron as the probability amplitude of the corresponding state, in which $|1\rangle$ is observed. According to the definition of quantum rotation gate and multi-qubits controlled-not gate, the output of quantum neuron can be written as

$$y = \sqrt{S_q\left(\cos 2\varphi(t_{q-1}) + \sin^2\varphi(t_{q-1})\right)} \tag{10}$$

where $\varphi(t_r) = \arcsin\left(\sqrt{S_r\cos 2\varphi(t_{r-1}) + \sin^2 t_{r-1}}\right)$, $r = 2,3,\cdots,q$, for **Figure 2(a)**, $S_r = \prod_{i=1}^{n}\sin^2\left(\theta_i(t_r) + \bar{\theta}_i(t_r)\right)$,

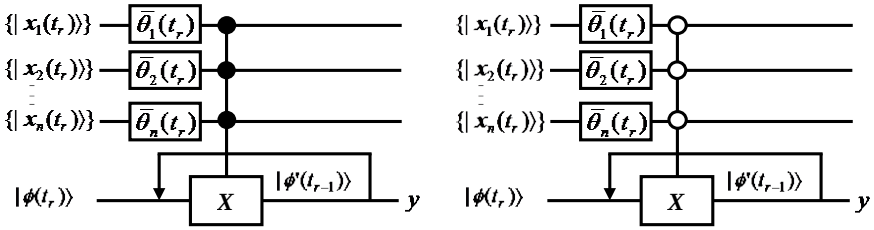


**Figure 2.** The model of quantum-inspired neuron with sequence input. (a) Type 1 control; (b) Type 0 control.

for **Figure 2(b)**, $S_r = \prod_{i=1}^{n} \cos^2\left(\theta_i(t_r) + \bar{\theta}_i(t_r)\right)$.

### 3.2. Quantum-Inspired Neural Network Model

In this paper, the QNNSI model is shown in **Figure 3**, where the hidden layer consists of quantum-inspired neurons with sequence input, and the output layer consists of classical neurons. The $\left\{\left|x_i(t_r)\right\rangle\right\}$ denote the input sequences, the $h_1, h_2, \cdots, h_p$ denote the hidden output, the $\omega_{jk}$ denotes the connection weights in output layer, and the $y_1, y_2, \cdots, y_m$ denote the network output. The Sigmoid function is used in output layer.

Unlike ANN, each input sample of QNNSI is described as a matrix instead of a vector. For example, the *l*-th sample can be written as

$$\begin{bmatrix} \left\{\left|x_1^l(t_r)\right\rangle\right\} \\ \left\{\left|x_2^l(t_r)\right\rangle\right\} \\ \vdots \\ \left\{\left|x_n^l(t_r)\right\rangle\right\} \end{bmatrix} = \begin{bmatrix} \left|x_1^l(t_1)\right\rangle & \left|x_1^l(t_2)\right\rangle & \cdots & \left|x_1^l(t_q)\right\rangle \\ \left|x_2^l(t_1)\right\rangle & \left|x_2^l(t_2)\right\rangle & \cdots & \left|x_2^l(t_q)\right\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \left|x_n^l(t_1)\right\rangle & \left|x_n^l(t_1)\right\rangle & \cdots & \left|x_n^l(t_q)\right\rangle \end{bmatrix} \tag{11}$$

Let $\left|\varphi_j^l(t_1)\right\rangle = 0,\ j = 1, 2, \cdots, p$, $\bar{h}_{jr} = \begin{cases} \prod_{i=1}^{n} \sin\left(\theta_i^l(t_r) + \theta_{ij}(t_r)\right), & j = 1, 3, 5, \cdots \\ \prod_{i=1}^{n} \cos\left(\theta_i^l(t_r) + \theta_{ij}(t_r)\right), & j = 2, 4, 6, \cdots \end{cases}$, According to the input/output relationship of quantum-inspired neuron, in interval $[0, t_r]$, the spatial and temporal aggregation results of the *j*-th quantum-inspired neuron in hidden layer can be written as

$$\begin{cases} h_j^l(t_1) = \bar{h}_{j1}^l \\ h_j^l(t_r) = \sqrt{\left(\bar{h}_{jr}^l\right)^2 \left(1 - 2\left(h_j^l(t_{r-1})\right)^2\right) + \left(h_j^l(t_{r-1})\right)^2} \end{cases} \tag{12}$$

The *j*-th output in hidden layer (namely, the spatial and temporal aggregation results in [0, *T*]) is given by

$$h_j^l = h_j^l(t_q) \tag{13}$$

The *k*-th output in output layer can be written as

$$y_k^l = \left(1 + \mathrm{e}^{-\sum_{j=1}^{p} \omega_{jk} h_j^l}\right)^{-1} \tag{14}$$

## 4. QNNSI Algorithm

### 4.1. Pretreatment of Input and Output Samples

Suppose the *l*-th sample in *n*-dimensional input space $\left\{\bar{X}^l(t_r)\right\} = \left[\bar{x}_1^l(t_r), \bar{x}_2^l(t_r), \cdots, \bar{x}_n^l(t_r)\right]^{\mathrm{T}}$, where $r = 1, 2, \cdots, q$, $l = 1, 2, \cdots, L$. Let

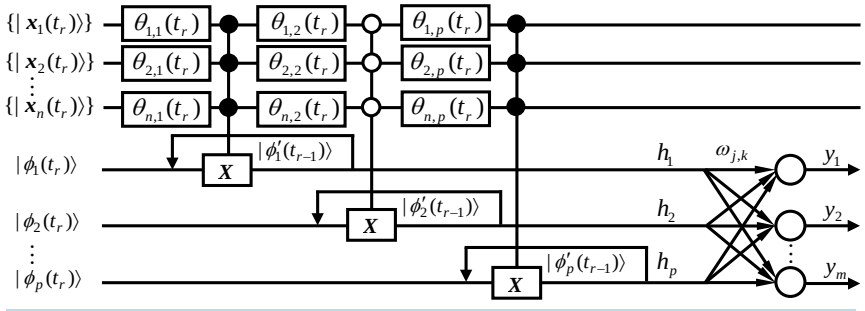


**Figure 3.** The model of quantum-inspired neural network with sequence input.

$$\begin{cases} \mathrm{Max}_{ir} = \max\left(\overline{x}_i^1(t_r), \overline{x}_i^2(t_r), \cdots, \overline{x}_i^L(t_r)\right) \\ \mathrm{Min}_{ir} = \min\left(\overline{x}_i^1(t_r), \overline{x}_i^2(t_r), \cdots, \overline{x}_i^L(t_r)\right) \end{cases} \tag{15}$$

$$\theta_i^l(t_r) = \begin{cases} \dfrac{\overline{x}_i^l(t_r) - \mathrm{Min}_{ir}}{\mathrm{Max}_{ir} - \mathrm{Min}_{ir}} \dfrac{\pi}{2} & \text{if } \mathrm{Max}_{ir} > \mathrm{Min}_{ir} \\ \dfrac{\pi}{2} & \text{if } \mathrm{Max}_{ir} = \mathrm{Min}_{ir} \neq 0 \\ 0 & \text{if } \mathrm{Max}_{ir} = \mathrm{Min}_{ir} = 0 \end{cases} \tag{16}$$

These samples can be converted into the quantum states as follows

$$\left\{\left|\overline{X}^l(t_r)\right\rangle\right\} = \left[\left\{\left|\overline{x}_1^l(t_r)\right\rangle\right\}, \left\{\left|\overline{x}_2^l(t_r)\right\rangle\right\}, \cdots, \left\{\left|\overline{x}_n^l(t_r)\right\rangle\right\}\right]^{\mathrm{T}} \tag{17}$$

where $\left|x_i^l(t_r)\right\rangle = \cos\left(\theta_i^l(t_r)\right)|0\rangle + \cos\left(\theta_i^l(t_r)\right)|1\rangle$.

Similarly, suppose the $l$-th output sample $\left\{\overline{Y}^l\right\} = \left[\left\{\overline{y}_1^l\right\}, \left\{\overline{y}_2^l\right\}, \cdots, \left\{\overline{y}_m^l\right\}\right]^{\mathrm{T}}$, where $l = 1, 2, \cdots, L$. Let

$$\begin{cases} \mathrm{Max}_k = \max\left(y_k^1, y_k^2, \cdots, \overline{x}_k^L\right) \\ \mathrm{Min}_k = \min\left(y_k^1, y_k^2, \cdots, \overline{x}_k^L\right) \end{cases} \tag{18}$$

then, these output samples can be normalized by the following equation

$$y_k^l = \begin{cases} \dfrac{\overline{y}_k^l - \mathrm{Min}_k}{\mathrm{Max}_k - \mathrm{Min}_k} & \text{if } \mathrm{Max}_{ir} > \mathrm{Min}_{ir} \\ 1 & \text{if } \mathrm{Max}_{ir} = \mathrm{Min}_{ir} \neq 0 \\ 0 & \text{if } \mathrm{Max}_{ir} = \mathrm{Min}_{ir} = 0 \end{cases} \tag{19}$$

## 4.2. QNNSI Parameters Adjustment

In QNNSI, the adjustable parameters include the rotation angles of quantum rotation gates in hidden layer, and the weights in output layer. Suppose $\overline{y}_1^l, \overline{y}_2^l, \cdots, \overline{y}_m^l$ denote the normalized desired outputs of the $l$-th sample, and $y_1^l, y_2^l, \cdots, y_m^l$ denote the corresponding actual outputs. The evaluation function is defined as follows

$$E = \max_{1\le l\le L}\max_{1\le k\le m}\left|e_k^l\right| = \max_{1\le l\le L}\max_{1\le k\le m}\left|\overline{y}_k^l - y_k^l\right| \tag{20}$$

Let

$$\begin{cases} \overline{h}_{jr}^l = \begin{cases} \prod_{i=1}^n \sin\left(\theta_i^l(t_r) + \theta_{ij}(t_r)\right) & j = 1, 2, 5, \cdots \\ \prod_{i=1}^n \cos\left(\theta_i^l(t_r) + \theta_{ij}(t_r)\right) & j = 2, 4, 6, \cdots \end{cases} \\ S_{jr}^l = \left(\overline{h}_{jr}^l\right)^2 \left(1 - 2\left(h_j^l(t_{r-1})\right)^2\right) \end{cases} \tag{21}$$

According to the gradient descent algorithm in Ref. [16], the gradient of the rotation angles of the quantum rotation gates can be calculated as follows

$$\frac{\partial e_k^l}{\partial \theta_{ij}(t_r)} = -y_k^l\left(1 - y_k^l\right)\omega_{jk}\prod_{s=r+1}^q\left(1 - 2\left(\overline{h}_{js}^l\right)^2\right)h_j^l(t_{s-1})\left(\overline{h}_{jr}^l\right)^2\left(1 - 2\left(h_j^l(t_{r-1})\right)^2\right)\cot\left(\theta_i^l(t_r)\right)\Big/\prod_{s=r}^q h_j^l(t_s) \tag{22}$$

where $j = 1, 2, \cdots, p$; $k = 1, 2, \cdots, m$; $r = 1, 2, \cdots, q$; $l = 1, 2, \cdots, L$.

The gradient of the connection weights in output layer can be calculated as follows

$$\frac{\partial e_k^l}{\partial \omega_{jk}} = -y_k^l \left(1 - y_k^l\right) h_j^l \left(t_q\right) \tag{23}$$

Because gradient calculation is more complicated, the standard gradient descent algorithm is not easy to converge. Hence we employ the *Levenberg-Marquardt* algorithm in Ref. [16] to adjust the QNNSI parameters.

Let **P** denote the parameter vector, **e** denote the error vector, and **J** denote the Jacobian matrix. **p**, **e**, and **J** are respectively defined as follows

$$\mathrm{P}^{\mathrm{T}} = \left[\theta_{11}\left(t_1\right), \theta_{11}\left(t_2\right) \cdots, \theta_{np}\left(t_q\right), \omega_{11}, \omega_{12}, \cdots, \omega_{pm}\right] \tag{24}$$

$$\mathrm{e}^{\mathrm{T}}\left(\mathrm{P}\right) = \left[e_1^1, \cdots, e_m^1, \cdots, e_1^L, \cdots, e_m^L\right] \tag{25}$$

$$\mathrm{J}\left(\mathrm{P}\right) = \begin{bmatrix} \frac{\partial e_1^1}{\partial \theta_{11}\left(t_1\right)} & \cdots & \frac{\partial e_1^1}{\partial \theta_{11}\left(t_1\right)} & \frac{\partial e_1^1}{\partial \omega_{11}} & \cdots & \frac{\partial e_1^1}{\partial \omega_{pm}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_m^1}{\partial \theta_{11}\left(t_1\right)} & \cdots & \frac{\partial e_m^1}{\partial \theta_{11}\left(t_1\right)} & \frac{\partial e_m^1}{\partial \omega_{11}} & \cdots & \frac{\partial e_m^1}{\partial \omega_{pm}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_1^L}{\partial \theta_{11}\left(t_1\right)} & \cdots & \frac{\partial e_1^L}{\partial \theta_{11}\left(t_1\right)} & \frac{\partial e_1^L}{\partial \omega_{11}} & \cdots & \frac{\partial e_1^L}{\partial \omega_{pm}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_m^L}{\partial \theta_{11}\left(t_1\right)} & \cdots & \frac{\partial e_m^L}{\partial \theta_{11}\left(t_1\right)} & \frac{\partial e_m^L}{\partial \omega_{11}} & \cdots & \frac{\partial e_m^L}{\partial \omega_{pm}} \end{bmatrix} \tag{26}$$

According to *Levenberg-Marquardt* algorithm, the QNNSI iterative equation is written as follows

$$\mathrm{P}_{t+1} = \mathrm{P}_t - \left(\mathrm{J}^{\mathrm{T}}\left(\mathrm{P}_t\right)\mathrm{J}\left(\mathrm{P}_t\right) + \mu_t \mathrm{I}\right)^{-1} \mathrm{J}^{\mathrm{T}}\left(\mathrm{P}_t\right)\mathrm{e}\left(\mathrm{P}_t\right) \tag{27}$$

where $t$ denotes the iterative steps, $\mathrm{I}$ denotes the unit matrix, and $\mu_t$ is a small positive number to ensure the matrix $\mathrm{J}^{\mathrm{T}}\left(\mathrm{P}_t\right)\mathrm{J}\left(\mathrm{P}_t\right) + \mu_t \mathrm{I}$ invertible.

### 4.3. Stopping Criterion of QNNSI

If the value of the evaluation function **E** reaches the predefined precision within the preset maximum of iterative steps, then the execution of the algorithm is stopped, else the algorithm is not stopped until it reaches the predefined maximum of iterative steps.

## 5. Simulations

To examine the effectiveness of the proposed QNNSI, the time series prediction for Mackey-Glass is used to compare it with the ANN with a hidden layer in this section. In this experiment, we implement and investigate the QNNSI in Matlab (Version 7.1.0.246) on a Windows PC with 2.19 GHz CPU and 1.00 GB RAM. Our QNNSI has the same structure and parameters as the ANN in the simulations, and the same *Levenberg-Marquardt* algorithm [16] is applied in two models.

Mackey-Glass time series can be generated by the following iterative equation [17]

$$x\left(t+1\right) - x\left(t\right) = \frac{ax\left(t-\tau\right)}{1 + x^{10}\left(t-\tau\right)} - bx\left(t\right) \tag{28}$$

where $t$ and $\tau$ are integers, $a = 0.2$, $b = 0.3$, $\tau = 17$, and $x\left(0\right) \in \left(0,1\right)$.

From the above equation, we may obtain the time sequence $\left\{x\left(t\right)\right\}_{t=1}^{1000}$. We take the first 800 as the training set, and the remaining 200 as the testing set. Our prediction schemes is to employ $n$ data adjacent to each other to predict the next one data. Namely, in our model, the sequence length equals to $n$. Therefore, each sample con-

sists of $n$ input values and an output value.

Hence, there is only one output node in QNNSI and ANN. In order to fully compare the approximation ability of two models, the number of hidden nodes are respectively set to $10, 11, \cdots, 40$. The predefined precision is set to 0.05, and the maximum of iterative steps is set to 100. The QNNSI rotation angles in hidden layer are initialized to random numbers in $(-\pi/2, \pi/2)$, and the connection weights in output layer are initialized to random numbers in (−1, 1). For ANN, all weights are initialized to random numbers in (−1, 1), and the *Sigmoid* functions are used as the activation functions in hidden layer and output layer.

Obviously, ANN has $n$ input nodes, and an ANN's input sample can be described as a $n$-dimensional vector. For the number of input nodes of QNNSI, we employ the following nine kinds of settings shown in **Table 1**. For each of these settings in **Table 1**, a single QNNSI input sample can be described as a matrix.

It is worth noting that, in QNNSI, an $n \times q$ matrix can be used to describe a single sequence sample. In general, ANN cannot deal directly with a single $n \times q$ sequence sample. In ANN, an $n \times q$ matrix is usually regarded as $nq$ dimensional vector samples. For fair comparison, in ANN, we have expressed the $n \times q$ sequence samples into the $nq$ dimensional vector samples. Therefore, in **Table 1**, the sequence lengths for ANN are not changed. It is clear that, in fact, there is only one kind of ANN in **Table 1**, namely, ANN36.

Our experiment scheme is that, for each kind of combination of input nodes and hidden nodes, one ANN and nine QNNSIs are respectively run 10 times. Then we use four indicators, such as *average approximation error*, *average iterative steps*, *average running time*, and *convergence ratio*, to compare QNNSI with ANN. Training result contrasts are shown in **Tables 2-5**, where QNNSIn_q denotes QNNSI with $n$ input nodes and $q$ sequence length.

From **Tables 2-5**, we can see that when the input nodes take 6, 9, and 12, the performance of QNNSIs are

**Table 1.** The input nodes and the sequence length setting of QNNSIs and ANN.

| QNNSI | | ANN | |
|---|---|---|---|
| Input nodes | Sequence length | Input nodes | Sequence length |
| 1 | 36 | 36 | 1 |
| 2 | 18 | 36 | 1 |
| 3 | 12 | 36 | 1 |
| 4 | 9 | 36 | 1 |
| 6 | 6 | 36 | 1 |
| 9 | 4 | 36 | 1 |
| 12 | 3 | 36 | 1 |
| 18 | 2 | 36 | 1 |
| 36 | 1 | 36 | 1 |

**Table 2.** Training result contracts of average approximation error.

| QNNSI | Hidden nodes | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 |
| QNNSI1_36 | 0.55 | 0.55 | 0.59 | 0.59 | 0.59 | 0.59 | 0.59 | 0.67 | 0.67 | 0.59 | 0.71 | 0.63 | 0.67 | 0.63 | 0.67 | 0.68 |
| QNNSI2_18 | 0.55 | 0.54 | 0.51 | 0.25 | 0.13 | 0.14 | 0.14 | 0.31 | 0.32 | 0.13 | 0.41 | 0.13 | 0.32 | 0.23 | 0.41 | 0.41 |
| QNNSI3_12 | 0.04 | 0.04 | 0.13 | 0.13 | 0.13 | 0.04 | 0.13 | 0.32 | 0.32 | 0.13 | 0.32 | 0.13 | 0.32 | 0.22 | 0.41 | 0.41 |
| QNNSI4_9 | 0.04 | 0.04 | 0.13 | 0.13 | 0.04 | 0.13 | 0.13 | 0.22 | 0.32 | 0.04 | 0.22 | 0.04 | 0.22 | 0.22 | 0.32 | 0.31 |
| QNNSI6_6 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 |
| QNNSI9_4 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 |
| QNNSI12_3 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 |
| QNNSI18_2 | 0.17 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.05 | 0.04 |
| QNNSI36_1 | 0.47 | 0.47 | 0.47 | 0.47 | 0.47 | 0.47 | 0.47 | 0.47 | 0.48 | 0.47 | 0.47 | 0.47 | 0.47 | 0.47 | 0.47 | 0.47 |
| ANN36 | 0.23 | 0.14 | 0.41 | 0.14 | 0.23 | 0.23 | 0.14 | 0.32 | 0.32 | 0.14 | 0.32 | 0.05 | 0.32 | 0.23 | 0.41 | 0.32 |

**Table 3.** Training result contracts of average iterative steps.

| QNNSI | Hidden nodes | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 |
| QNNSI1_36 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| QNNSI2_18 | 100 | 100 | 92.5 | 76.8 | 57.1 | 52.6 | 42.5 | 51.4 | 49.6 | 32.8 | 51.7 | 26.2 | 43.8 | 36.8 | 51.0 | 51.1 |
| QNNSI3_12 | 8.90 | 8.20 | 16.6 | 16.5 | 15.2 | 6.70 | 15.0 | 33.5 | 33.6 | 14.4 | 33.1 | 14.1 | 33.4 | 23.9 | 42.5 | 42.5 |
| QNNSI4_9 | 5.60 | 5.40 | 14.8 | 14.5 | 4.70 | 14.1 | 13.8 | 23.7 | 33.0 | 4.20 | 23.5 | 4.10 | 23.6 | 23.1 | 33.0 | 32.9 |
| QNNSI6_6 | 5.4 | 5.5 | 4.9 | 5.1 | 4.9 | 4.6 | 4.4 | 4.5 | 4.2 | 4.1 | 4.1 | 4.0 | 4.1 | 4.0 | 6.2 | 4.5 |
| QNNSI9_4 | 6.6 | 6.0 | 6.0 | 5.5 | 5.8 | 5.3 | 5.3 | 5.2 | 5.0 | 4.6 | 4.6 | 4.7 | 4.4 | 4.2 | 4.4 | 4.6 |
| QNNSI12_3 | 10 | 7.7 | 6.6 | 7.2 | 6.7 | 6.1 | 6.2 | 6.2 | 5.9 | 6.2 | 5.9 | 5.8 | 6.1 | 5.7 | 5.6 | 5.5 |
| QNNSI18_2 | 52.9 | 32.2 | 33.0 | 19.0 | 20.6 | 16.9 | 11.3 | 10.8 | 10.8 | 9.90 | 9.50 | 9.10 | 8.70 | 9.10 | 7.60 | 8.10 |
| QNNSI36_1 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| ANN36 | 32.9 | 21.0 | 48.3 | 20.5 | 30.2 | 29.6 | 20.6 | 38.2 | 45.4 | 20.2 | 38.2 | 10.0 | 37.0 | 5.50 | 46.8 | 36.9 |

**Table 4.** Training result contracts of average running time (s).

| QNNSI | Hidden nodes | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 |
| QNNSI1_36 | 99 | 124 | 149 | 180 | 210 | 244 | 279 | 321 | 359 | 396 | 435 | 489 | 538 | 600 | 657 | 722 |
| QNNSI2_18 | 83 | 103 | 117 | 117 | 104 | 111 | 108 | 146 | 158 | 123 | 209 | 126 | 223 | 212 | 316 | 350 |
| QNNSI3_12 | 9 | 11 | 23 | 27 | 30 | 19 | 40 | 92 | 104 | 57 | 131 | 70 | 164 | 135 | 253 | 281 |
| QNNSI4_9 | 7 | 9 | 21 | 26 | 14 | 33 | 39 | 69 | 97 | 21 | 89 | 25 | 112 | 122 | 188 | 208 |
| QNNSI6_6 | 7 | 9 | 10 | 12 | 14 | 15 | 17 | 19 | 21 | 24 | 26 | 27 | 31 | 33 | 51 | 44 |
| QNNSI9_4 | 7 | 9 | 11 | 12 | 15 | 17 | 18 | 21 | 23 | 25 | 27 | 30 | 32 | 33 | 38 | 43 |
| QNNSI12_3 | 9 | 9 | 10 | 13 | 15 | 16 | 19 | 22 | 24 | 28 | 30 | 33 | 37 | 40 | 42 | 44 |
| QNNSI18_2 | 37 | 30 | 37 | 28 | 35 | 35 | 29 | 32 | 37 | 38 | 42 | 45 | 49 | 55 | 53 | 61 |
| QNNSI36_1 | 69 | 88 | 109 | 131 | 150 | 176 | 204 | 235 | 269 | 306 | 346 | 389 | 436 | 486 | 540 | 598 |
| ANN36 | 13 | 13 | 32 | 19 | 32 | 38 | 33 | 66 | 89 | 50 | 101 | 37 | 128 | 139 | 203 | 182 |

**Table 5.** Training result contracts of convergence ratio (%).

| QNNSI | Hidden nodes | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 |
| QNNSI1_36 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| QNNSI2_18 | 0 | 0 | 20 | 60 | 90 | 90 | 90 | 70 | 70 | 90 | 60 | 90 | 70 | 80 | 60 | 60 |
| QNNSI3_12 | 100 | 100 | 90 | 90 | 90 | 100 | 90 | 70 | 70 | 90 | 70 | 90 | 70 | 80 | 60 | 60 |
| QNNSI4_9 | 100 | 100 | 90 | 90 | 100 | 90 | 90 | 80 | 70 | 100 | 80 | 100 | 80 | 80 | 70 | 70 |
| QNNSI6_6 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| QNNSI9_4 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| QNNSI12_3 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| QNNSI18_2 | 70 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| QNNSI36_1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ANN36 | 80 | 90 | 60 | 90 | 80 | 80 | 90 | 70 | 70 | 90 | 70 | 100 | 70 | 80 | 60 | 70 |

obviously superior to that of ANN, and the QNNSIs have better stability than ANN when the number of hidden nodes changes.

Next, we investigate the generalization ability of QNNSI. Based on the above experimental results, we only investigate three QNNSIs (QNNSI6_6, QNNSI9_4, and QNNSI12_3). Our experiment scheme is that three QNNSIs and one ANN train 10 times on the training set, and the generalization ability is immediately investigated on the testing set after each training. The average results of the 10 tests are regarded as the evaluation indexes. For convenience of description, let $\overline{E}_{avg}$ denote the average of the maximum prediction error, $\overline{E}_{mean}$ denote the average of the prediction error mean, and $\overline{E}_{\mathrm{var}}$ denote the average of prediction error variance.

Taking 30 hidden nodes for example, the evaluation indexes contrast of QNNSIs and ANN are shown in **Table 6**. The experimental results show that the generalization ability of three QNNSIs is obviously superior to that of corresponding ANN.

These experimental results can be explained as follows. For processing of input information, QNNSI and ANN take different approaches. QNNSI directly receives a discrete input sequence. In QNNSI, using quantum information processing mechanism, the input is circularly mapped to the output of quantum controlled-not gates in hidden layer. As the controlled-not gate's output is in the entangled state of multi-qubits, therefore, this mapping is highly nonlinear, which make QNNSI have the stronger approximation ability. In addition, QNNSI's each input sample can be described as a matrix with $n$ rows and $q$ columns. It is clear from QNNSI's algorithm that, for the different combination of $n$ and $q$, the output of quantum-inspired neuron in hidden layer is also different. In fact, The number of discrete points $q$ denotes the *depth* of pattern memory, and the number of input nodes $n$ denotes the *breadth* of pattern memory. When the *depth* and the *breadth* are appropriately matched, the QNNSI shows excellent performance. For the ANN, because its input can only be described as a $nq$-dimensional vector, it is not directly deal with a discrete input sequence. Namely, it only can obtain the sample characteristics by way of *breadth* instead of *depth*. Hence, in the ANN information processing, there exists inevitably the loss of sample characteristics, which affects its approximation and generalization ability.

It is worth pointing out that QNNSI is potentially much more computationally efficient than all the models referenced above in the Introduction section. The efficiency of many quantum algorithms comes directly from quantum parallelism that is a fundamental feature of many quantum algorithms. Heuristically, and at the risk of over-simplifying, quantum parallelism allows quantum computers to evaluate a function $f(x)$ for many different values of $x$ simultaneously. Although quantum simulation requires many resources in general, quantum parallelism leads to very high computational efficiency by using the superposition of quantum states. In QNNSI, the input samples have been converted into corresponding quantum superposition states after preprocessing. Hence, as far as a lot of quantum rotation gates and controlled-not gates used in QNNSI are concerned, information processing can be performed simultaneously, which greatly improves the computational efficiency. Because the above experiments are performed in classical computer, the quantum parallelism has not been explored. However, the efficient computational ability of QNNSI is bound to stand out in future quantum computer.

## 6. Conclusion

This paper proposes quantum-inspired neural network model with sequence input based on the principle of quantum computing. The architecture of the proposed model includes three layers, where the hidden layer consists of quantum-inspired neurons and the output layer consists of classical neurons. An obvious difference from classical ANN is that each dimension of a single input sample consists of a discrete sequence rather than a single value. The activation function of hidden layer is redesigned according to the principle of quantum computing. The *Levenberg-Marquardt* algorithm is employed for learning. With application of the information processing mechanism of quantum rotation gates and controlled-not gates, the proposed model can effectively obtain the sample

**Table 6.** The average prediction error contrasts of QNNSIs and ANN.

| QNNSI | | | | ANN | | | |
|---|---|---|---|---|---|---|---|
| Model | $\overline{E}_{avg}$ | $\overline{E}_{mean}$ | $\overline{E}_{\mathrm{var}}$ | Model | $\overline{E}_{avg}$ | $\overline{E}_{mean}$ | $\overline{E}_{\mathrm{var}}$ |
| QNNSI6_6 | 0.0520 | 0.0084 | 0.0001 | ANN36 | 0.3334 | 0.1598 | 0.0185 |
| QNNSI9_4 | 0.0541 | 0.0089 | 0.0001 | ANN36 | 0.3334 | 0.1598 | 0.0185 |
| QNNSI12_3 | 0.0566 | 0.0093 | 0.0001 | ANN36 | 0.3334 | 0.1598 | 0.0185 |

characteristics by ways of *breadth* and *depth*. The experimental results reveal that a greater difference between input nodes and sequence length leads to a lower performance of proposed model than that of classical ANN; on the contrary, it obviously enhances approximation and generalization ability of proposed model when input nodes are closer to sequence length. The following issues of the proposed model, such as continuity, computational complexity, and improvement of learning algorithm, are subjects of further research.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 61170132), Natural Science Foundation of Heilongjiang Province of China (Grant No. F2015021), Science Technology Research Project of Heilongjiang Educational Committee of China (Grant No. 12541059), and Youth Foundation of Northeast Petroleum University (Grant No. 2013NQ119).

## References

[1] Tsoi, A.C. and Back, A.D. (1994) Locally Recurrent Globally Feed Forward Network: A Critical Review of Architectures. *IEEE Transactions on Neural Network*, **7**, 229-239. http://dx.doi.org/10.1109/72.279187

[2] Kleinfeld, A.D. (1986) Sequential State Generation by Model Neural Network. *Proceedings of the National Academy of Sciences USA*, **83**, 9469-9473. http://dx.doi.org/10.1073/pnas.83.24.9469

[3] Waibel, A., Hanazawa, A. and Hinton, A. (1989) Phoneme Recognition Using Time-Delay Neural Network. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**, 328-339. http://dx.doi.org/10.1109/29.21701

[4] Lippmann, R.P. (1989) Review of Neural Network for Speech Recognition. *Neural Computation*, **1**, 1-38. http://dx.doi.org/10.1162/neco.1989.1.1.1

[5] Maria, M., Marios, A. and Chris, C. (2011) Artificial Neural Network for Earthquake Prediction Using Time Series Magnitude Data or Seismic Electric Signals. *Expert Systems with Applications*, **38**, 15032-15039. http://dx.doi.org/10.1016/j.eswa.2011.05.043

[6] Kak, S. (1995) On Quantum Neural Computing. *Information Sciences*, **83**, 143-160. http://dx.doi.org/10.1016/0020-0255(94)00095-S

[7] Gopathy, P. and Nicolaos, B.K. (1997) Quantum Neural Network (QNN's): Inherently Fuzzy Feed forward Neural Network. *IEEE Transactions on Neural Network*, **8**, 679-693. http://dx.doi.org/10.1109/72.572106

[8] Zak, M. and Williams, C.P. (1998) Quantum Neural Nets. *International Journal of Theoretical Physics*, **3**, 651-684. http://dx.doi.org/10.1023/A:1026656110699

[9] Maeda, M., Suenaga, M. and Miyajima, H. (2007) Qubit Neuron According to Quantum Circuit for XOR Problem. *Applied Mathematics and Computation*, **185**, 1015-1025. http://dx.doi.org/10.1016/j.amc.2006.07.046

[10] Gupta, S. and Zia, R.K. (2001) Quantum Neural Network. *Journal of Computer and System Sciences*, **63**, 355-383. http://dx.doi.org/10.1006/jcss.2001.1769

[11] Shafee, F. (2007) Neural Network with Quantum Gated Nodes. *Engineering Applications of Artificial Intelligence*, **20**, 429-437. http://dx.doi.org/10.1016/j.engappai.2006.09.004

[12] Li, P.C., Song, K.P. and Yang, E.L. (2010) Model and Algorithm of Neural Network with Quantum Gated Nodes. *Neural Network World*, **11**, 189-206.

[13] Adenilton, J., Wilson, R. and Teresa, B. (2012) Classical and Superposed Learning for Quantum Weightless Neural Network. *Neurocomputing*, **75**, 52-60. http://dx.doi.org/10.1016/j.neucom.2011.03.055

[14] Israel, G.C., Angel, G.C. and Belen, R.M. (2012) Dealing with Limited Data in Ballistic Impact Scenarios: An Empirical Comparison of Different Neural Network Approaches. *Applied Intelligence*, **35**, 89-109.

[15] Israel, G.C., Angel, G.C. and Belen, R.M. (2013) An Optimization Methodology for Machine Learning Strategies and Regression Problems in Ballistic Impact Scenarios. *Applied Intelligence*, **36**, 424-441.

[16] Martin, T.H., Howard, B.D. and Mark, H.B. (1996) Neural Network Design. PWS Publishing Company, Boston, 391-399.

[17] Mackey, M.C. and Glass, L. (1977) Oscillation and Chaos in Physiological Control System. *Science*, **197**, 287-289. http://dx.doi.org/10.1126/science.267326